Accepted Publication to the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)
October 27-30, 2019, Auckland, New Zealand

How to Match Tracks of Visual Features for Automotive Long-Term SLAM

Stefan Luthardt[*], Christoph Ziegler[*], Volker Willert[*], and Jürgen Adamy[*]

***Abstract*— Accurate localization is a vital prerequisite for future assistance or autonomous driving functions in intelligent vehicles. To achieve the required localization accuracy and availability, long-term visual SLAM algorithms like LLama-SLAM are a promising option. In such algorithms visual feature tracks, i. e. landmark observations over several consecutive image frames, have to be matched to feature tracks recorded days, weeks or months earlier. This leads to a more challenging matching problem than in short-term visual localization and known descriptor matching methods cannot be applied directly. In this paper, we devise several approaches to compare and match feature tracks and evaluate their performance on a long-term data set. With the proposed descriptor combination and masking ("CoMa") method the best track matching performance is achieved with minor computational cost. This method creates a single combined descriptor for each feature track and furthermore increases the robustness by capturing the appearance variations of this track in a descriptor mask.**

I. Introduction

An accurate estimation of the vehicle's position is an essential requirement for current and future intelligent vehicles. Only with a precisely known position, valuable information provided by digital maps or by other vehicles can be utilized. A promising approach for vehicle localization are V-SLAM (Visual Simultaneous Localization and Mapping) methods which can deliver superior accuracy and availability compared to GNSS (global navigation satellite system) positioning [1]. To determine the position of the camera respectively of the vehicle, V-SLAM algorithms need to find prominent points in the camera image and associate them with known points from previous images as shown in Fig. 1. These prominent points in the camera image are called **visual features** and detecting and matching them is one of the key issues in V-SLAMs and similar methods [2]–[4].

A. Visual Features

In general, a visual feature is defined as a point in the image that uniquely differs in intensity or texture from its immediate neighbors. In visual localization usually corners are chosen as visual features, i. e. locations in the image where there is a high intensity gradient in both directions [4]. To extract potential visual features from an image, a detector algorithm like FAST [5] is used. The result of the detector algorithm is a set of key points, where a **key point** $\mathbf{p}$ is the pixel position of the visual feature.

The detected key points then need to be matched to features from previous images, i. e. it has to be determined which features from two or more images belong to the same real 3D-point in the world (see Fig. 1). Hereafter, we call the real 3D-points **landmarks**, whereas we use the term feature to specify the projection of a landmark in a camera image. To find features that belong together, the surroundings of the key points are considered. The surrounding of a key point $\mathbf{p}$ is called **patch** and is usually limited to a certain size, e. g. 51 px. Theoretically, features could be matched by directly comparing their patches, but this is very costly. Therefore, **descriptor** methods are used to convert the patch to a vector $\mathbf{x}$, which can be seen as a compressed version of the patch. Descriptors aim to capture all relevant characteristics of the patch while being invariant to minor transformations and being robust to illumination changes and noise. Furthermore, the descriptor vector $\mathbf{x}$ should be compact and easy to compare. A visual feature θ is therefore composed of a key point $\mathbf{p}$ and a descriptor $\mathbf{x}$ describing its surrounding patch, i. e. $\theta = \{\mathbf{p}, \mathbf{x}\}$. Using the descriptors, features can be easily compared by computing a descriptor distance $\mathrm{d}_*(\mathbf{x}_i, \mathbf{x}_j)$ and thereby matching feature pairs can be determined using some matching criteria. There is a large variety of descriptor methods and we refer the reader to [6], [7] or [8] for a comprehensive overview.

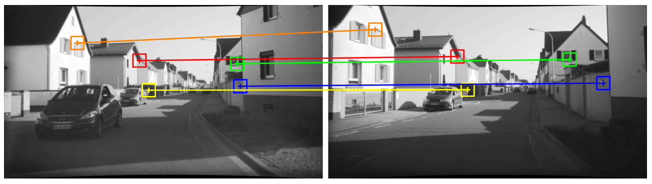


Fig. 1. Detecting and matching visual features is a key issue in visual localization methods. Here, this issue is illustrated for a simple example with two image frames. The key points of each visual feature are marked with a plus and a box indicates the corresponding patch. Matching features, i. e. features that belong to the same 3D-landmark, have the same color and are linked to each other.

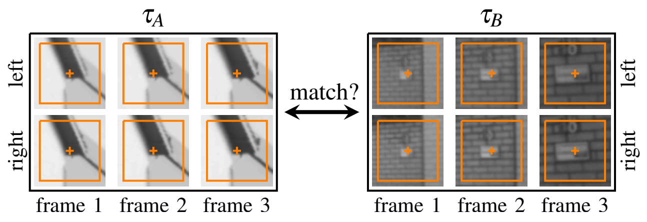


Fig. 2. In feature track matching it is already known which stereo feature pairs belong together in consecutive frames, i. e. which features form a feature track τ. To enable the recognition of landmarks over longer periods of time, we want to reliably identify matches between such feature tracks.

B. Visual Feature Tracks in Long-Term V-SLAM

The feature matching problem described above and depicted in Fig. 1 is the typical setup in optical flow, visual

[*]Control Methods and Robotics Lab, TU Darmstadt, Germany

odometry or short term V-SLAM methods, i. e. algorithms where the landmark observations lie usually just fractions of a second apart. Since only minor appearance changes occur in such matching problems, they are mostly solvable today. However, feature matching becomes more challenging in long-term V-SLAMs with landmark observations over several days, weeks and months [1], [2]. Long-term V-SLAMs like LLama-SLAM [9] are advantageous for automotive localization since they offer increased availability, scalability and updatability compared to visual short-term methods. In this paper, we will investigate the feature matching problem in such long-term V-SLAMs. Besides the more challenging matching task, long-term V-SLAMs also provide a different matching setup which is visualized in Fig. 2. Using a short-term matching system, it is already known which features belong together over several consecutive image frames, i. e. which features form a **feature track**. In the case of LLama-SLAM, the feature tracks are provided by an upstream Visual Odometry developed by Buczko et al. [10] which uses a Lucas-Kanade tracking [11] to match the features. Additionally, since a stereo-camera is used, there are actually two landmark observations available per image frame. Therefore, a feature track τ_k contains stereo feature pairs belonging to one landmark from several consecutive frames and can be written as $\tau_k = \{\theta_{k,1\mathrm{L}}, \theta_{k,1\mathrm{R}}, \theta_{k,2\mathrm{L}}, \theta_{k,2\mathrm{R}}, \ldots, \theta_{k,N\mathrm{L}}, \theta_{k,N\mathrm{R}}\}$.

