A Simple Hierarchical Pooling Data Structure for Loop Closure

Xiaohan Fei, Konstantine Tsotsos, Stefano Soatto

UCLA Vision Lab
University of California, Los Angeles
{feixh, ktsotsos, soatto}@cs.ucla.edu

Abstract. We propose a data structure obtained by hierarchically pooling Bag-of-Words (BoW) descriptors during a sequence of views that achieves average speedups in large-scale loop closure applications ranging from 2 to 20 times on benchmark datasets. Although simple, the method works as well as sophisticated agglomerative schemes at a fraction of the cost with minimal loss of performance.

Keywords: loop closure; hierarchical pooling; Bag-of-Words; descriptor aggregation

1 Introduction

We tackle the problem of *loop closure* in vision-based navigation. This is a particular classification task whereby a training set of images is indexed by location, and given a test image one wants to query the database to decide whether the former is present in the latter, and if so return the indexed location. This is closely related to *scene recognition*, where the focus is on a particular instance, as opposed to an object class (we want to determine whether we are at particular intersection in a given city, not whether we are at *some* intersection of *some* urban area). As such, test images are only subject to *nuisance variability* due to viewpoint, illumination and partial occlusion from moving objects, but otherwise there is no *intrinsic* (intra-class) variability.

The state-of-the-art for image retrieval is based on convolutional neural network (CNN) architectures, trained to marginalize nuisance and intrinsic variability. In a discriminatively trained network, the compositionality property afforded by linear convolutions, while critical to model intra-class variability, is unhelpful for loop closure, as there is no intrinsic variability. At the same time, a CNN does not respect the topology of data space at higher levels of the hierarchy, since filters at any given layer are supported on the entire feature map of the previous layer. In loop closure, locality is key, and while one could retrieve from the feature map the locations that correspond to active units, this requires some effort [26].

Given the critical importance of loop closure in location services ranging from smartphones to autonomous vehicles, we focus on its peculiarities, and attempt

to harvest some of the components of neural networks to improve the state-of-the-art. Stripped of the linear convolutions (we do not need to model intrinsic variability) and ReLu, what we have left is a *hierarchical spatially pooled data structure built upon local photometric descriptors [11, 18].* There are no filters, and no learning other than the trivial pooling of local descriptors. Motivated by this intuition, we propose a new hierarchical representation for loop closure, detailed in Sec. 2.

Loop closure is also closely related to location, or "place," recognition [35, 33, 6] and large-scale visual search [21, 5, 13], but with some important restrictions.

First, both previous data (training images) and current (test, or query) data are usually available as time-indexed sequences, even if they are captured by different agents, and training images may be aggregated into a "map" [14] or reduced to a collection of "keyframes" [20]. Second, as a binary classification task (at each instant of time, a loop closure is either detected or not), the cost of missed detections and false alarms are highly asymmetric: We pay a high price for declaring a loop closure that isn't, but there is minor harm in missing one, as temporal continuity affords many second chances in subsequent images. This is unlike large-scale image retrieval, where we wish to find what we are looking for (few missed detections, or high recall) even if we have to wade through some irrelevant hits (many false alarms, or low precision).

Like image retrieval, however, the challenge with loop closure is scaling. In navigation applications, it may be hours before we return to a previously seen portion of the scene. Therefore, we have to store, and search through, hundreds of thousands to millions of images. Our goal in this paper is to *design a hierarchical data structure that helps speed up matching by leveraging on the two domain-specific constraints above:* temporal adjacency, and high precision.

Assuming continuous trajectories, the first translates to proximity in pose space $SE(3)$ (position and orientation). For the second, the best trade-off with missed detections can be achieved by testing every datum in the training set via *linear search accelerated via an inverted index.* Our goal is to achieve similar performance at a fraction of the cost compared to inverted index search. This cannot be achieved in a worst-case setting. What matters instead is *average performance* trading off precision with computational cost. We evaluate such average performance empirically on the *KITTI* [9], *Oxford* [6] and *TUM RGB-D* [29] datasets, as well as demonstrate extensions to general image retrieval on the *ukbench* [21] and *INRIA Holidays* [13] datasets. To demonstrate scalability, we also evaluate our algorithm on augmented datasets with around 40K images.

We propose a simple data structure based on hierarchical pooling of location likelihoods – in the form of sample distributions of BoW descriptors – with respect to the topology of pose space. In practice, this means simply constructing BoW descriptors, that represent the likelihood of the locations that generated them, and pooling them temporally in a fine-to-coarse fashion, either by averaging, summing, or taking the index-wise maximum.

While averaging likelihoods may seem counter-productive, in Sec. 2 we show it makes sense in the context of the classical theories of sampling and anti-

aliasing. In Sec. 3 we show that, despite its simplicity, it works as well as sophisticated agglomerative schemes at a fraction of the effort.

1.1 Related work

Loop closure is a key component in robotic mapping (SLAM) [37], autonomous driving, location services on hand-held devices, and for wearables such as virtual reality displays. Loop closure methods can be roughly divided into 3 categories: appearance-only, map-only and methods in between. Appearance-only methods [6] are essentially large-scale image retrieval algorithms, influenced by [21] and more in general the literature of BoW object recognition and categorization [27]. Map-only methods [15] use the data (images, but most often range sensors) to infer the configuration of points in 3D space, and then seek to match subsets of these points, often using variants of ICP [4] as a building block. These methods do not scale beyond a few hundreds of thousands of points, or thousands of keyframes, and are often limited to what is referred to as "short-term" loop closure [15], necessary for instances when complete loss of visual reference occurs while tracking. There are also a variety of map-to-image and image-to-map [25] methods that show great promise, but have yet to prove scalability to the point where the map spans tens if not hundreds of kilometers [6].

