HBST: A Hamming Distance embedding Binary Search Tree for Feature-based Visual Place Recognition

Dominik Schlegel and Giorgio Grisetti

***Abstract*— Reliable and efficient Visual Place Recognition is a major building block of modern SLAM systems. Leveraging on our prior work, in this paper we present a Hamming Distance embedding Binary Search Tree (HBST) approach for binary Descriptor Matching and Image Retrieval. HBST allows for descriptor Search and Insertion in logarithmic time by exploiting particular properties of binary descriptors. We support the idea behind our search structure with a thorough analysis on the exploited descriptor properties and their effects on completeness and complexity of search and insertion. To validate our claims we conducted comparative experiments for HBST and several state-of-the-art methods on a broad range of publicly available datasets. HBST is available as a compact open-source C++ header-only library.**

I. Introduction

Visual Place Recognition (VPR) is a well known problem in Robotics and Computer Vision [1] and represents a building block of several applications in Robotics. These range from Localization and Navigation to Simultaneous Localization and Mapping (SLAM). The task of a VPR system is to localize an image within a database of places represented by other images. VPR is commonly cast as a data association problem and used in loop closing modules of SLAM pipelines. A robust VPR system consists of one or multiple of the following components, which progressively improve the solution accuracy:

- Image Retrieval: is the process of retrieving one or more *images* from a database that are similar to a query one.
- Descriptor Matching: consists of seeking *points* between images which look similar. The local appearance of such points is captured by feature *descriptors*.
- Geometric Verification: is a common pruning technique that removes points obtained from Descriptor Matching, which are inconsistent with the *epipolar geometry*.

In the domain of Image Retrieval, common approaches compress entire images in single *global* descriptors to obtain high processing speed [2], [3]. Recently, convolutional neural network (CNN) methods demonstrated highly accurate results, especially in the field of long-term VPR [4], [5], [6]. These methods, however, might suffer from a high ratio of false positives when queried at high frequency, and thus often require a further stage of *local* Descriptor Matching to reject wrong candidates. Brute-force (BF) and k-d trees [7] are two prominent methods for solving this task.

Since the introduction of the BRIEF descriptor [8], the computer vision community embraced the use of binary descriptors due to their low computation and matching cost. Many popular feature-based SLAM systems such as ORB-SLAM [9] are built on these binary descriptors.

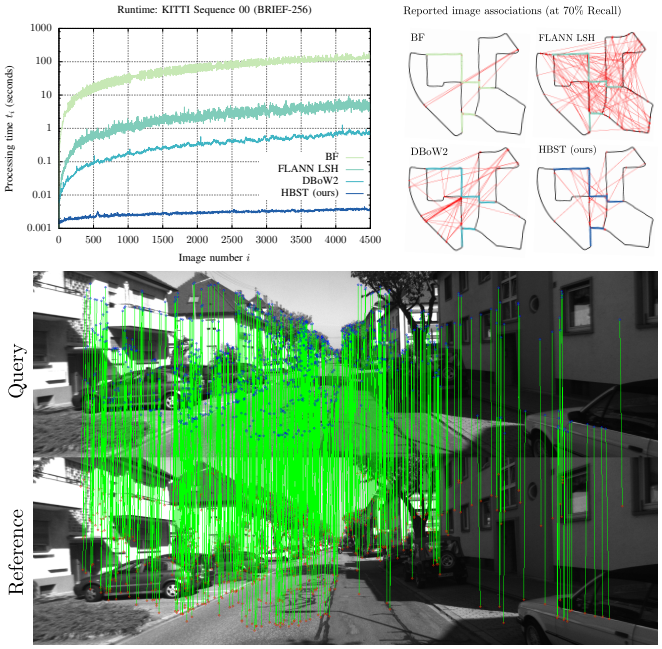


Fig. 1: Matching performance of the proposed HBST approach on KITTI Sequence 00. Top: Image processing times and image retrieval result of compared approaches at 70% Recall. Bottom: A single query and reference image with highlighted descriptor matches provided by HBST. The shown query image was acquired 4'500 frames after the reference image.

Whereas standard multi-dimensional search structures such as k-d trees are reported to perform well for incremental construction with floating point descriptors like SURF [10], the same approaches suffer a relevant performance loss when used with binary descriptors. This is the reason why in this work we focus on constructing a specific search structure, that is tailored to matching *binary* descriptors for VPR.

In this paper we propose an approach for binary descriptor matching *and* image retrieval that approximates the BF search. Our system does not need to construct any kind of dictionary and relies purely on a dynamically built binary search tree (BST) that allows for logarithmic searches and insertions of binary descriptors. Our approach runs several orders of magnitude faster than well-used implementations of other state-of-the-art methods, while retaining high matching accuracy. We provide our approach to the community in the form of a compact C++ header-only library[1] accompanied by several, straightforward use cases.

Both authors are with the Dept. of Computer, Control and Management Engineering, Sapienza University of Rome, Rome, Italy {lastname}@diag.uniroma1.it

[1]HBST is available at: www.gitlab.com/srrg-software/srrg_hbst
This article and related sources are part of our previous work [11]

II. Image Retrieval and Descriptor Matching

In this section we discuss in detail the two fundamental building blocks of VPR which we address in our approach: Image Retrieval and Descriptor Matching. We present related work directly in context of these two problems.

A. Image Retrieval

A system for *image retrieval* returns the image $\mathbb{I}_i^\star$ contained in a database $\{\mathbb{I}_i\}$ that is the most similar to a given query image $\mathbb{I}_q$ according to a similarity metric $e_\mathbb{I}$. The more similar two images $\mathbb{I}_i$ and $\mathbb{I}_q$, the lower the resulting distance becomes. More formally, image retrieval consists in solving the following problem:

$$\mathbb{I}_i^\star = \underset{\mathbb{I}_i}{\operatorname{argmin}}\, e_\mathbb{I}(\mathbb{I}_q, \mathbb{I}_i) : \mathbb{I}_i \in \{\mathbb{I}_i\}. \tag{1}$$

Often one is interested in retrieving *all* images in the database, whose distance to the query image $e_\mathbb{I}$ is within a certain threshold $\tau_\mathbb{I}$:

$$\{\mathbb{I}_i^\star\} = \{\mathbb{I}_i \in \{\mathbb{I}_i\} : e_\mathbb{I}(\mathbb{I}_q, \mathbb{I}_i) < \tau_\mathbb{I}\}. \tag{2}$$

The distance metric itself depends on the target application. A straightforward example of distance between two images is the Frobenius norm of the pixel-wise difference:

$$e_\mathbb{I}(\mathbb{I}_q, \mathbb{I}_i) = \|\mathbb{I}_q - \mathbb{I}_i\|_F. \tag{3}$$

This measure is not robust to viewpoint or illumination changes and its computational cost is proportional to the image size.

