Efficient Techniques for Accurate Visual Place Recognition

Master's Thesis in Robotics, Cognition, Intelligence

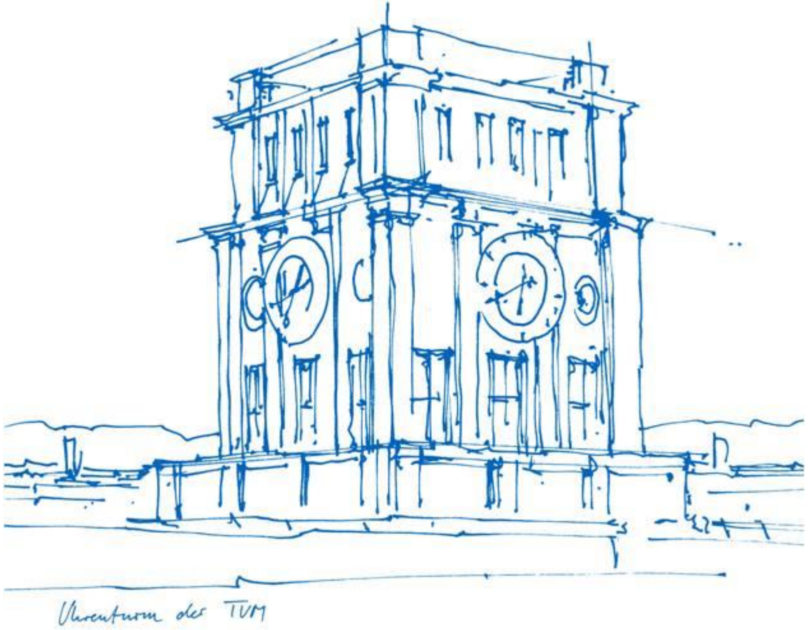
Tim Stricker

Technical University of Munich

Department of Informatics

Chair of Computer Vision and Artificial Intelligence

June 25, 2020

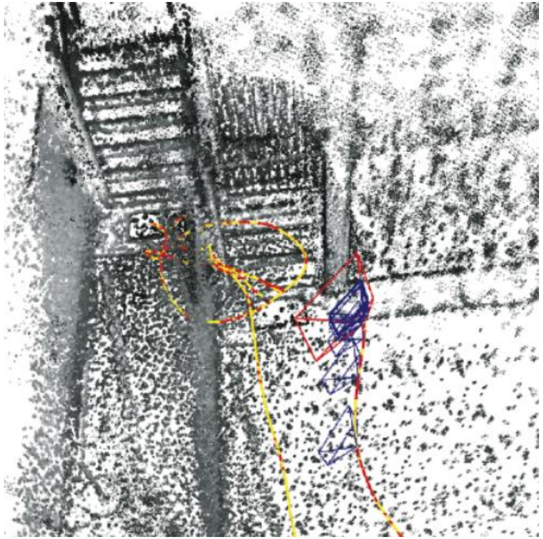


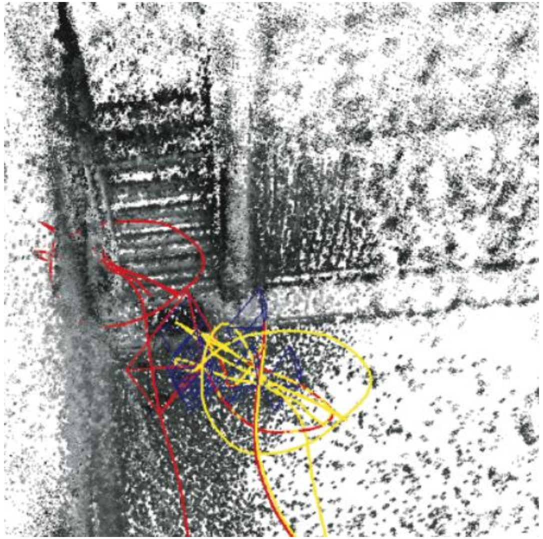
Introduction

Motivation, Problem Statement & Goals

Motivation

Loop Closure (Source: Gao *et al.* [1])





Problem Statement

Problem:

- Main performance drawback in SLAM / SfM: Image Matching
- Brute-force approach is extremely expensive
- Visual Place Recognition can be used to limit the search space

Extent of the Work:

- Visual Place Recognition based on local features
- Unordered image collections
- Pure appearance-based place recognition procedures
- Focus on efficient methods (real-time capability)