For scalability, the most common choice is to combine quantized local descriptors into a BoW and then use an inverted index. FAB-MAP [6] extends the basic setup by learning a generative model of the visual words using a Chow-Liu tree to model the probability of co-occurrence of visual words. FAB-MAP 2.0 scales further by exploiting sparsity to make the inverted index retrieval architecture more efficient. Starting from [8], SIFT or SURF descriptors were replaced by more efficient binary descriptors such as BRIEF [3] and ORB [23] to achieve comparable precision and recall to FAB-MAP 2.0 with an order of magnitude speed increase. Several recent mapping and localization systems adopt it as a module, including [17] and ORB-SLAM [19].

In addition to the specific loop closure literature, general ideas from spatial data structures and agglomerative clustering [31] are also relevant to this work, including k-d trees [24], dual trees and decision trees [10], as well as data structures used for retrieval such as pyramid matching [12] and its spatial version [16]. In more general terms, this work also relates to visual navigation and mapping, structure-from-motion, and location recognition, including the use of global descriptors [33].

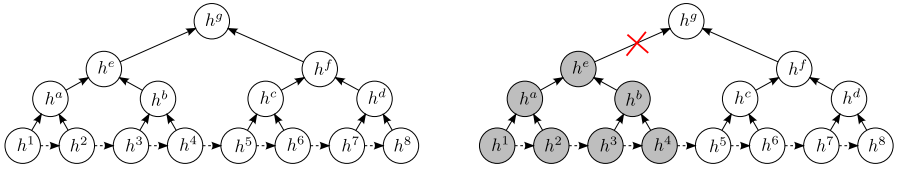
Our method can be considered appearance-only, but it is loosely informed by geometry, in the sense that the scene domain (pose space) provides the topology with respect to which we pool descriptors. Also closely related to our approach are [34, 32], which present techniques for merging only pairs of BoWs; in [5] queries are expanded by using retrieved and verified images, which is orthogonal to and can be viewed as a *query-end* version of our method.

2 Methodology

Since our focus is on a spatial structure that facilitates accelerated loop closure queries, we integrate components from recent state-of-the-art methods within our data structure and adopt such methods as a baseline, against which we compare our method. Specifically, we adopt [19] as a baseline, consisting of a BoW where each word is an element of a dictionary of descriptors obtained off-line by hierarchical k-means clustering, with each word weighted by its inverse document frequency. FAST detectors [22] and BRIEF descriptors [3] are employed, and TF-IDF [2, 1, 27] is used to weigh the BoW relative to the inverse document frequency. This standard pipeline, with different clustering procedures to generate the dictionary and different features, comprises most basic large-scale retrieval systems, including appearance-only loop closure. However, the number of false alarms in large-scale settings is crippling, so temporal consistency and geometric verification are typically used as correction mechanisms.

2.1 Hierarchical testing

Construction of hierarchy Our data structure can be interpreted as a hierarchical version of TF-IDF. To illustrate the method, we first assume that every frame is a "keyframe" and therefore we have a time-series of BoWs, obtained as described above, and organized into a linear structure or *un-oriented list*, as we wish to retrieve frames regardless of the direction of traversal. Each node is associated with a histogram, in the form of a BoW, representing the likelihood of a pose $g(t) \in SE(3)$ (position $T(t) \in \mathbb{R}^3$ and orientation $R(t) \in SO(3)$) given the data (the image at time t, $I(t)$): $h^t \doteq \text{BoW}(t) \sim p(I(t)|g(t))$, where the equivalence is up to normalization, and the density function is approximated with a histogram with N bins, equal to the size of the dictionary.



(a) Construction of hierarchy (b) Hierarchical testing

Fig. 1: (a) **Construction of hierarchy** for an 8-long sequence of (key)frames and constant branching factor of 2. Dashed lines indicate temporal order. (b) **Hierarchical testing**: If h^e does not score higher than the threshold, the whole sub-tree rooted at h^e (shaded) will not be searched. In the case of sum- or max-pooling, this would *not* introduce loss of precision compared to searching only the lowest level nodes.

We now construct a second level, or "layer", of the data structure, simply by pooling adjacent histograms (Fig. 1a). This is repeated for higher layers until either a maximum depth is reached, or until a single root node is left. Several

standard choices for the pooling operation are available which allow us to trade off between precision and cost (Sec. 3). Suppose h^p is the parent histogram which has child histograms $\{h^k\}, k = 1, 2 \dots K$. Both h^p and $h^k \in \mathbb{R}^N$. Mean- or average-pooling refers to $h^p = \frac{1}{K} \sum_{k=1}^{K} h^k$, sum-pooling refers to $h^p = \sum_{k=1}^{K} h^k$, and max-pooling refers to $h^p_i = \max_k \{h^k_i\}$, where $i = 1, 2 \dots N$. Once we have constructed the hierarchy for database histograms, raw histograms are used as queries for loop closure detection.

Query processing Similarities between pooled and query histograms are computed using the *intersection kernel* [30], that is the area of the intersection of the two histograms. Thus, if h^q (a query histogram) has bin values $h^q_1, \dots, h^q_N$, and similarly for h^p, we have that

$$\mathbb{I}(h^q, h^p) = \sum_{i=1}^{N} \min\{h^q_i, h^p_i\} \tag{1}$$

The intersection kernel is related to many divergence functions [36] as well as to metrics used in optimal transport problems.