C. Matching Feature Tracks

Contrary to the feature-to-feature matching problem between different frames in short term V-SLAM methods, we consider a track-to-track matching in long-term V-SLAMs: Matching feature tracks have to be identified, i. e. tracks that belong to the same real landmark. To our knowledge, there is up to now no published method which solves this problem. There is an approach published by Zhang et al. [12] that shares similar ideas, but it is limited to two feature observations and is not intended for feature track matching (more detailed comparison in Sec. II-D).

One challenge of long-term track-to-track matching is the possibly large variation in the feature's appearance due to different weather and lighting conditions which demands a more robust descriptor method. Furthermore, the existing descriptor comparison techniques cannot be used directly to compare feature tracks since they only allow the comparison of two features. Therefore, in order to compare feature tracks, the track information has to be consolidated in some stage of the feature comparison. Combining the actual patches of one track directly is obviously not a good idea, since it would produce a blurred patch that has lost most of the important characteristics. So the information consolidation has to be performed on the descriptor level or during the descriptor comparison. Consequently, we consider the following possible solutions for feature track comparison:

1) Compare all features of one track with all features from the other track and combine the resulting distances.
2) Choose one representative single feature from each track and use its descriptor as track descriptor.
3) Combine the descriptors of all features within the track to a new combined track descriptor.

In the first version of LLama-SLAM [9] we adopted the second approach[1] which lead to mediocre results. Thus, we developed new methods, based on three main approaches listed above. In this paper, we present all these methods and evaluate their performance using a real long-term data set.

D. Paper Overview

The remainder of this paper is organized as follows: We will first explain the main principles of binary descriptors in Sec. II, which are the basis of our track comparison methods. In Sec. III we give an in-depth description of the methods to compare feature tracks. Experimental results for the different methods are presented and discussed in the following Sec. IV. Finally, we will discuss strategies for comparing, matching and updating tracks in Sec. V and conclude the paper in Sec. VI.

II. Binary Feature Descriptors

The track-to-track comparison methods proposed in this paper use the binary descriptors BRIEF [14] and BOLD [15] as basis. We choose binary descriptors since they have a good ratio of accuracy to computational cost and their simple interpretable structure allows the design of descriptor combination schemes. This section introduces the basics of binary descriptors, starting with the "ancestor" BRIEF, followed by explanations of two major enhancements that lead to descriptors like BOLD.

A. The BRIEF Descriptor

The BRIEF descriptor invented by Calonder et al. in 2010 [14] was the first successful binary descriptor. It achieves comparable performance as sophisticated, histogram-based and real-valued descriptors like SIFT [16] or SURF [17] but follows a strikingly simple principle. A BRIEF descriptor $\mathbf{x}$ is a bit string where each bit x_v encodes the result of an intensity comparison in the surrounding area of the key point.

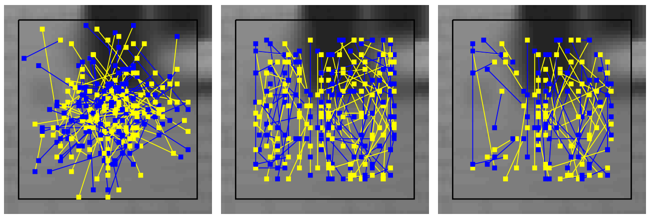
$$x_v = \begin{cases} 1 & \text{if } \tilde{\mathbf{I}}(\mathbf{p}+\mathbf{o}_{v,1}) < \tilde{\mathbf{I}}(\mathbf{p}+\mathbf{o}_{v,2}), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Therein, $\tilde{\mathbf{I}}$ is a smoothed version of the image and $\mathbf{o}_{v,1}$ and $\mathbf{o}_{v,2}$ specify the queried pixels relative to the key point $\mathbf{p}$. Figure 3(a) illustrates the principle of BRIEF displaying the test pattern of BRIEF overlaid on an example patch. The main design parameter of BRIEF is the choice of the tests, i. e. the queried pixel pairs $\{\mathbf{o}_{v,1}, \mathbf{o}_{v,2}\}$. Calonder et al. [14] found that the best choice are random tests, where $\mathbf{o}_{v,1}$ and $\mathbf{o}_{v,2}$ are independently drawn from a 2D-Gaussian distribution $N(0, \frac{1}{25}S^2)$, where S is the patch diameter. To compare two BRIEF descriptors the **Hamming distance**

$$\mathrm{d_h}(\mathbf{x}_i, \mathbf{x}_j) = \left|\mathbf{x}_i \oplus \mathbf{x}_j\right| \tag{2}$$

is used which is equal to the number of differing bits in the two bit strings. It can be easily computed using the

[1]More specifically, we used the BvB method described in Sec. III-C.2 with rBRIEF (ORB) from [13] as base descriptor.



(a) BRIEF (b) BOLD (c) masked BOLD

Fig. 3. Visualization of different descriptor test patterns. Only the first 128 tests are shown. For each test, the first pixel is marked with a blue square, the second pixel with an yellow square and both are connected by a line signaling the test result. If the second pixel is brighter than the first one, the test result is 1 (yellow line), otherwise it is 0 (blue line). The masked version of BOLD (c) contains a subset of the tests of BOLD (b). All tests which are unreliable for this specific patch are suppressed by the mask.