Goals of the Thesis

Overview of promising approaches:

- Visual place recognition, based on local features, pure image retrieval
- Additionally: Novel approach based on locality-sensitive hashing

Evaluation of place recognition methods:

- Newly developed benchmarking suite
- Parameter analysis, feature extractor influence, method comparison

Efficient implementation:

- Open-source library containing multiple place recognition approaches
- Improving efficiency of existing algorithms

Visual Place Recognition

Theory & Methods

Definition

Visual:

- Visual appearance of places
- Not the only possible source of data

Place:

- Many different definitions depending on context
- In our context: Different places have different appearance
- Perceptual aliasing can be a challenge

Source: Cummins and Newman [2]

Recognition:

- Perceiving something which is previously known
- In computer vision: Classifying a detection (*what* in contrast to *if* and *where)*

Components

Image Description:

- Describing images: Local, global and hybrid approaches
- We focus on locally extracted features

Mapping:

- Remembering previously visited places
- In our case: Database containing image representations (inverse index)

Belief Generation:

- Decision whether a perceived place has been visited
- Image similarity measures

Visual Bag of Words

Origins:

- Text retrieval: Finding relevant documents in a large collection
- Assumption: Similar documents contain a similar distribution of words

Transfer to image retrieval:

- Extract visual words from images (clustering)
- Represent images by occurrence or distribution of words

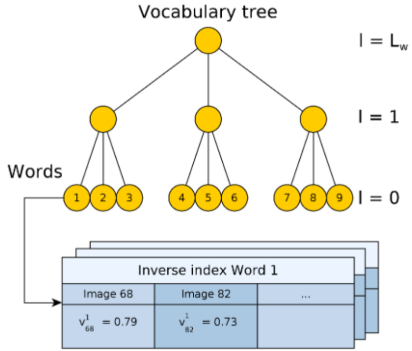
Advantages:

- Implicit pose invariance
- Simple and efficient implementation

Methods for Visual Place Recognition

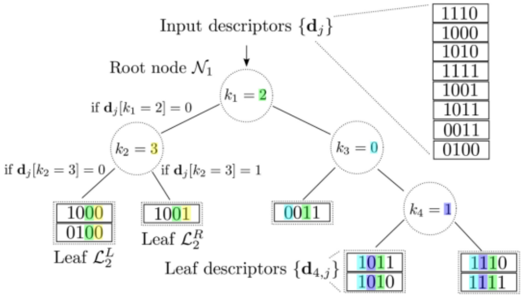
DBoW: Hierarchical Bag of Words [3]

- Vocabulary tree, constructed using hierarchical k-means++ clustering
- Cluster centers treated as terms in the bag-of-words scheme



HBST: Hamming Distance Embedding Binary Search Tree [4]

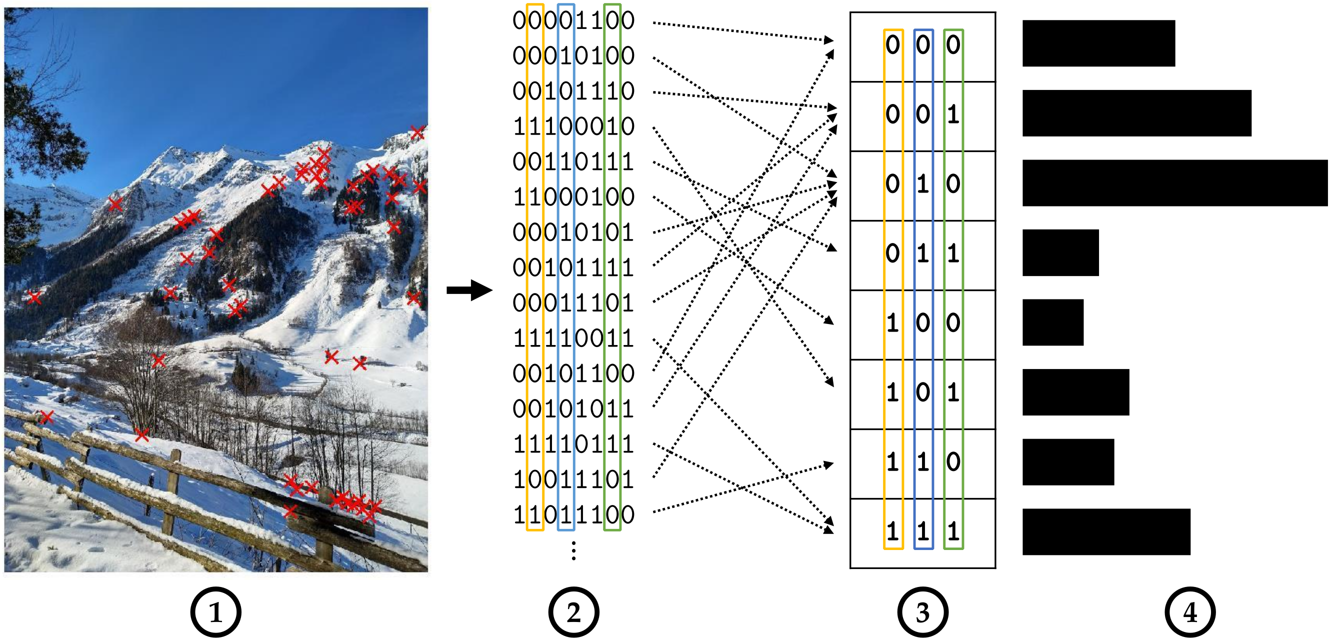
- Binary search tree, splitting based on bit indices
- Place recognition with voting scheme of descriptors in leaf nodes



