UCLA ENGINEERING
Henry Samueli School of Engineering and Applied Science
Birthplace of the Internet
UCLAVISIONLAB

A Simple Hierarchical Pooling Data Structure for Loop Closure

ECCV'16
EUROPEAN CONFERENCE ON COMPUTER VISION

Xiaohan Fei, Konstantine Tsotsos, Stefano Soatto
{feixh, ktsotsos, soatto}@cs.ucla.edu

Overview

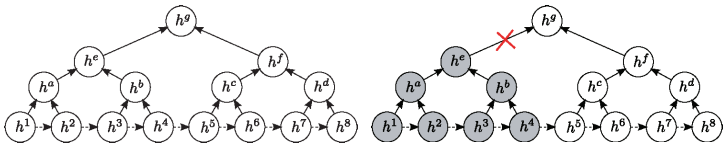
We propose a data structure obtained by **hierarchically pooling** Bag-of-Words descriptors during a sequence of views that achieves average **speedups** in large-scale **loop closure** applications ranging from **2 to 20 times** on benchmark datasets.

Loop closure 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.

The challenge with loop closure is **scaling**. Our goal here is to design a hierarchical data structure that helps speed up loop detection by leveraging on two domain-specific constraints: **temporal adjacency**, and **high precision**.

Construction of hierarchy

Given a list of histograms, we build a hierarchy on top of it recursively by pooling adjacent histograms.
Different pooling strategies:
max-pooling: $h_i^p = \max_{k=1\cdots K}\left(h_i^k\right), i = 1\cdots N$
sum-pooling: $h^p = \sum_{k=1}^{K} h^k$
mean-pooling: $h^p = \frac{1}{K}\sum_{k=1}^{K} h^k$
where h^p is a parent histogram; $h^k, k = 1\cdots K$, are its children. Each histogram has N bins, which is the vocabulary size.



(a) Construction of hierarchy (a) 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 a certain histogram does not score higher than the threshold, the sub-tree rooted at that node will not be searched.

Hierarchical testing

Goal: Given a query histogram and a threshold τ, find all the histograms scoring higher than τ in the database.
Intersection kernel is used to compare histograms

$$I(h^q, h^p) = \sum_{i=1}^{N} \min\{h_i^q, h_i^p\}$$

Baseline: Linear search accelerated by inverse index structure.
Hierarchical testing: If a parent histogram does not score higher than the threshold, prune the sub-tree rooted at that node. Each layer is also equipped with an inverse index structure.

Performance guarantee

For max-/sum-pooling, **no** histograms scoring higher than τ would be missed according to the following property

$$I(h^q, h^p) \geq I(h^q, h^k)$$

where h^q is a query, h^p is a parent node and h^k is its child.
For mean-pooling, parent histogram is similar to its children due to locality of the pooling operator and performance is only slightly hurt.

Extension to image retrieval

Hierarchy leveraging on extra labeling information to speed up image retrieval.

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

(a) *ukbench*

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

(b) *INRIA Holidays*

Table 1. A comparison of search in flat and hierarchical structure on *ukbench* and *INRIA Holidays*.

In-the-loop test

We use components of ORB-SLAM and build a hierarchy atop its single-layer inverse index based loop detection module.

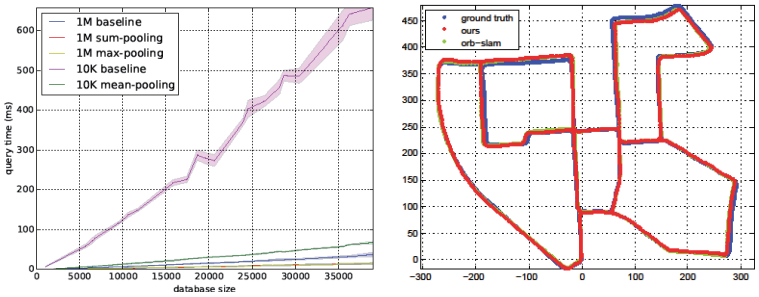


Fig.2: (a) **Scaling:** Timings for concatenated *KITTI* sequences (approx. 40K images) with 1M and 10K vocabularies. (b) Comparison to **ORB-SLAM** with and without our data structure. Multiple trials yield nearly identical trajectories with and without our data structure, with no loop closures missed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

Varying tree topology

We consider both fixed and adaptive tree topology.

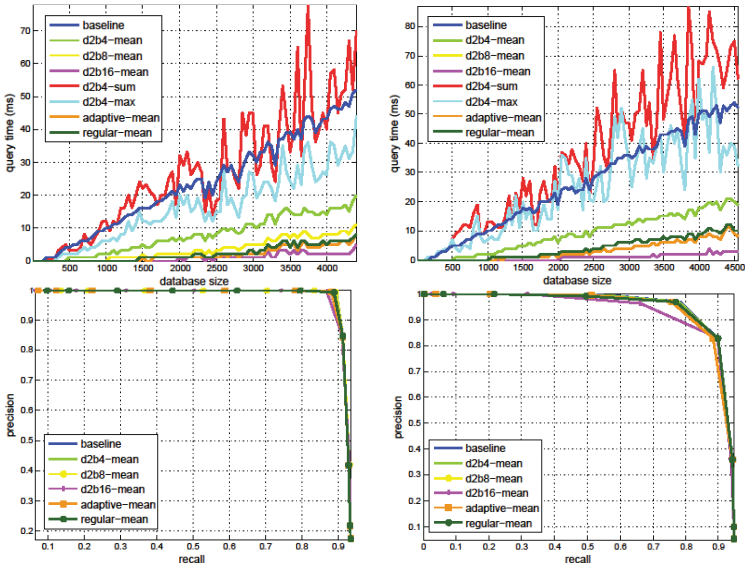


Fig. 3 Timings (top) and precision-recall curves (bottom) of baseline and proposed algorithm with different topologies and pooling strategies on *KITTI* dataset 00 and 02 using all frames. $d_i b_j$-X: a hierarchy with i layers, a branching factor of j and pooling strategy X. Adaptive sampling: spectral clustering in $SE(3)$. Regular sampling: sampling at the average rate of adaptive sampling scheme.

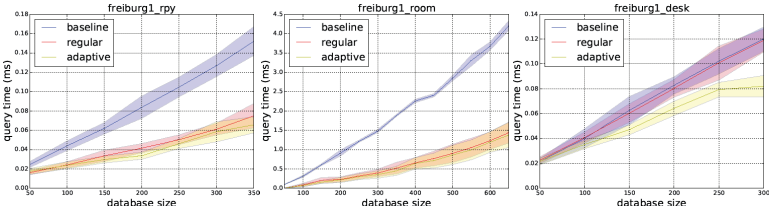


Fig 4. Sample results on the *TUM RGB-D* dataset using adaptive domain clustering. Adaptive (yellow) improves with more exciting motion.

Varying vocabulary size

Using a larger vocabulary reduces query time, but a sensible speedup is still available with our data structure. A larger vocabulary has finer division of descriptor space compared to a smaller vocabulary but is more sensitive to quantization errors. Vocabulary size should be determined by the task as well as the volume of the data.

Acknowledgements

Supported by AFRL FA8650-11-1-7156, ONR N00014-15-1-2261 and ARO W911NF-15-1-0564