Global image descriptors address these issues by compressing an entire image into a set of few values. In the remainder we will refer to a global descriptor obtained from an image $\mathbb{I}$ as: $\mathbf{d}(\mathbb{I})$. GIST of Olvia and Torralba [2] and Histogram of Oriented Gradients (HOG) by Dalal and Triggs [3] are two prominent methods in this class. GIST computes a whole image descriptor as the distribution of different perceptual qualities and semantic classes detected in an image. Conversely, HOG computes the descriptor as the histogram of gradient orientations in portions of the image.

When using global descriptors, the distance between images is usually computed as the L_2 norm of the difference between the corresponding descriptors:

$$e_\mathbb{I}(\mathbb{I}_q, \mathbb{I}_i) = \|\mathbf{d}(\mathbb{I}_q) - \mathbf{d}(\mathbb{I}_i)\|_2. \tag{4}$$

Milford and Wyeth considered *image sequences* instead of single images for VPR. With SeqSLAM [12] they presented an impressive SLAM system, that computes and processes contrast enhancing image difference vectors between subsequent images. Using this technique, SeqSLAM manages to recognize places that underwent heavy changes in appearance (e.g. from summer to winter).

In recent years, CNN approaches have shown to be very effective in VPR. They are used to generate powerful descriptors that capture large portions of the scene at different resolutions. For one, there is the CNN feature boosted SeqSLAM system of Bai *et al.* [6], accompanied by other off-the-shelf systems such as ConvNet of Sünderhauf *et al.* [4] or NetVLAD by Arandjelović *et al.* [5]. The large *CNN descriptors* increase the description granularity and therefore they are more robust to viewpoint changes than global descriptors. CNN descriptors are additionally resistant to minor appearance changes, making them suitable for *lifelong* place recognition applications. One generally obtains a handful of CNN descriptors per image, which enable for high-dimensional image distance metrics for $e_\mathbb{I}$.

However, if one wants to determine the *relative location* at which images have been acquired, which is often the case for SLAM approaches, additional effort needs to be spent. Furthermore, due to their holistic nature, global descriptors might disregard the geometry of the scene and thus are more likely to provide false positives. Both of these issues can be handled by *descriptor matching* and a subsequent geometric verification.

B. Descriptor Matching

Given two images $\mathbb{I}_q$ and $\mathbb{I}_i$, we are interested in determining which pixel $\mathbf{p}_q \in \mathbb{I}_q$ and which pixel $\mathbf{p}_j \in \mathbb{I}_i$, if any, capture the same point in the world. Knowing a set of these point correspondences, allows us to determine the relative position of the two images up to a scale using projective geometry [13]. To this extent it is common to detect a set of salient points $\{\mathbf{p}\}$ (keypoints) in each image. Among others, the Harris corner detector and the FAST detector are prominent approaches for detecting keypoints. Keypoints are usually characterized by a strong local intensity variation.

The local appearance around a keypoint $\mathbf{p}$ is captured by a *local descriptor* $\mathbf{d}(\mathbf{p})$ which is usually represented as a vector of either floating point or boolean values. SURF [10] is a typical floating point descriptor, while BRIEF [8], BRISK [14] and ORB [15] are well known boolean descriptors. The desired properties for local descriptors are the same as for global descriptors: light and viewpoint invariance. Descriptors are designed such that regions that appear locally similar in the image result in similar descriptors, according to a certain metric $e_\mathbf{d}$. For floating point descriptors, $e_\mathbf{d}$ is usually chosen as the L_2-norm. In the case of binary descriptors, the Hamming distance is a common choice. The Hamming distance between two binary vectors is the number of bit changes needed to turn one vector into the other, and can be effectively computed by current processors.

Finding the point $\mathbf{p}_j^\star \in \mathbb{I}_i$ that is the most similar to a query $\mathbf{p}_q \in \mathbb{I}_q$ is resolved by seeking the descriptor $\mathbf{d}(\mathbf{p}_j^\star)$ with the minimum distance to the query $\mathbf{d}(\mathbf{p}_q)$:

$$\mathbf{p}_j^\star = \underset{\mathbf{p}_j}{\operatorname{argmin}}\, e_\mathbf{d}(\mathbf{d}(\mathbf{p}_q), \mathbf{d}(\mathbf{p}_j)) : \mathbf{p}_j \in \mathbb{I}_i. \tag{5}$$

If a point $\mathbf{p}_q \in \mathbb{I}_q$ is not visible in $\mathbb{I}_i$, Eq. (5) will still return a point $\mathbf{p}_j^\star \in \mathbb{I}_i$. Unfeasible matches however will have a high distance, and can be rejected whenever their distance $e_\mathbf{d}$ is greater than a certain *matching threshold* τ. The most straightforward way to compute Eq. (5) is the brute-force (BF) search. BF computes the distance $e_\mathbf{d}$ between $\mathbf{p}_q$ and *every* $\mathbf{p}_j \in \mathbb{I}_i$. And hence *always* returns the *closest* match for each query. This unbeatable accuracy

comes with a computational cost proportional to the number of descriptors $N_{\mathbf{d}} = |\{\mathbf{d}(\mathbf{p}_j)\}|$. Assuming $N_{\mathbf{d}}$ is the average number of descriptors extracted for each image, finding the best correspondence for each keypoint in the query image would require $\mathcal{O}(N_{\mathbf{d}}^2)$ operations. In current applications, $N_{\mathbf{d}}$ ranges from 100 to 10'000, hence using BF for descriptor matching quickly becomes computationally prohibitive.

To carry out the correspondence search in a more effective way it is common to organize the descriptors in a search structure, typically a tree. In the case of floating point descriptors, FLANN (Fast Approximate Nearest Neighbor Search Library) of Muja and Lowe [16] with k-d tree indexing is a common choice. The increased speed of FLANN compared to BF comes at a decreased accuracy of the matches. For binary descriptors, the (Multi-Probe) Locality-sensitive hashing (LSH) [17] by Lv *et al.* and hierarchical clustering trees (HCT) of Muja and Lowe [18] are popular methods to index the descriptors with FLANN. While LSH allows for *database incrementation* at a decent computational cost, HCT quickly violates real-time constraints. Accordingly, we consider only LSH in our result evaluations.