Sum- and max-pooling operators have the following upper bound property when intersection kernel is applied: For a query histogram h^q, a parent histogram h^p and its child h^k in the database,

$$\mathbb{I}(h^q, h^p) > \mathbb{I}(h^q, h^k) \tag{2}$$

therefore if $\mathbb{I}(h^q, h^p) < \tau$, $\mathbb{I}(h^q, h^k) < \tau$ must hold.

Since our goal is to search for the closest match, or at least for all matches that exceed a threshold $\tau > 0$ (we seek large values of $\mathbb{I}$), if $\mathbb{I}(h^q, h^p) < \tau$, the chance of any of h^p's descendants exceeding the threshold is rare (or impossible, in the case of max- or sum-pooling as shown by the upper bound property), therefore we stop searching the sub-tree rooted at h^p (Fig. 1b).

Therefore, search in a hierarchical TF-IDF setting simply boils down to *greedy breadth-first search, while maintaining an inverted index for each layer.* If only one layer is used, this reduces to standard linear search using an inverted index.

A key point is that with sum- or max-pooling, the proposed method *has exactly the same precision-recall behavior as standard inverted index search* while still achieving a substantial speedup. With mean-pooling, a large speedup can be achieved with only a minimal loss of precision (Sec. 3).

Different trees with different depths and different branching factors can be constructed, trading off expected risk and computation time, characterized empirically in Sec. 3.4. In addition to a fixed depth and branching factor, one could devise more clever schemes to determine the topology of the tree, discussed in Sec. 2.2. However, we find that the benefit is limited compared to the straightforward fixed-topology architecture.

2.2 Keyframes and adaptive tree topology

So far we have assumed that the time-series of data $\{h^t\}_{t=1}^T$ is sampled regularly (at constant time or space intervals), but it can also be sampled adaptively, by exploiting statistics of the data stream to decide which samples, or *keyframes*, to use. The data structure above does not change, since all that is required is a topology or adjacency structure to construct the tree.

Adaptive (sub)-sampling can be done in many ways, and there are a wide variety of standard heuristics for selecting keyframes. Our goal here is not to determine the best method for selecting keyframes, but to focus on the data structure regardless of the sub-sampling mechanism. Consequently, we limit ourselves to constructing it either on the raw time series, or on any subsampling of it, as generated by standard keyframe selection methods.

Just like selecting keyframes, building the hierarchy can be understood as a form of (sub)-sampling. Regardless of whether subsampling is regular (as in building the tree above) or adaptive (as in selecting keyframes), classical sampling theory [28] suggests that what should be stored at the samples is *not* the value of the function, but the local average relative to the topology of the domain where the data are defined (*anti-aliasing*). This lends credence to the use of mean-pooling, which initially may seem counter-intuitive since our goal is to maintain high precision.

In our case, the domain is time, or the order of keyframes, as a proxy of location in $SE(3)$. The range of the data is the space of likelihood functions, approximated by histograms h^t. Therefore, anti-aliasing simply reduces to averaging neighboring histograms. The study of the optimal averaging, both in terms of support and weights, is beyond our scope here, where for mean-pooling we simply average nearest neighbors in the tree topology relative to a uniform prior. We do not delve into considering more sophisticated anti-aliasing schemes, since we have found that simple topologies yield attractive precision-computational cost trade-off, which is unlikely to be significantly disrupted by fine-tuning the weights.

The practice of averaging likelihood functions as a way of anti-aliasing descriptors has also been recently shown by [7] in the context of pooling local descriptors for correspondences in wide-baseline matching. Our method can be considered an extension (or special case) where the correspondence and pooling are performed in time, and the descriptors are histograms of visual words, a mid-level representation, rather than histograms of gradient orientation, the result of low-level processing.

While the choice of heuristics for keyframe selection has no effect on our method, which can be applied to the raw time series or to the sequence of keyframes, the same (adaptive sampling) heuristics used to (down)-sample keyframes from the regularly sampled images could be used to aggregate nodes at one level into parents one level above. This would give rise to trees having different levels of connectivity at different layers, and indeed potentially at each node.

We have found that, in practice, these heuristics fail to yield significant performance improvements when compared to trees with fixed topology having con-

stant splitting factors that match the average of their adaptive counter-part. Representative experiments are shown in Sec. 3.4.

3 Evaluation

The most important evaluation for the proposed method is to test performance in-the-loop when incorporated into a real system (ORB-SLAM [19], in this case), discussed in Sec. 3.2 where we find a 65% reduction in mean query time with no loss in localization performance and no missed loop closures relative to the baseline. We investigate query-time reduction and precision-recall behavior while varying vocabulary size and tree topology in Sec. 3.3 and Sec. 3.4, respectively. In Sec. 3.5 we augment standard datasets to explore various test-time scenarios, and Sec. 3.6 presents a generalization of our method to other image retrieval tasks. Sec. 3.1 discusses the datasets and methodology used throughout the evaluation.

3.1 Datasets and methodology for loop closure

We perform experiments using the common loop closure datasets of *KITTI*, *Oxford City Centre*, and *Oxford New College* [9, 6]. The *KITTI* dataset consists of several sequences on the order of 1000 stereo pairs in length. To provide additional experimental evaluation at large scale, we augment *KITTI* by concatenating all sequences, to form the *concatenated KITTI* dataset consisting of approx. 40K images. For all sequences we construct the data structure using all frames unless otherwise noted, in which case we adopt the keyframe selection strategy of our baseline (Sec. 3.2).