HashBoW: Hashing-Based Bag of Words

- Clustering based on Locality-Sensitive Hashing (LSH)
- Hash codes treated as terms in the bag-of-words scheme
- Training: Entropy maximization of hash codes

HashBoW: Image Representation



Evaluation

Benchmarking Suite, Contents & Results

Benchmarking Suite

Data Preparation

- Download of datasets
- Conversion into unified format
- Python, YAML

Data Processing

- Feature extraction
- Place recognition methods
- Output of results
- C++, OpenCV, YAML

Results Evaluation

- Analyze and visualize results
- Accuracy and run time
- Python, Jupyter Notebook

Evaluation Contents

Methods:

- DBoW3
- HBST
- HashBoW

Feature Extractors:

- AKAZE
- BRISK
- ORB

Datasets:

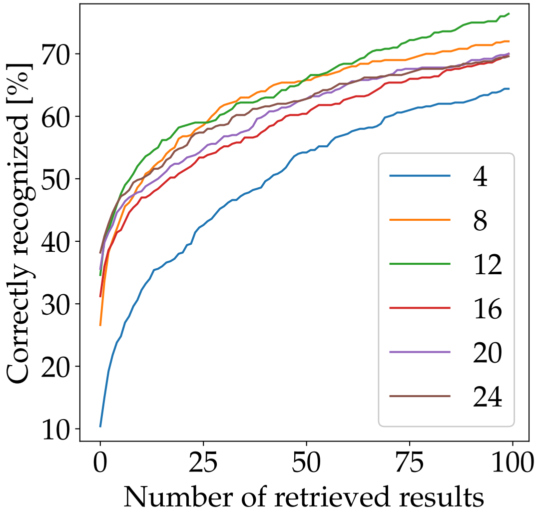
- Oxford Buildings
- Paris Buildings
- INRIA Holidays

Metrics:

- Percentage of correctly recognized places
- Recall
- Cumulated run time

Evaluation

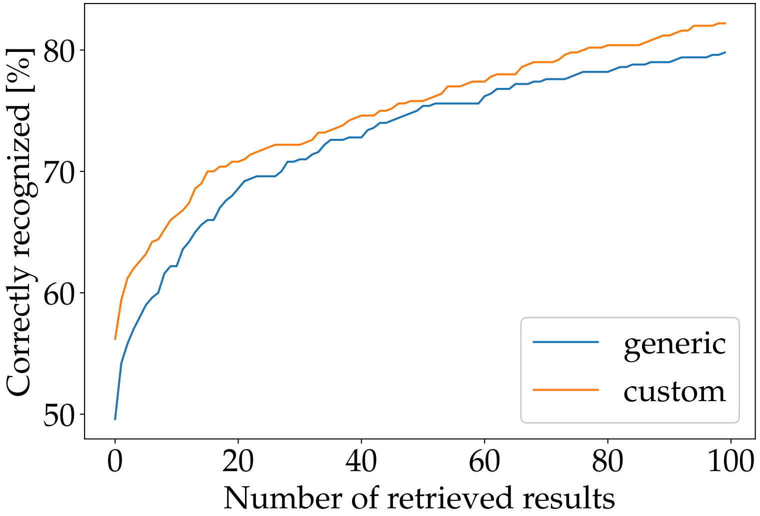
Parameter Analysis: HashBoW



Bits	Add	Query
4	0.12 s	0.19 s
8	0.15 s	1.23 s
12	0.27 s	3.19 s
16	0.56 s	2.09 s
20	0.84 s	1.11 s
24	1.19 s	0.71 s