In our previous work [19] we presented a BST constructed according to a descriptor-based bit selection strategy. The BST exhibits a logarithmic search time for binary descriptor matching between images. Its shortcoming is that an individual tree has to be constructed for every image stored. In contrast, the approach presented in this paper utilizes a single, incrementally constructed BST for the entire database.

In [20] Feng *et al.* address binary descriptor matching for localizing in a known 3D model whose 3D points are labeled with binary descriptors. Similar to our approach, they utilize BSTs constructed according to a bit selection strategy. However, this technique exploits the knowledge about the provided 3D model. Alike, Komorowski *et al.* [21] propose a further bit selection strategy for BST-based binary descriptor matching with known descriptor to 3D point correspondences. Contrary to [20] and [21], our approach solely requires individual image data and does not rely on any kind of 3D point correspondence information.

C. Image Retrieval based on Descriptor Matching

Assuming to have an efficient method to perform descriptor matching as defined in Eq. (5), one could design a simple, yet effective image retrieval system by using a voting scheme. An image $\mathbb{I}_i$ will receive at most one vote $\langle \mathbf{p}_q, \mathbf{p}^\star_{i,j} \rangle$ for each keypoint $\mathbf{p}_q \in \mathbb{I}_q$ that is successfully matched with a keypoint of another image $\mathbf{p}^\star_{i,j} \in \mathbb{I}_i$. The distance between two images $\mathbb{I}_q$ and $\mathbb{I}_i$ is then the number of votes $\left|\left\{\langle \mathbf{p}_q, \mathbf{p}^\star_{i,j} \rangle\right\}\right|$ normalized by the number of descriptors in the query image $N_{\mathbf{d}}$:

$$e_{\mathbb{I}}(\mathbb{I}_q, \mathbb{I}_i) = \frac{\left|\left\{\langle \mathbf{p}_q, \mathbf{p}^\star_{i,j} \rangle\right\}\right|}{N_{\mathbf{d}}}. \tag{6}$$

The above procedure allows to gather reasonably good matches at a cost proportional to both, the number of descriptors in the query image $N_{\mathbf{d}}$ and the cost of retrieving the most likely descriptor as defined in Eq. (5).

An alternative strategy to enhance the efficiency of image retrieval, when local descriptors are available, is to compute *a single* global descriptor from *multiple* local descriptors. The well-known BOW (Bag-of-visual-Words) approaches follow this strategy by computing a global descriptor as the *histogram of the distribution of words* appearing in an image. A *word* represents a group of nearby descriptors, and is learned by a clustering algorithm such as k-means from a set of train descriptors. Similary, Jégou *et al.* [22] introduced with VLAD (Vector of Locally Aggregated Descriptors) a high-dimensional global descriptor, directly encoding a set of clustered local descriptors. BOW and VLAD approaches are reported to be both robust and efficient [9], [23], [24] however they do not provide point correspondences, that are required for geometric verification.

Notably, the open-source library DBoW2 by Galvez-Lopez and Tardos [25] extends the data structures used in BOW to add point correspondences to the system (*Direct Index*). DBoW2 is integrated within the recently published ORB-SLAM2 [9] by Mur-Artal *et al.* and displays fast and robust performance for ORB descriptors [15]. Another well-known BOW based approach is FAB-MAP [23] developed by Cummins *et al.* FAB-MAP allows to quickly retrieve similar images on very large datasets. FAB-MAP uses costly SURF descriptors [10] to maintain a certain level of individuality between the massive number of images described. Typically, BOW is used to determine a preliminary set of image candidates, on which BF, FLANN or BST descriptor matching is performed. This is a common practice for SLAM systems, that require high numbers of matches for few images.

In this paper, we present a novel approach that:

- Allows to perform image retrieval and descriptor matching *with* correspondences faster than BOW approaches perform image retrieval *without* correspondences.
- Yields levels of search correctness and completeness comparable to the ones achieved by state-of-the-art methods such as LSH [17] and DBoW2 [25].
- Allows for incremental insertion of subsequent descriptor sets (i.e. images) in a time bounded by the dimension of the descriptors $\dim(\mathbf{d})$.

Furthermore, we provide our approach as a compact C++ header-only library, that does not require a vocabulary or any other pretrained information. The library[1]is accompanied by a set of simple use cases and includes an OpenCV wrapper.

III. Our Approach

We arrange binary descriptors $\{\mathbf{d}_j\}$ extracted from each image $\mathbb{I}_i$ of an image sequence $\{\mathbb{I}_i\}$ in a binary tree. This tree allows us to efficiently perform descriptor matching. Additionally, we build a voting scheme on top of this method that enables fast and robust image retrieval.

A. Tree Construction

In our tree, each leaf $\mathcal{L}_i$ stores a subset $\{\mathbf{d}_{i,j}\}$ of the input descriptors $\{\mathbf{d}_j\}$. The leafs partition the input set such that each descriptor $\mathbf{d}_j$ belongs to a single leaf. Every non-leaf node $\mathcal{N}_i$ has exactly two children and stores an index

$k_i \in [0, .., \dim(\mathbf{d}) - 1]$. Where $\dim(\mathbf{d})$ is the descriptor dimension, corresponding to the number of bits contained in each descriptor. We require that in each path from the root to a leaf a specific index value k_i should appear at most once. This limits the depth of the tree h to the descriptor dimension. Fig. 2 illustrates an example tree constructed from 8 binary input descriptors $\{\mathbf{d}_j\}$ according to these rules.

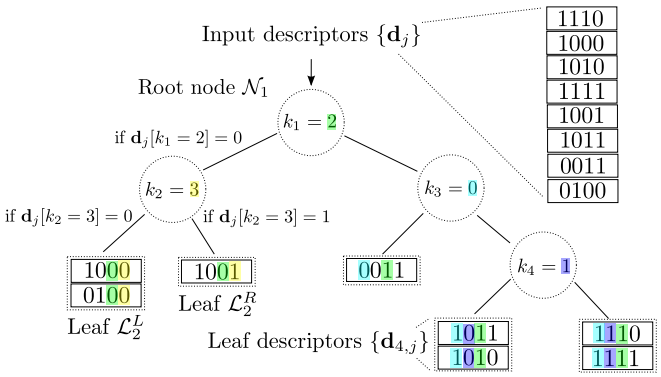


Fig. 2: HBST tree construction for a scenario with 8 input descriptors of dimension $\dim(\mathbf{d}) = 4$. The tree contains 4 nodes $\{\mathcal{N}_i\}$ (circles) with bit indices $\{k_i\}$, 5 leafs $\{\mathcal{L}_i\}$ (rectangles) and has maximum depth $h = 3$.