$\oplus$-operator (bitwise XOR) and the L1-Norm $|\cdot|$ which in this case is equivalent to counting the ones in the bit string.

B. *Offline Test Optimization*

As stated above the pixel pairs used for the tests in BRIEF are generated randomly. However, in later publications it was discovered that the choice of tests largely influences the matching performance and substantially better results can be achieved if the used tests are optimized [13], [15]. This idea was investigated for the first time by Rublee et al. and led to rBRIEF, an optimized BRIEF test pattern which is a main part of the popular ORB-feature [13]. They emphasized that in an optimized test pattern, each bit should have a variance close to the maximum value of 0.25 and the tests should be uncorrelated. Optimizing these two criteria ensures a high entropy of the descriptor and thereby improves its discriminability. Consequently, such an optimization increases the distance between dissimilar patches, i. e. **the inter-class distance**. Given a suitable training data set with dissimilar patches, a greedy search finds an optimized test pattern for this data set by investigating variance and correlation of all possible tests for a given patch size S. Due to the resulting performance benefit, this offline test optimization approach is part of many descriptors with binary intensity comparisons, like BOLD [15], SPHORB [18] and mdBRIEF [8]. Some more sophisticated descriptors, using other methods to extract the descriptor values, also apply such data-driven optimization, e. g. DAISY [19] or BinBoost [20].

To provide an intuition of the optimization results we display the test pattern for BOLD in Fig. 3(b). This optimized test pattern appears more structured and meaningful than the completely random pattern of BRIEF shown in Fig. 3(a). The test points are much wider distributed and they have a strong vertical tendency, i. e. the tests cover only small horizontal distances but large vertical distances.

C. *Online Test Selection*

Another enhancement of BRIEF is online test selection. This enhancement is used by the descriptor methods BOLD [15], mdBRIEF [8] and TailoredBRIEF [21]. While the offline test optimization described above increases the inter-class distance, a good matching performance is only reached if the descriptor also provides a low intra-class distance. In this context, a class contains all patches that match, i. e. that belong to the same real landmark. The intra-class distance can be reduced by using only the “reliable” tests of the test pattern. A test is denoted reliable if it always gives the same result when applied to matching patches belonging to the same class. The set of reliable tests may vary from class to class. Thus, for each class a different set of reliable tests is selected. Therefore, this approach is denoted online test selection, since the test selection can only be performed online during the actual descriptor computation when the specific patches of a class are given. The test selection can be done very efficiently by introducing a binary mask $\mathbf{w}_i$ for every class which has the same number of bits as the descriptor. If the test x_v is reliable, the bit $w_{i,v}$ of the mask is one, otherwise the bit is zero. This mask $\mathbf{w}_i$ is then used in the computation of the **masked descriptor distance**

$$\mathrm{d_m}(\mathbf{x}_i, \mathbf{w}_i, \mathbf{x}_j) = \left|(\mathbf{x}_i \oplus \mathbf{x}_j) \wedge \mathbf{w}_i\right|. \tag{3}$$

Therein, each zero-bit in the mask suppresses the contribution of an unreliable test to the hamming distance. So effectively the test pattern is reduced to a reliable test subset as displayed in Fig. 3(c). In most cases, each class contains just one single patch, since matches between patches are usually not known beforehand. Therefore, the class is artificially augmented by simulating viewpoint changes and generating synthetic variations of the patch. In TailoredBRIEF [21], for example, 25 synthetic variations of the patches are generated by simulating random scaling and 3D-rotation. BOLD [15] and mdBRIEF [8] generate only two synthetic patches with simulated in-plane rotations since they found this is sufficient to achieve good results. Using this augmentation method, a mask for each single patch can be created, i. e. each feature is given a descriptor $\mathbf{x}_i$ and a mask $\mathbf{w}_i$. To compare such features, the **symmetric masked descriptor distance**

$$\mathrm{d_{m2}}(\mathbf{x}_i, \mathbf{w}_i, \mathbf{x}_j, \mathbf{w}_j) = \frac{128}{|\mathbf{w}_i|}\left|(\mathbf{x}_i \oplus \mathbf{x}_j) \wedge \mathbf{w}_i\right| + \frac{128}{|\mathbf{w}_j|}\left|(\mathbf{x}_i \oplus \mathbf{x}_j) \wedge \mathbf{w}_j\right| \tag{4}$$

is used which is a more elaborated version of (3). This symmetric masked distance is computed by using the masks of both descriptors. To avoid the mask length biasing the distance measure, both terms are normalized by the length $|\mathbf{w}|$ of the mask. We furthermore add the factor 128 to yield a similar value range as for unmasked 256-bit descriptors.

To compare feature tracks we will also use online test selection. However, we do not have to generate synthetic variations of the feature like the aforementioned approaches but can use real appearance variations.

D. *Descriptor Methods using Multi-Frame-Appearances*

To our knowledge, there is so far only one other method published that compares features using the appearance of the feature in multiple frames: Zhang et al. [12] published a method in 2016 that aims to detect “loop-closures”, i. e. matching image frames that show roughly the same scene. In contrast, we focus on landmarks and want to identify matching feature tracks that belong to the same landmark. To identify matching frames they compare sets of codewords and these codewords are BOLD-like descriptors extracted from two consecutive frames. However, their approach is limited to two observations of the feature whereas we use an arbitrary number of (stereo-)observations in our approach. Furthermore, they compute the BOLD-descriptor on an artificial median patch, which is created by averaging the patches from the two frames. This averaging step may blur important structures in the patch and we therefore discarded such intensity averaging approaches completely, as we already argued in Sec. I-C. So although the method of Zhang et al. shares a similar idea, they follow a different objective and their method is less sophisticated and capable compared to the methods we propose below.