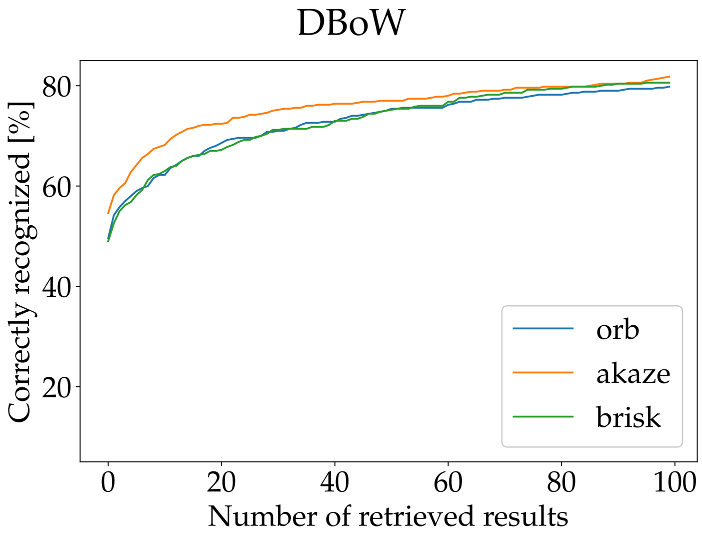
Evaluation

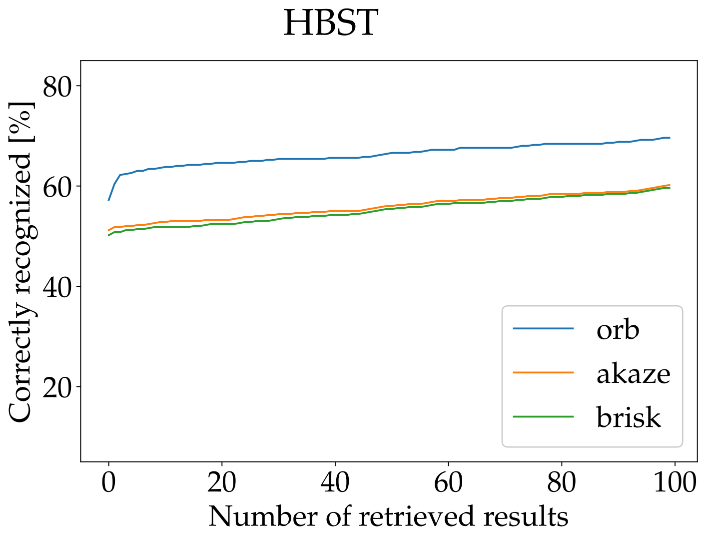
Influence of Training Dataset (DBoW)



Evaluation

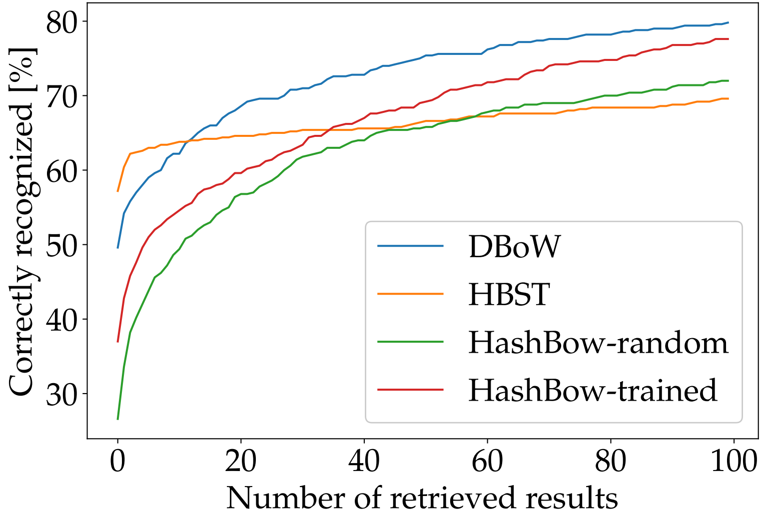
Influence of Feature Extractor: DBoW & HBST