A descriptor $\mathbf{d}_j$ is stored in the left or in the right subtree depending on $\mathbf{d}_j[k_i]$, that is the bit value of $\mathbf{d}_j$ at index k_i. The structure of the tree is determined by the choice of the bit indices $\{k_i\}$ in the intermediate nodes and by the respective number of descriptors stored in the leafs $\{\mathcal{L}_i\}$.

B. Descriptor Search and Matching

The most similar descriptor $\mathbf{d}^\star_{i,j}$ to a query $\mathbf{d}_q$ is stored in a leaf $\mathcal{L}^\star_i$. This leaf is reached by traversing the tree, starting from the root. At each intermediate node $\mathcal{N}_i$ the search branches according to $\mathbf{d}_q[k_i]$. Eventually, the search will end up in a leaf $\mathcal{L}^\star_i$. At this point all leaf descriptors $\{\mathbf{d}_{i,j}\}$ stored in $\mathcal{L}^\star_i$ are sequentially scanned (BF matching) to seek for the best match according to Eq. (5). Fig. 3 illustrates two examples of the proposed search strategy.

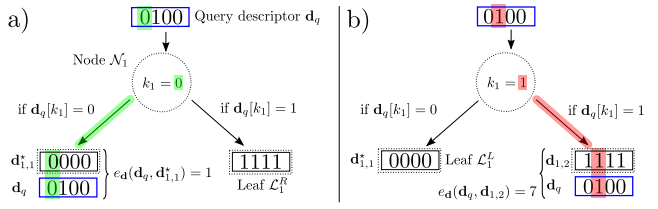


Fig. 3: Search scenarios a) and b) for a small tree of depth $h = 1$. The only configuration change between the scenarios is the value of k_1. In this example only a single descriptor is contained in each leaf. For $\mathbf{d}_q$ the best matching descriptor is $\mathbf{d}^\star_{1,1}$, which is found in a) but not in b).

Organizing N_j descriptors in a balanced tree (Sec. III-D) of depth h, results in having an average of $\frac{N_j}{2^h}$ descriptors in a leaf. Consequently, the time complexity of a search is:

$$\mathcal{O}(h + \frac{N_j}{2^h}) \tag{7}$$

since h operations are needed to find the leaf and the descriptor matching in the leaf can be performed in $\frac{N_j}{2^h}$ steps.

If a query descriptor $\mathbf{d}_q$ is already contained in the tree, the search is guaranteed to correctly return the stored descriptor $\mathbf{d}^\star_{i,j}$. This, however does not hold for nearest neighbor searches when one is interested in finding the descriptor in the tree that is similar to $\mathbf{d}_q$. This is a consequence of the binary search procedure that preforms a greedy search based on the bit index k_i at each node. Once a leaf is reached, only descriptors in that leaf are considered as potential results. Thus we can say that in general the nearest neighbor search in the tree is not ensured to be *correct*. In practice, however, one is usually interested in finding a descriptor $\mathbf{d}_{i,j}$ that is *similar enough* to $\mathbf{d}_q$. Hence incorrect matches are tolerated as long as they are not too far off w.r.t. $\mathbf{d}_q$, according to the metric $\mathbf{d}_{i,j} : e_\mathbf{d}(\mathbf{d}_q, \mathbf{d}_{i,j}) < \tau$.

If we want to retrieve *all* descriptors that lay within a certain distance $\{\mathbf{d}^\star_{i,j} : e_\mathbf{d}(\mathbf{d}_q, \mathbf{d}_{i,j}) < \tau\}$, the search in the tree might be not *complete*. Incompleteness occurs when only a subset of the feasible matches are returned from a search. If a search is complete, it is also correct.

C. Completeness Analysis

A bounded nearest neighbor search for a query descriptor $\mathbf{d}_q$ and a threshold τ is said to be *complete* if all possible matching descriptors $\{\mathbf{d}_j^{(\tau,q)}\}$ such that $e_\mathbf{d}(\mathbf{d}_q, \mathbf{d}_j^{(\tau,q)}) < \tau$ are returned. Given $\mathbf{d}_q$, our search procedure returns all descriptors $\{\mathbf{d}_{i,j}^{(\tau,q)}\}$ in the leaf $\mathcal{L}_i$ whose distance is below τ. These matching descriptors are necessarily a subset of all feasible ones $\{\mathbf{d}_{i,j}^{(\tau,q)}\} \subset \{\mathbf{d}_j^{(\tau,q)}\}$. A straightforward measure of *completeness* for a *single* descriptor search is:

$$c_\tau(\mathbf{d}_q) = \frac{|\{\mathbf{d}_{i,j}^{(\tau,q)}\}|}{|\{\mathbf{d}_j^{(\tau,q)}\}|} \in [0, 1]. \tag{8}$$

Given a set of input descriptors $\{\mathbf{d}_j\}$, a set of query descriptors $\{\mathbf{d}_q\}$, a search threshold τ and a search tree constructed from $\{\mathbf{d}_j\}$, we can evaluate the *mean completeness* $\bar{c}_\tau(\{\mathbf{d}_q\})$ over *all* searches. This gives us a meaningful measure of the overall completeness of our search. Since the structure of the search tree is governed by the choice of bit indices $\{k_i\}$, we conducted an experiment to evaluate how the choice of k_i influences the completeness, under different matching thresholds τ. For that reason we evaluated the resulting *bitwise* mean completeness $\bar{c}_\tau(k_1)$ at depth $h = 1$ by considering two descriptor sets $\{\mathbf{d}_q\}$ and $\{\mathbf{d}_j\}$ for every possible bit index k_1 in the root node.

In Fig. 4 we report the results of our bitwise completeness analysis on descriptors extracted from images of Sequence 06 of the KITTI benchmark dataset [26]. A broader analysis, featuring also FREAK and A-KAZE descriptors as well as many other datasets, is available on the project website[1].