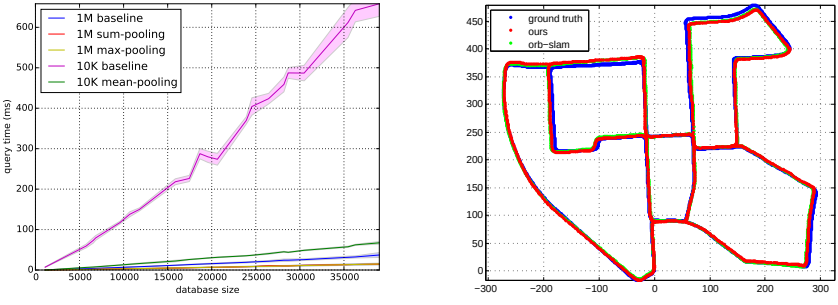
Unless otherwise stated, we build the hierarchical data structure using the left stereo images of the sequences (when stereo is available) and evaluate loop closure correctness using the provided ground truth poses. The evaluation protocol is as follows: traverse the sequence and insert BoW of images into the database incrementally, while using each image to query the database before it is added. Two images are regarded as a correct match if they were taken within 15 meters of each other. To avoid trivial matches, we prevent the query from matching temporally adjacent images. This evaluation protocol mimics loop closure in a practical SLAM system, which we test in Sec. 3.2.

To evaluate matching, missed detection and false alarms are traded off by an arbitrary choice of threshold, as in any detection algorithm. Since the threshold affects the average query time (we can make that quite short by choosing a threshold that yields no false alarms while rejecting every hypothesis) we must come to a reasonable choice. Unless otherwise stated, we adopt the following policy: We generate precision-recall curves on *KITTI* 00. Then, we select the smallest threshold that yields zero false alarms and use it on other sequences. Of course, that may yield a non-zero false alarm rate in datasets that are not used in setting the threshold, but this (as is customary) can be handled by verification steps afterwards. This is a limitation inherent to the choice of image representation, in this case Bag-of-Words, and not a sensitivity that our hierarchical data structure is designed to circumvent.

3.2 In-the-loop with the baseline

We use components of ORB-SLAM [19], made available by the authors, as the baseline for our experiments. We use this as a *black box* and implement our hierarchy atop its single-layer inverted index architecture for performing image queries. As a result, we also inherit some of the limitations of its components (e.g. keyframe selection, discriminability of quantized descriptors and BoW representations, sensitivity to matching threshold selection), which are common to the majority of SLAM systems.

We first show that when using ORB-SLAM *as is*, with no change in thresholds or tuning, a significant reduction in image query time can be achieved simply by applying our max-pooling hierarchy, which by construction achieves identical precision-recall performance to the original system, missing no loop closures that may be critical to pose-graph optimization algorithms. In Fig. 2b, we compare the trajectories estimated by ORB-SLAM with and without our max-pooling hierarchy on *KITTI*. Errors relative to ground truth are similar (within 1σ of each other over multiple trials); mean query times are reduced by 65% (2.04ms from 5.80ms). No loop closures are missed by our max-pooling method that would not be missed without our data structure, confirming that improvement in speed comes at no loss of classification performance. In Fig. 2a we show this speedup holds with increasing scale by showing query times for the *concatenated KITTI* dataset for different vocabulary sizes (Sec. 3.3) and various pooling strategies using the methodology of Sec. 3.5.



(a) Scaling (b) Comparison to ORB-SLAM

Fig. 2: (a) **Scaling:** Timings for concatenated *KITTI* sequences (approx. 40K images) with 1M and 10K vocabularies. (b) Comparison to **ORB-SLAM** with and without our data structure. Multiple trials yield nearly identical trajectories with and without our data structure, with no loop closures missed while achieving a 2-3x speedup.

3.3 Varying vocabulary size

Some may argue that a speedup could be easily gained by just using a larger vocabulary. It is true that with a larger vocabulary, each visual word is associated

with a much smaller list of documents in the inverted index system which leads to shorter query time. However, the vocabulary size should be determined by the performance of the specific task as well as the volume of the data and a larger vocabulary is not always better. A larger vocabulary has finer division of feature space compared to a smaller vocabulary but is also more sensitive to quantization errors (two slightly different images may have completely different histograms). In this case, mean-pooling may not be ideal as shown in Fig. 4c and 4d. However, sum/max-pooling can still be applied to gain further speedup while maintaining same precision-recall as shown in Fig. 3c and 3d, and also on augmented dataset as shown in Fig. 2a.