Evaluation

Final Method Comparison



Method	Add	Query
DBoW	14.15 s	10.27 s
HBST	5.09 s	2.67 s
HashBoW-random	0.15 s	1.23 s
HashBoW-trained	0.32 s	3.74 s

→ HashBoW-random: 8 bits, random bit sampling
→ HashBoW-trained: 12 bits, entropy maximized hash codes

Efficient Implementation

Motivation, Structure & Improvements

Motivation & Goals

Motivation:

- Structure of different bag-of-words approaches is very similar
- No reference collection of algorithms available
  → Performance, code quality and usage can vary widely
- DBoW: Accurate but comparatively slow

Goals of the new library:

- Well-documented: Easy to use and understand
- Extensible: New methods can be added easily
- Lightweight: Straightforward to incorporate
- Efficient reference implementations

Library Structure

Descriptors

- Binary: std::bitset
- Real-valued: Eigen::Matrix
- Additional wrapper template

BoW Generators

- Abstract base class defines interface
- Actual implementation in derived classes
- Currently implemented: HashBoW, DBoW

BoW Vectors

- Mimics std::vector interface
- Contains word identifiers and values
- Additional normalization functionality

Database

- Generic database implementation
- Inverted index for fast queries
- Scoring: L_1, L_2, Cosine Similarity

Improvements

HashBoW

Training procedure:

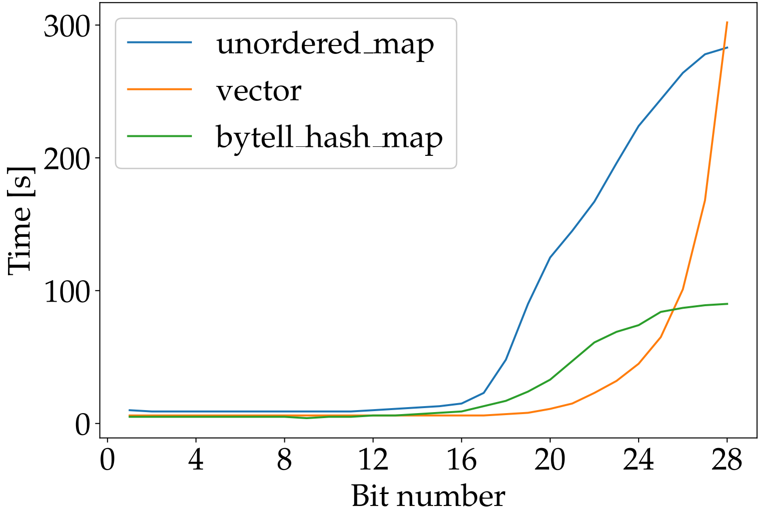
- Entropy maximization of hash codes
- Count associated descriptors for every hash code
- Trade-off between run time and memory efficiency for large hash codes

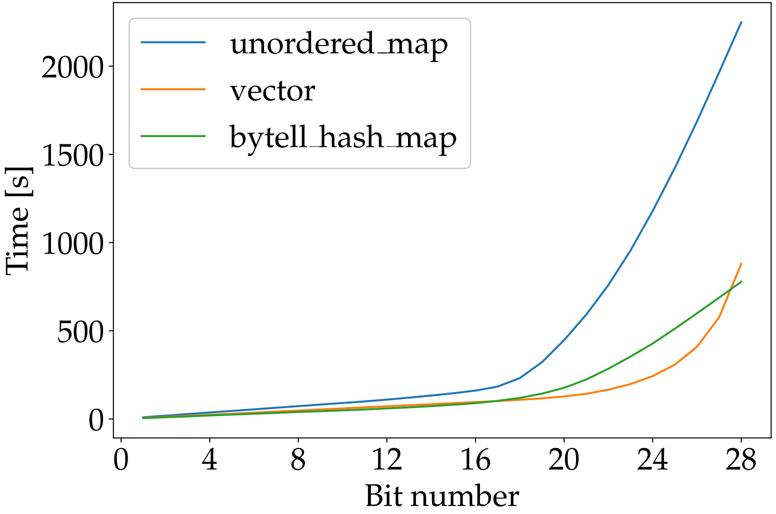
Choice of container:

- std::unordered_map: Memory-efficient but slow
- std::vector: Fast (at first) but memory-inefficient
- ska::bytell_hash_map [5]: Good compromise