Based on the results shown in Fig. 4 we infer that:

- The choice of the bit index k_1 has a limited impact on the mean completeness $\bar{c}_\tau(k_1)$ at $h = 1$. This behavior is similar for different types of binary descriptors.
- The higher the matching distance threshold τ, the lower the mean completeness $\bar{c}_\tau(k_1)$ becomes.

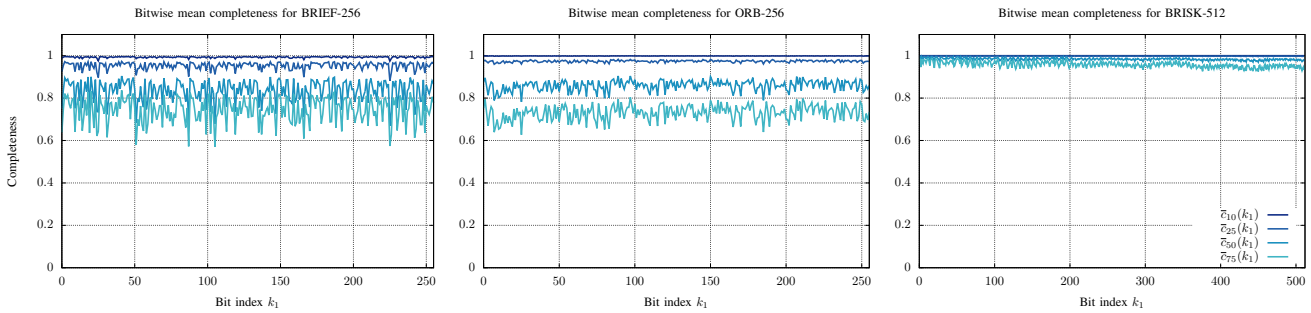


Fig. 4: Bitwise mean completeness $\bar{c}_\tau(k_1)$ for matching thresholds $\tau \in \{10, 25, 50, 75\}$ at depth $h = 1$, for different descriptor types. A number of $N_\mathbf{d}$ =1'000 descriptors has been extracted per image. The ground truth for this experiment consisted of 5'153 image pairs, corresponding to over 1.5 million descriptors. The colorization and legend based on τ is identical for all plots and can be inspected in the rightmost figure. Note that the BRISK-512 descriptor has $\dim(\mathbf{d}) = 512$ and therefore the considered matching threshold τ is much more restrictive with respect to BRIEF-256 and ORB-256.

The above experiment (Fig. 4) only considers trees of depth $h = 1$. It's results, however, can be used to predict the evolution of the completeness at higher depths. Let $\bar{c}_\tau(h)$ be the *overall* mean completeness of a tree of depth h with maximum matching distance τ. Performing a search on the tree would result in applying the decision rule $\mathbf{d}[k_i]$ exactly h times, and each decision would result in a potential loss of completeness according to Eq. (8). Assuming that $C_\tau = \bar{c}_\tau(h = 1) \in [0, 1]$ is evaluated on a representative set of query and input descriptors, we expect that the relative completeness loss does not change significantly on other tree levels as well. Thus we predict $\bar{c}_\tau(h)$ to decrease exponentially with the depth h as follows:

$$\bar{c}_\tau(h) \simeq \hat{c}_\tau(h) = C_\tau^h \in [0, 1]\,. \tag{9}$$

To support our conjuncture, we extend the evaluation conducted in the previous experiment (Fig. 4), by measuring $\bar{c}_\tau(h)$ for increasing depths $h = \{0, 1, .., 9\}$. The number of possible full trees at a certain depth h is $\frac{\dim(\mathbf{d})!}{(\dim(\mathbf{d})-2^{h+1}+1)!}$. Hence, to be able to cover higher depths in reasonable time, we conducted a Monte Carlo simulation by considering 100 full trees per depth. A tree of depth h is constructed by setting the bit index k_i of each tree node $\mathcal{N}_i$ to a value randomly sampled without repetitions from the indices of the descriptor. Fig. 5 reports the results of this evaluation.

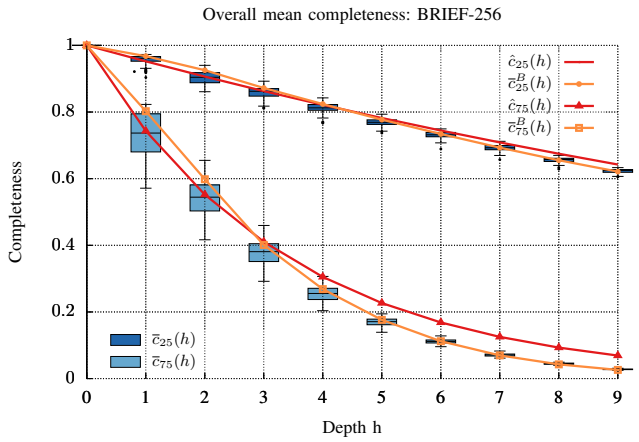


Fig. 5: Predicted mean completeness $\hat{c}_\tau(h)$ of Eq. (9) (red) and measured mean completeness $\bar{c}_\tau(h)$ of 100 random trees per depth (blue boxplots). Additionally, we display the measured mean completeness $\bar{c}^B_\tau(h)$ when choosing $\{k_i\}$ according to Eq. (10), favoring a balanced tree (Sec. III-D).

From the analysis of Fig. 4 and Fig. 5 we conclude that:

- The bit index k_i does not significantly affect the mean completeness $\bar{c}_\tau(k_i)$ at a given depth h.
- The overall mean completeness $\bar{c}_\tau(h)$ decreases exponentially with increasing depth h of the search tree.
- Eq. (9) captures the relation between depth of the tree and achievable completeness $\bar{c}_\tau(h)$ reasonably well, especially for low depths h.

The reader might notice that Eq. (9) is slightly optimistic for higher depths. This occurs because enforcing the construction of a full tree at high depth h with only a limited number of descriptors $N_\mathbf{d}$ results in very small leafs, containing few descriptors. Thus the approximate search approaches an exact one and the threshold τ becomes irrelevant.

D. Balanced Tree Construction

In this section we describe how to organize a set of descriptors in a *balanced* tree of depth h. Considering Eq. (7) and Eq. (9), for a given threshold τ, we have a trade-off between search time and completeness. Higher values of h will result in increased search speed at the cost of a reduced completeness. These results however, hold only in the case of balanced trees, and both search speed and completeness will decrease when the tree becomes unbalanced.