3.4 Varying tree topology

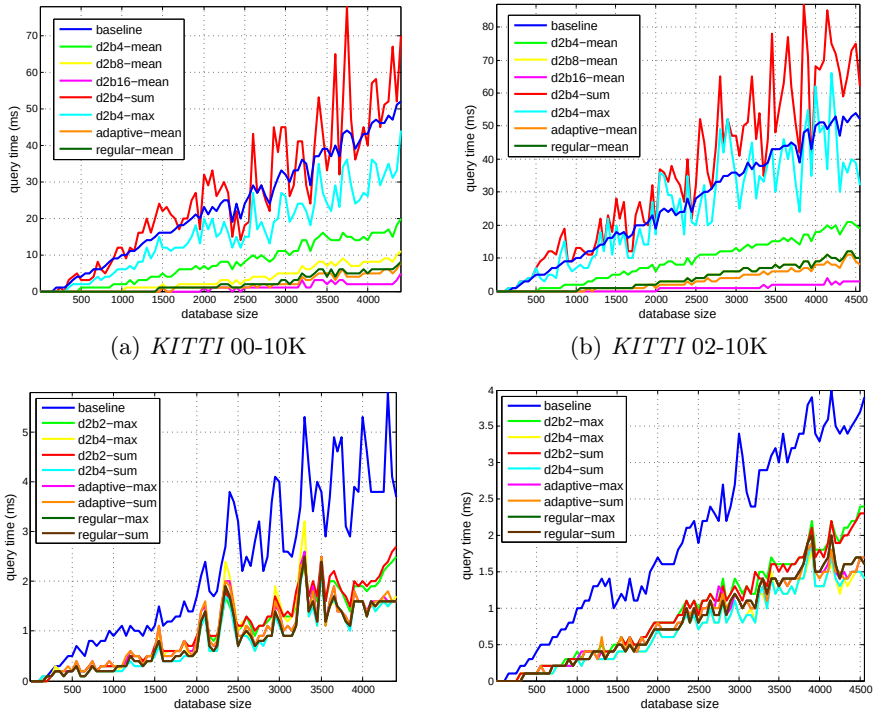
Variable depth and branching factor Fig. 3 shows timings of the baseline and our algorithm with different topologies and pooling schemes at the same threshold on two of the *KITTI* sequences with many loop closures. Only time to query the database is counted, time for feature extraction and descriptor quantization are excluded. Fig. 4a and Fig. 4b show precision-recall curves for the mean-pooling variants. We use $d_i b_j$-X to denote a hierarchy with i layers, a branching factor of j and pooling strategy X. Note that for baseline and our proposed algorithm with configuration $d_2 b_4$-mean and $d_2 b_8$-mean, the precision-recall curves are nearly identical, while our approach is 2-5 times faster. For configuration $d_2 b_{16}$-mean, while its performance is slightly worse, it achieves *an order of magnitude speedup* relative to the baseline.

As mentioned in Sec. 2.1, sum/max-pooling have *exactly the same precision-recall behavior* as the baseline. In these two datasets, sum/max-pooling are slightly slower than inverted index search. Since both of these operations rapidly reduce sparsity in the histograms, we expect slower performance relative to mean-pooling. However, sum/max-pooling have their advantages when a much larger vocabulary is used as shown in Sec. 3.3.

Adaptive domain-based clustering In addition to the baseline algorithm, we generate a second baseline by applying the same algorithm to keyframes, rather than to all stored images. In principle, the heuristics involved in the selection of keyframes could be propagated to all nodes of the data structure, as discussed in Sec. 2.2. However, our experiments indicate that this yields minor benefits compared to simple averaging. The second row of Tab. 2a shows average time-cost rate [1] for searching via an inverted index among keyframes, which is worse than searching in a simple hierarchy built on raw images, as shown in the second row of Tab. 1c. A simple regular sampling strategy on top of keyframes can speedup searching by a large margin as shown in Tab. 2a.

Instead of a fixed topology of the data structure, corresponding to regular grouping, we can consider adaptive grouping based on a variety of criteria.

[1] Time-cost rate is defined as the increase of query time per thousand (1k) images in the database. Average time-cost rate is the average of time-cost rates computed for all sequences in each dataset.



(a) *KITTI* 00-10K (b) *KITTI* 02-10K

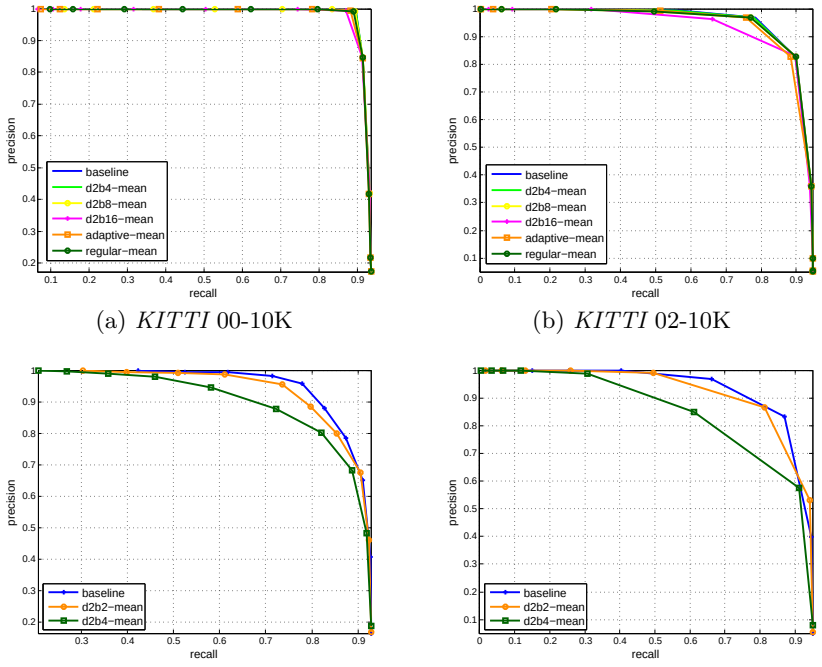
(c) *KITTI* 00-1M (d) *KITTI* 02-1M

Fig. 3: Timings of baseline and proposed algorithm with different topologies and pooling strategies on *KITTI* dataset 00 and 02 using *all* frames. $d_i b_j$ -X: a hierarchy with i layers, a branching factor of j and pooling strategy X. Adaptive sampling: spectral clustering in $SE(3)$. Regular sampling: sampling at the average rate of adaptive sampling scheme. Baseline: inverted index search. Two different vocabulary sizes (10K and 1M) are considered.