III. Comparing Feature Tracks

The binary descriptors described in Sec. II allow the comparison of single visual features. In this section we will now discuss methods to compare whole tracks of visual features with each other. We will use the function d_* to generally indicate a distance measure of a binary descriptor. So for BRIEF this function represents the hamming distance $\mathrm{d}_\mathrm{h}(\cdot,\cdot)$ from (2) whereas for BOLD it stands for the symmetric masked distance $\mathrm{d}_{\mathrm{m2}}(\cdot,\cdot,\cdot,\cdot)$ from (4).

A. Feature Track Comparison Matrix

We first want to introduce the feature track comparison matrix as basis for track comparison methods. Given a track τ_k of length N with descriptor set $\mathbf{X}_k$ and a second track τ_l of length M with descriptor set $\mathbf{X}_l$, the **feature track comparison matrix** for these two tracks is given by

$$\mathbf{C}_{kl} \in \mathbb{R}^{N\times M}, \quad \text{with} \quad c_{kl,ij} = (\mathbf{C}_{kl})_{ij} = \mathrm{d}_*(\mathbf{x}_{k,i}, \mathbf{x}_{l,j}). \quad (5)$$

So each element $c_{kl,ij}$ in this matrix is equal to the distance between the ith descriptor of track τ_k and the jth descriptor of track τ_l.

A track τ_k can also be compared to itself which leads to the **intra-track comparison matrix**

$$\mathbf{C}_{kk} \in \mathbb{R}^{N\times N}, \quad \text{with} \quad c_{kk,ij} = (\mathbf{C}_{kk})_{ij} = \mathrm{d}_*(\mathbf{x}_{k,i}, \mathbf{x}_{k,j}). \quad (6)$$

The main diagonal of this matrix contains only zeros, since comparing a descriptor with itself always gives zero. The matrix is also symmetric since the used descriptor distances are commutative, i. e. $c_{kk,ij} = c_{kk,ji}$.

B. Splitting Feature Tracks using Distance Levels

Before we present our approaches to compare tracks with each other, we want to describe an auxiliary measure we used to facilitate the track matching. Inside one track the observation distance, i. e. the distance from the camera to the feature, can vary in a wide range. For example, a feature could be observed from $z_\text{first} = 30\,\text{m}$ in the first frame of the track and observed from $z_\text{last} = 10\,\text{m}$ in the last frame.[2] Obviously, the appearance and the size of the feature in the image will change significantly with such distance variation. Although many descriptors are robust against minor viewpoint changes, they cannot handle the large appearance transformations resulting from such distance changes.

[2]The distance to the feature decreases from the first to the last frame, since the vehicle with the camera usually moves forward.

TABLE I. Used distance levels for the tracks in our data set. (patch size: 51 px × 51 px, focal length of cameras: 1212.5 px)

level L	z_L	distance range	real size
1	9.00 m	7.35 m to 11.02 m	0.38 m
2	13.50 m	11.02 m to 16.53 m	0.57 m
3	20.25 m	16.53 m to 24.80 m	0.85 m
4	30.38 m	24.80 m to 37.20 m	1.28 m
5	45.56 m	37.20 m to 55.80 m	1.92 m
6	68.34 m	55.80 m to 83.70 m	2.87 m

TABLE II. Comparison of key figures for the tracks in our data set with and without distance levels.

	without level	with levels
number of tracks	282	503
average length of tracks	5.61	3.15
average of mean($\mathbf{c}_{kk,i}$) for BOLD5	34.3	24.5
average of max($\mathbf{c}_{kk,i}$) for BOLD5	63.7	44.0
number of possible comparisons	39 621	28 483

To limit the size and appearance changes of the features in a track to a reasonable extent, we introduce distance levels, where each level corresponds to a certain distance range. The feature tracks are split into new tracks such that in the new tracks all features belong to one single distance level according to their observation distances. Since our track data set contains stereo features, we can easily determine the observation distance of a feature from its disparity. Table I shows the used distance levels for our data set. Each level L has a reference distance z_L and the factor between them is always 1.5, i. e. $z_{L+1} = 1.5 \cdot z_L$. The distance range covered by one level is given by $\left[z_L/\sqrt{1.5},\, z_L \cdot \sqrt{1.5}\right)$. Since the size of the features changes inversely proportional with the distance, the possible size change of a feature within one level lies in the interval $\left(1/\sqrt{1.5},\, \sqrt{1.5}\right]$ respectively $\left(0.82,\, 1.23\right]$. In most cases, binary descriptors are robust against such smaller size changes because the patch around the key point is smoothed before the intensity tests (cf. Sec. II-A and Sec. IV-B).

The levels can also be interpreted such that features have different real sizes for different levels respectively distance ranges (see last column of Tab. I). The real size states how large the side of a real-world square in distance z_L is that is covered by the descriptor patch with size 51 px × 51 px. For example, very close features on level 1 are only 38 cm × 38 cm in size whereas far distant features on level 6 have a size of 2.87 m × 2.87 m.

To illustrate the benefit of this level approach, Tab. II lists some key figures for our data set before and after splitting the tracks into different levels. The splitting approximately reduces the length of the tracks by half and consequently doubles the number of tracks. The most important difference is the reduced intra-track distance due to limited variability within a track. Tab. II lists the averaged mean and the maximum value of the columns $\mathbf{c}_{kk,i}$ of the intra-track comparison matrix (6), i. e. the mean and maximum distance of a descriptor to all other descriptors within the same track.

By splitting the tracks according to their distance levels, the intra-track distances are reduced by approximately 30%, both in mean and maximum value. The usage of distance levels also facilitates the comparison of feature tracks: We only allow comparison between tracks of the same level, i. e. only between landmarks with a similar real size. Therefore, the number of possible matches between the track decreases, although the total number of tracks increases (see Tab. II).