A straightforward strategy to build a balanced tree from a set of input descriptors $\{\mathbf{d}_j\}$ consists in recursively splitting the current input descriptors evenly. Since the structure of the tree is governed by the choice of $\{k_i\}$, to achieve an even partitioning of $\{\mathbf{d}_j\}$, we choose the bit index for which the bit $\mathbf{d}_j[k_i]$ is set for half of $\{\mathbf{d}_j\}$ for *every* node $\mathcal{N}_i$. The chosen bit index $k_i^\star$ will therefore be the one whose *mean value* among all descriptors is the *closest* to 0.5:

$$k_i^\star = \underset{k_i}{\operatorname{argmin}} \left| 0.5 - \frac{1}{N_j} \sum_j \mathbf{d}_j[k_i] \right|. \tag{10}$$

Note that when selecting k_i we have to neglect all the indices that have been used in the nodes ancestors. Fig. 5 shows the resulting completeness $\bar{c}^B_\tau(h)$ obtained by using Eq. (10).

If the minimized norm in Eq. (10) is below a certain threshold δ_{max}, we say that the mean value is *close enough* to 0.5 and pick $k_i^\star$ for splitting. In case that no such mean value is available, we do not split the descriptors and the recursion stops.

Constructing a tree of depth h for N_j descriptors according to Eq. (10) has a complexity of $\mathcal{O}(N_j \cdot h)$. In typical applications such as SLAM, N_j grows significantly for every new image, as new descriptors are added to the set. Therefore constructing the tree from scratch for all descriptors of all images for every new image quickly leads to runtimes not adequate for real-time applications. To overcome this computational limitation we propose an alternative strategy to *insert* new images (i.e. descriptors) into an *existing* tree.

E. Incremental Tree Construction

In this section we describe an alternative strategy that allows to augment an initial tree with additional descriptors while limiting its depth. The idea is to accumulate descriptors in a leaf until a number N_{max} (maximum leaf size) is reached. Whereas hierarchical clustering trees [18] use the maximum leaf size as termination criterion for the clustering process, we on the other hand evaluate it to determine if a clustering (i.e. splitting) is necessary. When the maximum leaf size is exceeded we turn the leaf $\mathcal{L}_i$ in an intermediate node $\mathcal{N}_i$. The bit index k_i for $\mathcal{N}_i$ is selected according to the criterion in Eq. (10), and the descriptors previously contained in $\mathcal{L}_i$ are organized in two new leafs spawning from $\mathcal{N}_i$. Fig. 6 illustrates the proposed procedure.

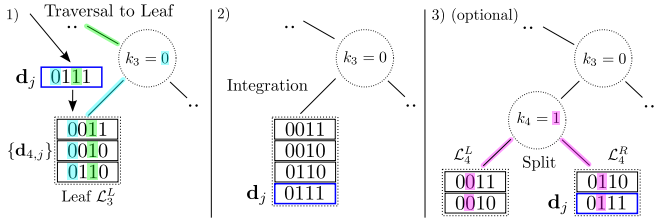


Fig. 6: Descriptor insertion procedure for a single input descriptor $\mathbf{d}_j$. Only the affected part of the tree is shown. Step 1) The leaf $\mathcal{L}_i$ containing the most similar descriptor(s) to $\mathbf{d}_j$ is found. Step 2) $\mathbf{d}_j$ is integrated into the leaf descriptor set $\{\mathbf{d}_{4,j}\}$. Step 3) The leaf ($N_i = 4$) breaks into two child leafs and becomes an intermediate node (in this example we set $N_{max} = 3$).

Notably the tree traversal needed to find $\mathcal{L}_i$ is the same as for the search. This enables us to perform both search and insertion at the same time. Albeit this straightforward insertion technique does not guarantee a balanced tree, it succeeds in limiting the depth of the tree as shown in Fig. 7.

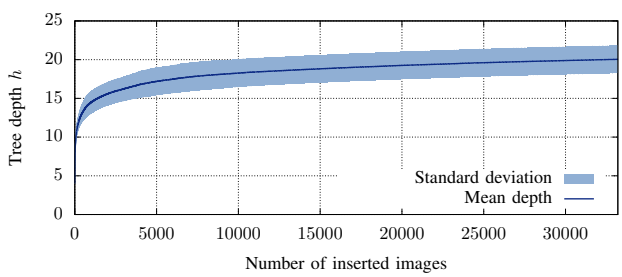


Fig. 7: Mean and standard deviation of the tree depth h for increasing numbers of sequentially inserted images. $N_{\mathbf{d}} = 1'000$ BRIEF-256 descriptors were extracted for each of the 33'197 images in the dataset. For this experiment we set $\tau = 25, \delta_{max} = 0.1$ and $N_{max} = 100$.

Note that using a re-balancing approach such as for Red-Black trees is not straightforward in our case, since the constraint that a bit index would appear at most once in a path from root to leaf would be violated by moving nodes. In our approach, no tree re-balancing is performed as we are able to enforce a desired balance to a satisfiable degree using the parameter δ_{max} (Sec. III-D).

To enable image retrieval, we augment each stored descriptor with the index of the image from which it was extracted. This allows us to implement a voting scheme for image retrieval at a minimal additional cost.

IV. Experiments

In this section we report the results of a comparative evaluation of our approach with several state-of-the-art methods (Sec. IV-A). We measure the image retrieval accuracy and the runtime of each method on multiple publicly available datasets (Sec. IV-B). To quantify the accuracy of image retrieval, we extract a VPR ground truth on which images should match for the analyzed datasets, using a brute-force offline procedure (Sec. IV-C).

For each dataset and each approach we process the images sequentially. Every time a new image is acquired, the approaches are queried for image retrieval. Based on the reported image matches and on the provided ground truth we then can compute Precision-Recall curves and F_1 score [1]. The database is subsequently augmented by inserting the new image, so that it can be returned as a match in future queries. We gather runtime information by measuring the time t spent for both of these operations for each image.

A. Compared Approaches

Our comparison has been conducted on the following state-of-the-art image retrieval approaches:

BF: The classical brute-force approach. We utilize the current OpenCV3 implementation. BF is expected to achieve the highest precision and recall (and F1 score), while requiring the highest processing time t per image.