Adaptive sampling, or grouping, based on *geometry* includes performing spectral clustering in $SE(3)$. Curves in Fig. 3 indicate that adaptive sampling achieves marginal improvements compared to regular sampling at a constant rate equal to the average of the adaptive sampling rate. Similarly, parallax-based sampling, based on clustering only the translational component of pose, also yields underwhelming improvements. We do, however, expect adaptive sampling to win in some cases, as it has in a number of smaller-scale experiments we conducted with different motion characteristics from smooth driving, for instance the *TUM RGB-D* dataset (Fig. 5) [29].

3.5 Quantifying speedup using synthetic ground-truth

Depending on the particular query image, our method could reduce or increase search time relative to the mean. The former occurs when correspondence fails early allowing us to rule out subsequent tests at finer scales. However, in the worst-case we may end up performing more comparisons than inverted index search when the test reaches the finest scale too often. In practice, what matters



(a) *KITTI* 00-10K (b) *KITTI* 02-10K

(c) *KITTI* 00-1M (d) *KITTI* 02-1M

Fig. 4: Precision-recall curves of baseline and proposed algorithm with different topologies on *KITTI* dataset 00 and 02 using *all* frames. Two different vocabulary sizes (10K and 1M) are considered. Notations have the same meanings as in Fig. 3.

is that our algorithm shortens test time *on average* during long sequences. Since most *KITTI* sequences contain few or no loop closures, we generate synthetic positive and negative queries as follows: For sequences 01 to 21, we generate positive queries by sampling the right stereo images of each sequence (slightly different from the left images from which we constructed the database), and generate negative queries by sampling images from sequence 00. For the *Oxford* datasets, we construct the database using odd-numbered images, generate positive queries from the even-numbered images, and negative queries again from *KITTI* 00.

Overall performance is measured by combining both sets of queries. Of course, even in the negative case our algorithm could find erroneous correspondences, which are then labeled as false alarms. Similarly, we may find no correspondence in the former case (missed detection). We use average time-cost rate to evaluate how the searching algorithm scales with size of the database. Tab. 1 reports experiment results on raw *KITTI*. Tab. 2 reports average speedup when keyframe selection is applied on both *KITTI* and *Oxford*. Fig. 2a shows linear scaling of average query time on the much larger *concatenated KITTI*. Practical deployment on even larger datasets typically comes with context (*e.g.* GPS or odometry) that limits the data volume.

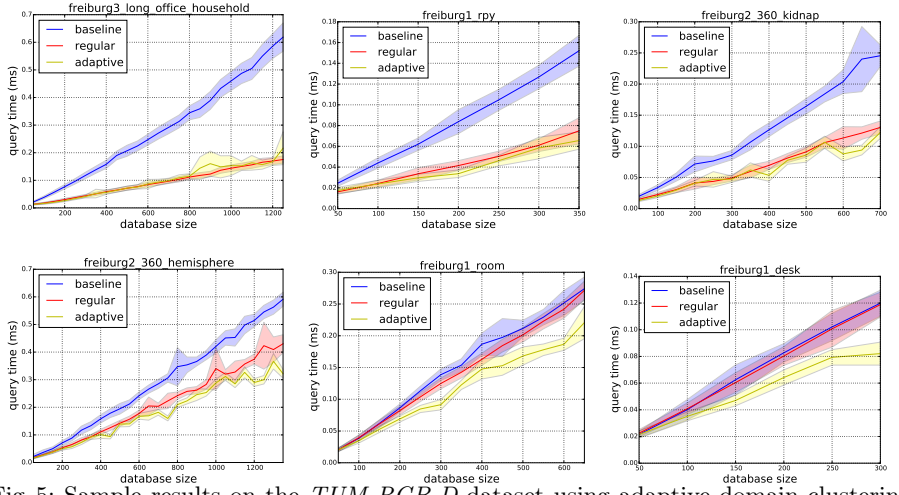


Fig. 5: Sample results on the *TUM RGB-D* dataset using adaptive domain clustering (Sec. 2.2). The experiment setup is similar to that for the *Oxford* dataset in Sec. 3.5. Adaptive (yellow) improves with more exciting motion (left to right, up to down). Limited speedup relative to baseline due to very small dataset size. Variance shown is derived from multiple trials with sightly differing cluster assignments.

3.6 Experiments in image retrieval tasks

Although our approach is geared towards the loop closure scenario, its usage is not restricted to it. A hierarchical structure of this form could be built on top of any histogram-based representation of images where some proxy of topology is available. In more general settings when a temporal stream of images is unavailable, extra labeling information, such as geotags, class labels, or textual annotations could be used. A hierarchy can be constructed using affinity between these alternate forms of metadata, provided that affinity implies proximity in the solution space. We test this using two publicly available image retrieval benchmarks: *ukbench* [21] and *INRIA Holidays* [13].

ukbench [2] consists of 2550 groups of 4 images each (10200 total). Each group contains the same object under different viewpoint, rotation, scale and lighting conditions. We use the same evaluation protocol provided by the author: Count how many of 4 images are top-4 when using a query image from that set of four images. We use pre-computed visual words provided by the authors, which are quantized SIFT descriptors using a 1M vocabulary.