C. Track-to-Track Comparison Methods

In this section, we describe different distance measures which tell us whether two feature tracks are similar or not. We devise three different main approaches for a similarity measure $\mathrm{D}(\cdot,\cdot)$ to compare two feature tracks:

1) Comparing All Descriptors: An obvious possibility to compare two tracks is to compare all features of one track with all features of the other track, i. e. using the feature track comparison matrix introduced in Sec. III-A. Either the mean of all entries, $\mathrm{mean}\big(\mathbf{C}_{k,l}\big)$, or the largest entry within this matrix, $\max\big(\mathbf{C}_{k,l}\big)$, can serve as a similarity measure for the two tracks. Since all features of one track are compared to all features of the other track we call this approach **"All vs All" (AvA)** and define the two track distances:

$$\mathrm{D}_{\mathrm{meanAvA}}(\tau_k,\tau_l) = \mathrm{mean}(\mathbf{C}_{kl})\,, \tag{7}$$

$$\mathrm{D}_{\mathrm{maxAvA}}(\tau_k,\tau_l) = \max(\mathbf{C}_{kl}) \tag{8}$$

with comparison matrix $\mathbf{C}_{kl}$ given by (5). An apparent disadvantage of this approach is the large number of comparisons necessary to compare two tracks. For our data set, with a mean track length of 3.15, the mean number of necessary comparisons per track pair is approximately ten. Consequently, this approach causes ten times more computational cost then the two following approaches requiring only one descriptor comparison.

2) Comparing Representative Descriptors: To reduce the number of comparisons, we could choose one representative descriptor for each track and compute the distance only between these descriptors. One simple choice is the first left descriptor of each track, which leads to the track distance

$$\mathrm{D}_{\mathrm{FvF}}(\tau_k,\tau_l) = \mathrm{d}_*\big(\mathbf{x}_{k,1},\mathbf{x}_{l,1}\big)\,. \tag{9}$$

This naive approach is called **"First vs First" (FvF)** and it is used as a baseline for our experiments in Sec. IV-C, since it is very similar to comparing single features.

To get a more meaningful comparison method than this simple baseline, the descriptor should rather be chosen such that it represents the whole track. A reasonable choice for a representative is the descriptor $\mathbf{x}_{k,a^*}$ within the track that has minimal distance to all other features in the track. This descriptor can be seen as the "best" representative of the track and we use it in a method called **"Best vs Best" (BvB)**. The track distance for this method is defined as

$$\begin{aligned}\mathrm{D}_{\mathrm{BvB}}(\tau_k,\tau_l) &= \mathrm{d}_*\big(\mathbf{x}_{k,a^*},\mathbf{x}_{l,b^*}\big)\,,\ \text{with}\\ a^* &= \mathrm{argmin}_i\big(\mathrm{sum}(\mathbf{c}_{kk,i})\big)\\ \text{and}\quad b^* &= \mathrm{argmin}_i\big(\mathrm{sum}(\mathbf{c}_{ll,i})\big)\,.\end{aligned} \tag{10}$$

Therein, the best descriptor $\mathbf{x}_{k,a^*}$ with minimal distance to all other descriptors in the track is determined by finding the column $\mathbf{c}_{kk,a^*}$ in the intra-track comparison matrix $\mathbf{C}_{kk}$ with the smallest column sum. The same goes for the second track τ_l and finding its best descriptor $\mathbf{x}_{l,b^*}$ by evaluating $\mathbf{C}_{ll}$.

Instead of determining the best descriptor we can also establish the broad assumption that the "best" descriptor is always the median descriptor in the track. Since the median frame is captured from a roughly "median" camera-position, the median feature presumably has an appearance which is most similar to the features appearance in all other frames. This leads to a simplified version of the "Best vs Best" method, where we always pick the descriptor from the left median feature as representative. The resulting track distance is called **"Median vs Median" (MvM)** and is defined as

$$\begin{aligned}\mathrm{D}_{\mathrm{MvM}}(\tau_k,\tau_l) &= \mathrm{d}_*(\phi(\mathbf{X}_k),\phi(\mathbf{X}_l))\,,\ \text{with}\\ \phi(\mathbf{X}_k) &= \begin{cases}\mathbf{x}_{k,N/2} & \text{if } N \text{ even,}\\ \mathbf{x}_{k,(N+1)/2} & \text{if } N \text{ odd.}\end{cases}\end{aligned} \tag{11}$$

For tracks with an even number of features N, we decided to always choose the descriptor of the left first median feature. Of all comparison methods MvM is the least costly because it involves just one single descriptor comparison and determining the median feature is a very cheap computation.

3) Comparing Combined Descriptors: Another approach to compare tracks is combining the descriptor set $\mathbf{X}_k$ of a track to one combined descriptor $\bar{\mathbf{x}}_k$. This combined descriptor can then be conveniently compared to the combined descriptor $\bar{\mathbf{x}}_l$ of another track. For the considered binary descriptors the best choice to combine the descriptors is a bit-wise majority vote. So if a specific bit x_v has the value 1 in the majority of the descriptors of a track, it should also be 1 in the combined descriptor. For tracks with an even number of features there can be a tie. In this case, we take the bit-value from the median descriptor of the track, since the median descriptor should be on average a good representative of the whole track, as already stated above. A majority voting with this tiebreaker can be easily performed by using simple mean computation and thresholding which leads to the following formula for the **combined descriptor**

$$\bar{\mathbf{x}}_k = \sigma(\mathbf{X}_k) = [\bar{x}_{k,1}\ \bar{x}_{k,2}\ \ldots\ \bar{x}_{k,V}]^{\mathrm{T}}, \tag{12}$$

with the combined bit value

$$\begin{aligned}\bar{x}_{k,v} &= \begin{cases}1 & \text{if } \mu_{k,v} > 0.5,\\ 0 & \text{if } \mu_{k,v} < 0.5,\\ x_{k,N/2,v} & \text{if } \mu_{k,v} = 0.5 \text{ and } N \text{ even,}\\ x_{k,(N+1)/2,v} & \text{if } \mu_{k,v} = 0.5 \text{ and } N \text{ odd,}\end{cases}\\ \text{with}\quad \mu_{k,v} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} x_{k,i,v}.\end{aligned} \tag{13}$$