LSH: We utilize the current OpenCV3 implementation of FLANN with Multi-Probe LSH indexing. The LSH index is built using the parameters: $\text{table\_number} = 10$, $\text{key\_size} = 20$ and $\text{multi\_probe\_level} = 2$.

DBoW2-DI/SO: We used the official DBoW2 library of [25] (revision: 82401ca), that is also implemented in ORB-SLAM2 [9]. We enabled Direct Indexing (DI), so that image matches are pruned by decreasing number of matches obtained through descriptor matching using the provided DBoW2 indices. DBoW2-DI was run with parameters: $\text{use\_di} = \text{true}$ and $\text{di\_levels} = 2$. Additionally we evaluated the image Score Only (SO) mode by setting $\text{use\_di} = \text{false}$. Note that in this configuration, DBoW2 does not report matching descriptors but only matching images.

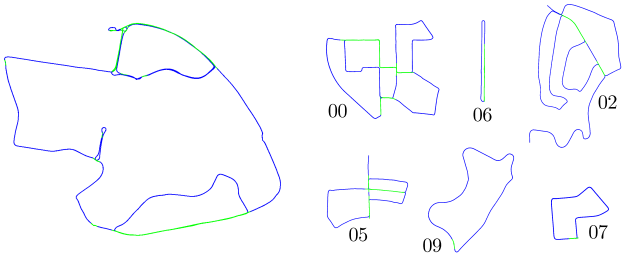
HBST-10/50: The approach proposed in this paper, with parameters: $\delta_{max} = 0.1$, $N_{max} = 10$ (HBST-10) and $N_{max} = 50$ (HBST-50). HBST-50 is designed to hold larger leafs and hence aims to provide a higher accuracy than HBST-10 at the price of a higher processing time t.

For all approaches we considered a maximum descriptor matching distance of $\tau = 25$ and we extracted for each image

$N_\mathbf{d}$ = 1'000 BRIEF-256 descriptors. All results were obtained on the same machine, running Ubuntu 16.04.3 with an Intel i7-7700K CPU@4.2GHz and 32GB of RAM@4.1GHz. A more extensive evaluation featuring various binary descriptor types (e.g. ORB, BRISK, FREAK and A-KAZE) is available on the project website[1].

B. Datasets

We performed our result evaluation on 3 publicly available large-scale visual SLAM datasets: St. Lucia [27], KITTI [26] and Málaga [28]. Each dataset contains ground truth camera trajectories of multiple kilometers length described by thousands of images. For space reasons, we report in this paper only the results of KITTI and St. Lucia. The results being in line with the results on Málaga, which can be inspected on the project website[1]. In Fig. 8 one can inspect the evaluated camera trajectories of St. Lucia and KITTI.



(a) St. Lucia: 9.5 km, 33'197 images. (b) KITTI: 14.6 km, 15'756 images.

Fig. 8: Selected datasets sequences with provided ground truth camera trajectories in blue. Matching image segments (defined by the VPR ground truth of Sec. IV-C) are highlighted in green. Best viewed in color.

C. Ground Truth Computation

Obtaining the ground truth for image retrieval is a crucial aspect of our evaluation. To this extent we employ a brute-force approach aided by the ground truth camera pose information available in the datasets. We report a *match* (true positive) between a query $\mathbb{I}_q$ and an image $\mathbb{I}_i$ in the database, whenever *all* of the following criteria are met:

1) The fields of view at which the images were acquired must overlap, and the camera positions have to be close. This occurs when two images are acquired at positions closer than 10 meters, and the optical axes of the cameras have an angular distance below 20 degrees.
2) Since all approaches are designed to approximate the BF accuracy, we require that matching images are supported by a minimum number of matching descriptors. This test is passed when more than 10% of the descriptors are within the matching threshold $\tau = 25$.
3) To confirm the usability of returned descriptor matches for image registration, we perform a geometric verification for the keypoint correspondences $\langle \mathbf{p}_q, \mathbf{p}_j \rangle$. A correspondence $\langle \mathbf{p}_q, \mathbf{p}_j \rangle$ is valid, if the essential constraint $\mathbf{p}_q^\top \mathbf{E} \mathbf{p}_j = 0$ is approached [13].

The tool we used to generate such a ground truth for image matches is available online[2]. The subset of matches that passes our criteria forms the set of *ground truth matches*.

[2]Benchmark project: `www.gitlab.com/srrg-software/srrg_bench`

D. Results

We evaluated Runtime, Precision-Recall curves and the maximum F1 score of each compared approach on each dataset sequence using our benchmark utility[2]. In Fig. 9 we present a detailed performance analysis on the St. Lucia sequence with 33'197 images (Fig. 8a) extracting BRIEF-256 descriptors. We processed every 5th image in the sequence, resulting in a subset of 6'575 images.

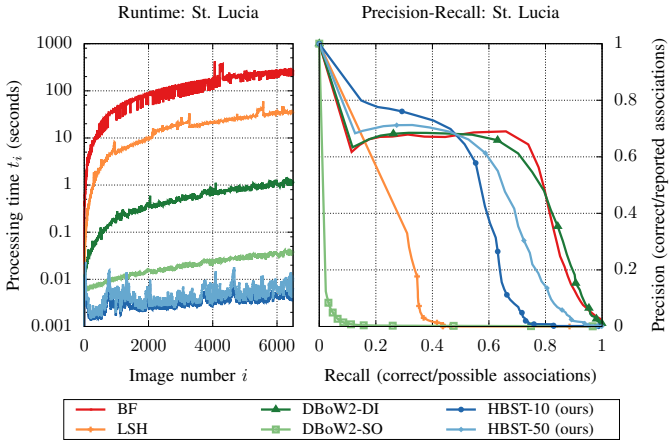


Fig. 9: UQ St. Lucia Stereo Vehicular Dataset: Wide-ranging urban environment. We processed 6'575 images, obtained at a subsampling rate of 5. 1'000 BRIEF-256 descriptors were extracted for each processed image.

We examined the Runtime and Precision-Recall curves of all approaches in Fig. 9 and conclude:

BF: Brute-force is the most accurate approach, at the cost of a higher computation time that grows linearly with the number of inserted images. Real-time performance is prohibited after inserting a handful of images.

LSH: LSH manages to sustain a mediocre accuracy. However, the hashing does not allow for sufficient discretization in a dataset of this size, resulting in a relatively high false positive rate at low recall.