INRIA Holidays [3] contains 500 image groups (1491 total), each of which represents a distinct scene under different rotations, viewpoint and illumination changes, blurring, etc. Performance is measured by mean average precision (mAP) averaged over all 500 queries. We use the 4.5 million SIFT descriptors and 100K vocabulary provided by the authors.

The baseline remains to search using an inverted index system. We use a three-layer hierarchy with the original histograms at the bottom layer. At the

[2] `http://vis.uky.edu/~stewe/ukbench/`

[3] `https://lear.inrialpes.fr/~jegou/data.php`

Table 1: Average time-cost rate and speedup over 21 sequences of *KITTI* using *all* frames. 1st col: grouping strategies. 2nd col: pooling operations. 3rd col: average time-cost rate, which describes how the query time increases per 1k images inserted into the database. In 1a, 1b and 1c, a 10K vocabulary is used; in 1d, a 1M vocabulary is used.

(a) positive queries; KITTI - 10K

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	10.07	1.00
hierarchical	mean	**0.69**	**14.59**
	sum	8.70	1.16
	max	6.65	1.52

(b) negative queries; KITTI - 10K

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	9.86	1.00
hierarchical	mean	**0.34**	**29.00**
	sum	6.28	1.57
	max	5.04	1.96

(c) overall; KITTI - 10K

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	9.88	1.00
hierarchical	mean	**0.38**	**26.00**
	sum	7.92	1.25
	max	6.06	1.63

(d) overall; KITTI - 1M

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	0.64	1.00
hierarchical	mean	N/A	N/A
	sum	**0.30**	**2.13**
	max	**0.30**	**2.13**

Table 2: A comparison of search in flat and hierarchical structure on *KITTI* and *Oxford* dataset. Notations have the same meanings as in Tab. 1 except that 3rd column describes average time-cost rate over the 21 *KITTI keyframe* sequences and all 4 sequences in the *Oxford* dataset respectively. The keyframes are generated by running ORB-SLAM.

(a) overall; KITTI - 10K

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	8.97	1.00
hierarchical	mean	**0.88**	**10.14**
	sum	7.87	1.14
	max	6.00	1.50

(b) overall; Oxford - 10K

structure	pooling	rate(ms/1k)	speedup
inverted index	N/A	6.98	1.00
hierarchical	mean	**1.61**	**4.34**
	sum	4.71	1.48
	max	4.20	1.66

second layer, histograms belonging to the same object/scene are pooled (pooling based on prior information available about the data and problem space). At the top layer, we compare two different strategies to build the hierarchy: Random grouping and greedy affinity grouping. Random grouping: We randomly group every N histograms from the second layer. Greedy affinity grouping: We greedily group every N histograms based on their nearest neighbors in affinity (which is the histogram intersection score). In each setup, we also compare the different choices of pooling operators. Tab. 3a and Tab. 3b show results on the *ukbench* and *INRIA Holidays* datasets with $N = 16$.

In these image retrieval tasks, we *completely discard the threshold and only search down those nodes which have top 10 highest scores*. Thus even for sum/max-pooling, the precision-recall behavior should be different from the baseline. All hierarchical approaches, regardless of pooling operation and grouping scheme, are faster than the baseline. The observation that speedup is available even for the random grouping scheme shows that the speedup does not just hinge on grouping similar images, though grouping similar images can boost the speedup

Table 3: A comparison of search in flat and hierarchical structure on *ukbench* and *INRIA Holidays*. 1st col: grouping strategies. 2nd col: pooling operations. 3rd col: average query time. *ukbench* takes average number of top-4 retrieved images as score. *INRIA Holidays* takes mAP as evaluation metric.

(a) ukbench

structure	pooling	time(ms)	speedup	score
inverted index	N/A	1.47	1.00	2.72
Random hierarchical	mean	0.38	3.87	2.80
	sum	**0.37**	**3.97**	**2.83**
	max	0.39	3.77	2.82
Greedy affinity hierarchical	mean	0.38	3.87	2.80
	sum	0.38	3.87	**2.83**
	max	**0.37**	**3.97**	2.82

(b) INRIA Holidays

structure	pooling	time(ms)	speedup	mAP
inverted index	N/A	9.11	1.00	0.56
Random hierarchical	mean	**5.57**	**1.63**	0.58
	sum	6.19	1.47	**0.63**
	max	6.24	1.46	0.62
Greedy affinity hierarchical	mean	**5.58**	**1.63**	0.57
	sum	6.82	1.34	**0.63**
	max	6.53	1.40	0.62

further as we have shown in previous experiments on the driving data. We also notice improved score/mAP in these two experiments, likely due to the grouping of histograms of the same object/scene at the second layer of our hierarchy and the top-4 scoring mechanism imposed by the benchmark.

4 Discussion

We have presented a hierarchical data structure consisting of pooled local descriptors representing the likelihood of locations given the images they generate, while maintaining an inverted index at each level of the data structure. While mean-pooling of histograms may seem counter-productive, it is a sensible choice when considered an anti-aliasing procedure in the context of classical sampling theory, where the data structure, as well as keyframes, are tasked with *down-sampling* the native rate. We have compared several pooling strategies, and found that mean-pooling provides the most speedup at a small cost to performance; sum-pooling has the upper-bound property and accelerates search to a reasonable degree without loss of performance; and max-pooling shares the same property with sum-pooling but exhibits a larger speedup due better approximating the nodes below it.