Besides the possibility to create a combined descriptor, we can also investigate the reliability of particular descriptor bits. Similar to the online test selection described in Sec. II-C, we can identify the most reliable bits for this particular landmark and create a binary mask containing only the reliable bits. However, we do not have to generate synthetic variations of the patch but can rather use the real appearance variations contained in the track. Reliable bits of a descriptor are bits where the value differs from the combined bit value only for a small fraction of the descriptors. Using the mean values $\mu_{k,v}$ from (13) the binary **reliability mask** for a track can be easily computed as

$$\boldsymbol{\omega}_k = [\,\omega_{k,1}\ \omega_{k,2}\ \dots\ \omega_{k,V}\,]^{\mathrm{T}},$$
$$\text{with}\quad \omega_{k,v} = \begin{cases} 1 & \text{if } \mu_{k,v} \le E \text{ or } \mu_{k,v} \ge 1-E, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where a 1 in the mask represents a reliable bit and E is a preset allowed variation threshold. This reliability criteria is equal to the requirement that the bit-variance is less than $E - E^2$. If the used base descriptor already uses a mask, e. g. BOLD, the combination function $\sigma(\cdot)$ from (12) is also applied to the descriptors' masks. The resulting combined mask is then joined with the reliability mask from (14) using a bit-wise AND, i. e.

$$\boldsymbol{\omega}_k^* = \sigma(\mathbf{W}_k) \wedge \boldsymbol{\omega}_k. \tag{15}$$

Based on this combined track descriptor we can define the **"Combined vs Combined" (CvC)** distance as

$$\mathrm{D}_{\mathrm{CvC}}(\tau_k, \tau_l) = \begin{cases} \mathrm{d_h}(\bar{\mathbf{x}}_k, \bar{\mathbf{x}}_l) & \text{for BRIEF,} \\ \mathrm{d_{m2}}(\bar{\mathbf{x}}_k, \sigma(\mathbf{W}_k), \bar{\mathbf{x}}_l, \sigma(\mathbf{W}_l)) & \text{for BOLD.} \end{cases} \tag{16}$$

If we add the reliability mask $\boldsymbol{\omega}$, we get the **"Combined and Masked" (CoMa)** distance

$$\mathrm{D}_{\mathrm{CoMa}}(\tau_k, \tau_l) = \begin{cases} \mathrm{d_{m2}}(\bar{\mathbf{x}}_k, \boldsymbol{\omega}_k, \bar{\mathbf{x}}_l, \boldsymbol{\omega}_l) & \text{for BRIEF,} \\ \mathrm{d_{m2}}(\bar{\mathbf{x}}_k, \boldsymbol{\omega}_k^*, \bar{\mathbf{x}}_l, \boldsymbol{\omega}_l^*) & \text{for BOLD.} \end{cases} \tag{17}$$

IV. EXPERIMENTS

In the previous section we define seven different methods to compare feature tracks: meanAvA, maxAvA, FvF, BvB, MvM, CvC and CoMa. To evaluate the performance of these methods, we performed an extensive experiment on a long-term data set with 282 feature tracks.

A. Feature Track Data Set

Since none of the existing feature matching data sets is suitable to extract feature tracks according to the notion in Sec. I-B, the evaluation is performed on a long-term data set recorded by our own research vehicle[3]. The feature tracks were extracted from three recordings from three different days (March 16, March 23 and May 5, 2017) on the same, 100 m long route section in a sub-urban area. To compose the feature tracks, the features are matched between consecutive frames by a Visual Odometry System [10] with Lucas-Kanade tracking [11]. On this route section the Visual Odometry System extracted approximately 100 feature tracks for each recording with 282 tracks in total. The extracted 282 original tracks are split according to their distance level as described in Sec. III-B, resulting in 503 tracks. To generate ground truth labels, first, candidate matches were identified with a simple, very liberal matching heuristic. In a second step these candidate matches were investigated and labeled in three iterations by human experts. The final data set contains 234 matching track pairs and 28 249 non-matching pairs. Please remember that we only allow comparisons between tracks with the same distance level, which significantly reduces the number of possible track pairs.

B. Used Descriptor Configurations

In this experiment, we use the BRIEF descriptor [14] and the BOLD descriptor [15], which were introduced in Sec. II, as base descriptors for the comparison methods. Both descriptors are used in their 256-bit version and they are computed on a patch with a diameter of $S = 51$ px.[4] Before the descriptor computation the image patches are smoothed with a 7×7 box filter. The prior smoothing reduces the noise sensitivity of the binary tests and thus increases stability and repeatability of the descriptors [14].

The BOLD descriptor is used in four variations: BOLD0 (no rotation, i. e. no bits are masked), BOLD5 ($\pm5°$), BOLD10 ($\pm10°$) and BOLD20 ($\pm20°$). Balntas et al. [15] reported that they achieve the best results when they generate two synthetic patches by an in-plane rotation of $\pm20°$ (BOLD20). However, we notice that on the used data set this configuration diminishes the performance because too many bits are masked. Therefore, we also use three additional BOLD variants with less rotation and therefore less masking.

The reliability masks for the CoMa method are computed with an allowed variation threshold of $E = 0.15$.

C. Experiment Results

To assess the performance of the comparison methods, we use ROC curves which is a common evaluation procedure for descriptors (cf. [7], [12], [15], [20], [21]). A ROC curve is constructed by varying the distance threshold to separate matches and non-matches and plotting the false positive rate (FPR) and the true positive rate (TPR) for each threshold. Hence, it nicely illustrates all possible trade-offs between detecting a high percentage of true matches (TPR) while maintaining a low percentage of false detections (FPR).