DBoW2-DI: The evaluation of the reported descriptor matches based on the direct index of DBoW pays off. DBoW2-DI lies head-to-head with BF, achieving an excellent precision at high recall. Albeit requiring 2 magnitudes more processing time than HBST.

DBoW2-SO: The precision of DBoW2-SO drastically decreases already at low recall. The visual word based histogram abstraction shows to be too coarse for a dataset of this size. Its processing time is significantly less than DBoW2-DIs, making it the fastest approach after HBST.

HBST-10/50: Our method outperforms all other approaches in processing speed while providing a competitive accuracy. As expected, HBST-50 maintains a slightly higher precision than HBST-10, at an increased computational cost.

In Fig. 10 we report the results of all approaches on the KITTI benchmark dataset (Fig. 8b). Instead of Precision-Recall and processing time t, here we report the maximum F1 scores and the mean processing time $\bar{t}$. This to keep the result details as concise as possible for the considered sequences. Full plots are available on the project website[1].

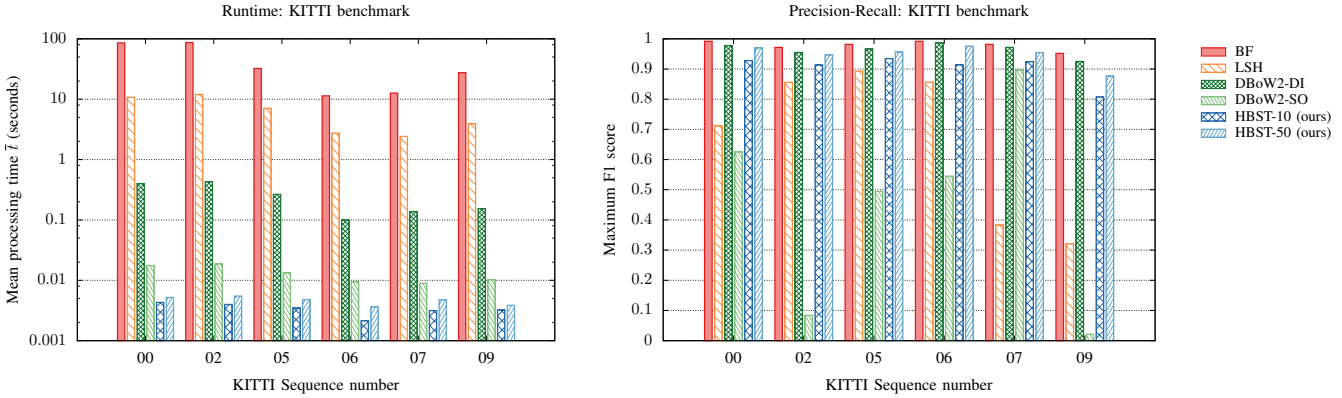


Fig. 10: KITTI Visual Odometry/SLAM Evaluation 2012: Large-scale urban and rural environments. We extracted 1'000 BRIEF-256 descriptors per image.

For each compared approach in Fig. 10 we observe:
BF: BF is the most accurate but also the slowest approach.
LSH: The LSH-indexed FLANN search achieves decent F_1 scores between DBoW2-SO and HBST-10. Clearly, LSH is not applicable for real-time processing in our scenario.
DBoW2-DI: The BOW approach achieves the best F_1 score after BF, yet it is two orders of magnitude slower than HBST.
DBoW2-SO: The histogram comparison leads to the poorest F_1 score. However, it is the fastest approach after HBST.
HBST-10/50: Our approach achieves accuracy between LSH and DBOW2-DI, while being the fastest approach compared.

V. Conclusions

In this paper we present a binary feature-based search tree approach for Visual Place Recognition. We conducted an analysis of the behavior of binary descriptors. Based on this analysis we provide an approach that can address descriptor matching and image retrieval. While retaining an adequate accuracy, our approach significantly outperforms state-of-the-art methods in terms of computational speed. All of our results were obtained on publicly available datasets and can be reproduced using the released open-source library.

References

[1] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Trans. on Robotics*, 2016.

[2] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[4] N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," in *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[5] R. Arandjelovi, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 40, no. 6, pp. 1437–1451, 2018.

[6] D. Bai, C. Wang, B. Zhang, X. Yi, and X. Yang, "CNN Feature boosted SeqSLAM for Real-Time Loop Closure Detection," *CoRR*, vol. abs/1704.05016, 2017.

[7] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, 1975.

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. of the Eur. Conf. on Computer Vision (ECCV)*, 2010.

[9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Underst.*, 2008.

[11] D. Schlegel and G. Grisetti, "HBST: A Hamming Distance embedding Binary Search Tree for Visual Place Recognition," *CoRR*, vol. abs/1802.09261, 2018.

[12] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[14] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2011.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2011.

[16] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," *VISAPP*, 2009.

[17] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe LSH: efficient indexing for high-dimensional similarity search," in *Proc. of the 33rd Int. Conf. on Very large data bases*, 2007.

[18] M. Muja and D. G. Lowe, "Fast Matching of Binary Features," in *Proc. of the Ninth Conf. on Computer and Robot Vision*, 2012.

[19] D. Schlegel and G. Grisetti, "Visual localization and loop closing using decision trees and binary features," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[20] Y. Feng, L. Fan, and Y. Wu, "Fast Localization in Large-Scale Environments Using Supervised Indexing of Binary Features," *IEEE Trans. on Image Processing*, vol. 25, no. 1, pp. 343–358, 2016.

[21] M. Komorowski and T. Trzciński, "Random Binary Search Trees for Approximate Nearest Neighbour Search in Binary Space," in *IEEE Trans. on Pattern Analysis and Machine Intell.*, pp. 473–479, 2017.

[22] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[23] A. J. Glover, W. P. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "OpenFABMAP: An open source toolbox for appearance-based loop closure detection.," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[24] S. Lowry and H. Andreasson, "Lightweight, Viewpoint-Invariant Visual Place Recognition in Changing Environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 957–964, 2018.

[25] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. on Robotics*, 2012.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[27] M. Warren, D. McKinnon, H. He, and B. Upcroft, "Unaided stereo vision based pose estimation," in *Proc. of the Australasian Conf. on Robotics and Automation*, 2010.

[28] J.-L. Blanco, F.-A. Moreno, and J. Gonzalez-Jimenez, "The Málaga Urban Dataset: High-rate Stereo and Lidars in a realistic urban scenario," *Int. Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014.