For simplicity, we chose a fixed topology (depth and branching factor) and studied the resulting performance empirically. We have found that sophisticated heuristics do not improve performance enough to justify the added complexity. We have benchmarked our scheme on public datasets, where we have shown that even a shallow tree can significantly cut down on test time with minimal impact to precision, which is the main goal of loop closure.

Acknowledgements

This work was supported by AFRL FA8650-11-1-7156, ONR N00014-15-1-2261 and ARO W911NF-15-1-0564.

References

1. Aizawa, A.: An information-theoretic perspective of tf–idf measures. Information Processing & Management 39(1), 45–65 (2003)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
3. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: Computer Vision–ECCV 2010, pp. 778–792. Springer (2010)
4. Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P.: The trimmed iterative closest point algorithm. In: Pattern Recognition (ICPR), 2002 IEEE Intl. Conf. on. vol. 3, pp. 545–548. IEEE (2002)
5. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: Computer Vision (ICCV), 2007 IEEE Intl. Conf. on. pp. 1–8. IEEE (2007)
6. Cummins, M., Newman, P.: Highly scalable appearance-only slam-fab-map 2.0. In: Robotics: Science and Systems. vol. 5. Seattle, USA (2009)
7. Dong, J., Soatto, S.: Domain size pooling in local descriptors: Dsp-sift. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conf. on (2015)
8. Galvez-Lopez, D., Tardos, J.D.: Real-time loop detection with bags of binary words. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Intl. Conf. on. pp. 51–58. IEEE (2011)
9. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. Intl. J. of Robotics Res. p. 0278364913491297 (2013)
10. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. Pattern Analysis and Machine Intelligence, IEEE Trans. on 18(1), 1–14 (1996)
11. Girshick, R., Iandola, F., Darrell, T., Malik, J.: Deformable part models are convolutional neural networks. arXiv preprint arXiv:1409.5403 (2014)
12. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: Computer Vision (ICCV), 2015 IEEE Intl. Conf. on. vol. 2, pp. 1458–1465. IEEE (2005)
13. Jégou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Computer Vision–ECCV 2008, pp. 304–317. Springer (2008)
14. Jones, E., Soatto, S.: Visual-inertial navigation, localization and mapping: A scalable real-time large-scale approach. Intl. J. of Robotics Res. (april 2011)
15. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. of the 2007 6th IEEE and ACM Intl. Symp. on Mixed and Augmented Reality. pp. 1–10. IEEE Computer Society (2007)
16. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Computer Vision and Pattern Recognition (CVPR), 2006 IEEE Conf. on. vol. 2, pp. 2169–2178. IEEE (2006)
17. Lim, H., Lim, J., Kim, H.J.: Real-time 6-dof monocular visual slam in a large-scale environment. In: Robotics and Automation (ICRA), 2014 IEEE Intl. Conf. on. pp. 1532–1539. IEEE (2014)
18. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conf. on. pp. 5188–5196. IEEE (2015)
19. Mur-Artal, R., Montiel, J., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. Robotics, IEEE Trans. on 31(5), 1147–1163 (2015)

20. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on. pp. 1498–1505. IEEE (2010)
21. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Computer Vision and Pattern Recognition (CVPR), 2006 IEEE Conf. on. vol. 2, pp. 2161–2168. IEEE (2006)
22. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Computer Vision–ECCV 2006, pp. 430–443. Springer (2006)
23. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE Intl. Conf. on. pp. 2564–2571. IEEE (2011)
24. Samet, H.: The design and analysis of spatial data structures, vol. 85. Addison-Wesley Reading, MA (1990)
25. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2d-to-3d matching. In: Computer Vision (ICCV), 2011 IEEE Intl. Conf. on. pp. 667–674. IEEE (2011)
26. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Workshop at International Conference on Learning Representations (2014)
27. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Computer Vision (ICCV), 2003 IEEE Intl. Conf. on. pp. 1470–1477. IEEE (2003)
28. Smale, S., Zhou, D.X.: Shannon sampling ii: Connections to learning theory. Applied and Computational Harmonic Analysis 19(3), 285–302 (2005)
29. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Intelligent Robot Systems (IROS), 2012 IEEE/RSJ Intl. Conf. on (2012)
30. Swain, M.J., Ballard, D.H.: Color indexing. Intl. J. of Computer Vision 7(1), 11–32 (1991)
31. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. In: Proc. of the Allerton Conf. (2000)
32. Torii, A., Sivic, J., Pajdla, T.: Visual localization by linear combination of image descriptors. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE Intl. Conf. on. pp. 102–109. IEEE (2011)
33. Torralba, A., Murphy, K.P., Freeman, W.T., Rubin, M.A.: Context-based vision system for place and object recognition. In: Computer Vision (ICCV), 2003 IEEE Intl. Conf. on. pp. 273–280. IEEE (2003)
34. Turcot, P., Lowe, D.G.: Better matching with fewer features: The selection of useful features in large database recognition problems. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE Intl. Conf. on. pp. 2109–2116. IEEE (2009)
35. Ulrich, I., Nourbakhsh, I.: Appearance-based place recognition for topological localization. In: Robotics and Automation (ICRA), 2000 IEEE Intl. Conf. on. vol. 2, pp. 1023–1029. IEEE (2000)
36. Vasconcelos, N.: On the efficient evaluation of probabilistic similarity functions for image retrieval. Information Theory, IEEE Trans. on 50(7), 1482–1496 (2004)
37. Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., Tardós, J.: A comparison of loop closing techniques in monocular slam. Robotics and Autonomous Systems (2009)