Figure 4 shows the ROC curves for all comparison methods with BOLD5 as base descriptor. Additionally, Tab. III lists the coordinates for some important FPR-TPR-points on the ROC curves. We first note that five of the proposed track comparison methods achieve a better performance than the baseline method FvF (black, dotted line). This confirms that considering the whole track instead of just one feature improves the matching performance. Only the maxAvA (orange) comparison method is worse than the baseline. Presumably, the maximum value of the track comparison matrix is too noisy for a reliable matching. We also see that MvM (green) has a distinctly weaker performance than the other four methods. Although BvB (brown) displays reasonable

[3]The utilized vehicle and cameras were provided by Continental as part of the PRORETA 4 research project. Our visual feature track data set is available at `https://www.proreta.tu-darmstadt.de/p4data`.

[4]We use the BRIEF pattern defined in OpenCV (`opencv_contrib/modules/xfeatures2d/src/generated_32.i`) in its original size of 49 px. For BOLD we use the pattern provided by the authors at `https://github.com/vbalnt/bold` and re-scaled it to 51 px.

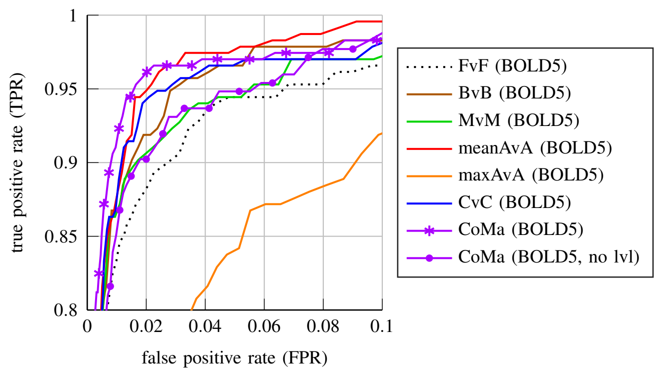


Fig. 4. ROC curves for the proposed comparison methods with BOLD5 as base descriptor. A method is superior to another method if it has a lower false positive rate and a higher true positive rate, i. e. its ROC lies left and above the other ROC. Please note that the plot only shows the upper left section of the ROC to make the differences visible.

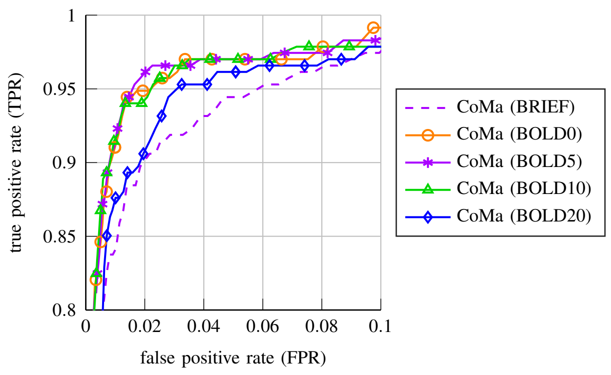


Fig. 5. ROC curves for the CoMa method using different base descriptors.

TABLE III. Comparison of significant FPR-TPR-points for the different track comparison methods with BOLD5 as base descriptor. FvF is used as baseline and the according performance improvement is given in parenthesis. The last row gives the results when the track splitting is not used (no lvl).

	method	FPR@TPR=95%		TPR@FPR=1%		TPR@FPR=0.1%	
with lvl	FvF	6.7%		83.5%		64.2%	
	BvB	2.9%	(−3.8%)	86.8%	(+3.3%)	65.4%	(+1.2%)
	MvM	5.5%	(−1.2%)	86.6%	(+3.2%)	61.7%	(−2.5%)
	meanAvA	2.0%	(−4.6%)	87.7%	(+4.2%)	58.9%	(−5.3%)
	CvC	2.7%	(−3.9%)	87.5%	(+4.0%)	65.3%	(+1.1%)
	CoMa	**1.6%**	(−5.1%)	**91.4%**	(+8.0%)	**68.8%**	(+4.6%)
	CoMa (no lvl)	5.6%	(−1.0%)	85.4%	(+1.9%)	64.4%	(+0.1%)

results, the track comparison methods that combine multiple appearances of the feature (meanAvA, CvC, CoMa) display the best performance. This shows that combining the short-term appearance changes within a track facilitates the long-term matching of feature tracks. The best results are achieved by CoMa (purple) and meanAvA (red), but they produce quite different ROC curve progressions: For low FPR values CoMa offers higher TPR values, whereas meanAvA achieves higher TPR values for FPR values above 3%. The superior performance of CoMa for low FPR values is also evident in the FPR-TPR-points listed in Tab. III.

In long-term V-SLAMs a low FPR value, i. e. a ROC at the far left, is much more important than high TPR values. A low FPR value ensures that almost all detected matches are true matches and that there are very few false positives that could corrupt the position estimation. This is more important than a high TPR value, since a V-SLAM does not necessarily need to find all matches – it only needs a sufficient number of matches to compute a reliable position estimate. Furthermore, there are much more non-matches than matches (cf. end of Sec. IV-A), which makes a low FPR even more important. For example in our data set, a 1% increase of the FPR is equal to 282.5 more false positive matches, whereas a drop of 1% in TPR equals only 2.3 lost matches. Because of this FPR/TPR-cost imbalance, the CoMa method is clearly favorable and the distance threshold to distinguish matches and non-matches should be chosen with strong preference on a very low FPR value. CoMa is also more efficient than meanAvA, since only one descriptor distance computation is needed for each track comparison, instead of the $N \times M$ computations necessary for meanAvA.

Figure 4 and Table III also show the performance change if the track splitting from Sec. III-B is not applied (no lvl). Without this prior splitting the performance of CoMa drops to a performance level similar to the baseline method FvF. This affirms that the splitting of the tracks considerably facilitates the matching, since it reduces the intra-track variations and provides additional matching constraints.

Figure 5 displays the performance of CoMa for different base descriptors. We note that the BOLD based methods (solid lines) provide an overall better performance than BRIEF (dashed line). Furthermore, we see that the BOLD20 variant recommended by [15] delivers worse performance than the other three less-masking BOLD variants. The best performance is achieved by BOLD5 with BOLD0 and BOLD10 following closely.

V. How to Match Feature Tracks

Based on the experiment results we can now specify how to match tracks of visual features: To facilitate the matching task, we recommend to split the feature tracks using distance levels as described in Sec. III-B. The best choice to compare feature tracks with each other is the CoMa method, since it offers the best performance with the highest TPR values for very low FPR values (see Tab. III). Furthermore, it is computationally inexpensive: The CoMa-descriptor of the track is computed just once and then only one descriptor distance computation is necessary for each track comparison. The distance between the CoMa-descriptors of the tracks serves as a basis for the track matching process. The threshold to classify a track pair as a match should be chosen very conservative to achieve a very low FPR, which is crucial for long-term V-SLAMs. To increase the matching accuracy further, additional matching strategies should be utilized, like matching only the nearest neighbor or considering the distance ratio to the second nearest neighbor (cf. [7]).

In a long-term V-SLAM like LLama-SLAM [9] the track-to-track matching can be extended to a track-to-landmark matching problem: In such a setup, the descriptor of the landmark is continuously updated with the features from all tracks that belong to this landmark. A CoMa-descriptor can be easily updated by repeating the computations from

(12) to (15) with appropriate weights. This update of the descriptor could lead to improved robustness, since more observed variations are captured in the reliability mask.

If more suitable for a specific SLAM implementation, the CoMa-descriptor can also be used to solve feature-to-landmark matching problems. In this case, the base descriptor computed on a single feature is directly compared to the CoMa-descriptor of the landmark using the asymmetric masked distance from (3) and the landmark's CoMa-descriptor is updated with each newly matched feature. However, we would rather recommend the track-to-track or track-to-landmark matching scheme above, since greater matching accuracy can be achieved if information from multiple feature observations is exploited in the comparison.

VI. Conclusion

We presented approaches to match tracks of visual features, i. e. sequences of consecutive feature observations. To facilitate the track matching problem, we first introduced a distance level scheme: The feature tracks are split into smaller tracks according to the observation distances of the contained features. This procedure reduces the intra-track variations and adds a useful constraint to the track matching. Only tracks of the same level, i. e. landmarks with a similar real size, are compared to each other. This reduces the number of track pairs that have to be investigated and reduces the chance of false matches.

In the main part of this publication we proposed seven feature track comparison methods that are built upon binary descriptors. Although we investigated only the track matching performance for BOLD and BRIEF, these comparison methods can also be used with other binary descriptor. With minor modifications, the comparison methods can even be used with most real-valued descriptor, if the elements of the descriptor carry independent information such that a mask can be applied. Furthermore, the proposed comparison methods have no restrictions on the camera setup, the image rate or the length of the feature tracks.

For the CoMa track comparison method the median appearance and the appearance variation of all features of a track are compressed into two binary strings. This compressed representation of the track displayed the best track-to-track matching performance in our experiments and has the same computational cost as usual feature-to-feature matching. In a long-term V-SLAM the CoMa comparison method can be used to match tracks to other tracks, respectively to match tracks to landmarks, by using a conservative distance threshold and possibly additional matching criteria. In addition, the CoMa method offers the ability to update a landmark's descriptor with new feature observations.

Since the used data set mainly contains well structured features like elements of buildings, roads and so on, it remains to proof the performance of the proposed methods on more versatile data sets. Furthermore, it is of great interest what results can be achieved if our track comparison methods are combined with other base descriptors or embedded in other long-term V-SLAMs beside LLama-SLAM.

Acknowledgment

We kindly thank Continental for their great cooperation within PRORETA 4, a joint research project of TU Darmstadt and Continental to investigate future concepts for intelligent and learning driver assistance systems.

References

[1] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Transport. Syst.*, vol. 2, no. 3, pp. 194–220, 2017.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part I: The first 30 years and fundamentals," *IEEE Robot. Automat. Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[4] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robot. Automat. Mag.*, vol. 19, no. 2, pp. 78–90, 2012.

[5] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, 2010.

[6] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Computer Vision - ECCV 2012*. Springer, 2012, pp. 759–773.

[7] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[8] S. Urban, M. Weinmann, and S. Hinz, "mdBRIEF - a fast online-adaptable, distorted binary descriptor for real-time applications using calibrated wide-angle or fisheye cameras," *Computer Vision and Image Understanding*, vol. 162, pp. 71–86, 2017.

[9] S. Luthardt, V. Willert, and J. Adamy, "LLama-SLAM: Learning high-quality visual landmarks for long-term mapping and localization," in *IEEE 21st Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 2645–2652.

[10] M. Buczko and V. Willert, "Flow-decoupled normalized reprojection error for visual odometry," in *IEEE 19th Int. Conf. on Intelligent Transportation Systems*, 2016, pp. 1161–1167.

[11] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," Intel Corporation. Microprocessor Research Labs, 2000.

[12] G. Zhang, M. J. Lilly, and P. A. Vela, "Learning binary features online from motion dynamics for incremental loop-closure detection and place recognition," in *IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 765–772.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE Int. Conf. on Computer Vision*, 2011, pp. 2564–2571.

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*. Springer, 2010, pp. 778–792.

[15] V. Balntas, L. Tang, and K. Mikolajczyk, "BOLD - binary online learned descriptor for efficient image matching," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, pp. 2367–2375.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[18] Q. Zhao, W. Feng, L. Wan, and J. Zhang, "SPHORB: A fast and robust binary feature on the sphere," *Int. J. Comput. Vision*, vol. 113, no. 2, pp. 143–159, 2015.

[19] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 43–57, 2011.

[20] T. Trzcinski, M. Christoudias, and V. Lepetit, "Learning image descriptors with boosting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 597–610, 2015.

[21] A. Richardson and E. Olson, "TailoredBRIEF: Online per-feature descriptor customization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 74–81.