Improvements

HashBoW: Container Performance





Improvements

DBoW

Descriptors:

- Change cv::Mat to std::bitset / Eigen::Matrix
- Faster in mean and distance calculation

Bag-of-words vectors:

- Change std::map to std::unordered_map
- Constant instead of logarithmic complexity (search & insert)

Improvements

DBoW

Inverted index:

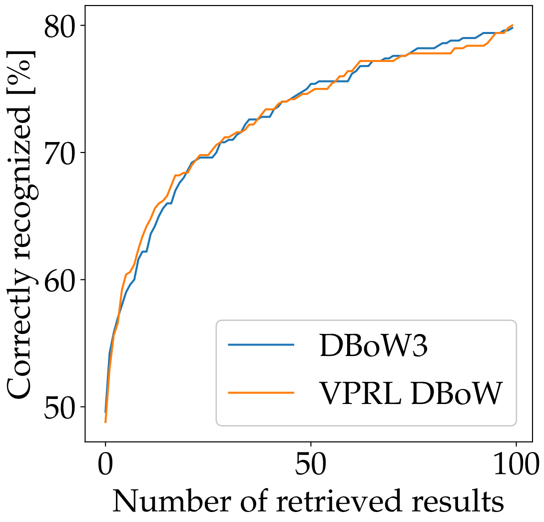
- Change std::vector<std::list> to std::unordered_map<std::vector>
- Improves run-time and memory efficiency

Additional improvements:

- More modern C++
- Improved documentation
- Small changes to further improve run time

Improvements

DBoW: Accuracy and run time



Method	Train	Add	Query
DBoW3	74 m 12 s	14.22 s	10.42 s
VPRL DBoW (ours)	14 m 41 s	3.12 s	1.98 s

Conclusion

Contributions & Future Work

Main Contributions

1. Overview and evaluation of efficient techniques for visual place recognition
2. Novel hashing-based bag-of-words approach
3. Benchmarking suite which is easy to use and extend
4. Efficient and well-documented library for bag-of-words methods

Future Work

Benchmarking Suite:

- More datasets, place recognition methods, evaluation metrics
- Different pipelines, e.g. loop closure detection

Library:

- More methods & database implementations
- DBoW: Direct Index

HashBoW:

- Performance improvements: Different hashing function, locality-preserving hashing
- Extension to real-valued descriptors

Check out the code

Benchmarking Suite:
https://gitlab.vision.in.tum.de/vpr/vpr_benchmark

VPR Library:
https://gitlab.vision.in.tum.de/vpr/vpr_library

Pretrained vocabularies & full evaluation data:
https://gitlab.vision.in.tum.de/vpr/vpr_data

Thank you for your attention.

Tim Stricker

tim.stricker@tum.de

Literature

[1] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2198–2204.

[2] M. Cummins and P. Newman, "Probabilistic appearance based navigation and loop closing," in Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 2042–2048

[3] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

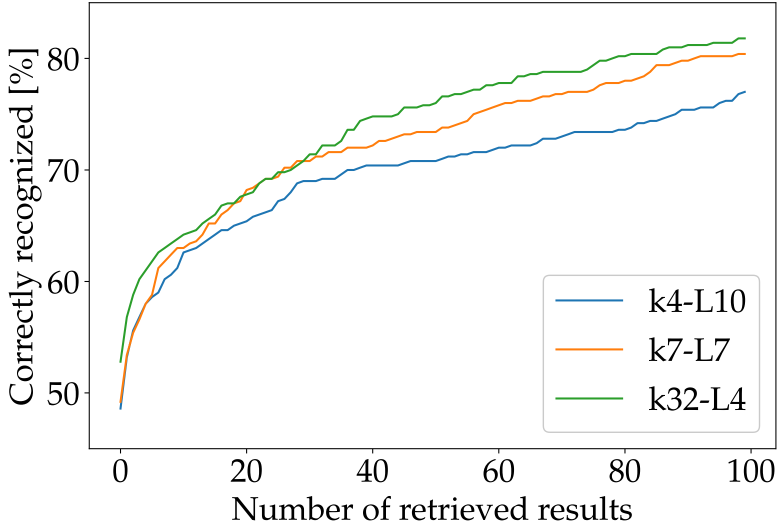
[4] D. Schlegel and G. Grisetti, "HBST: A hamming distance embedding binary search tree for feature-based visual place recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3741–3748, 2018 .

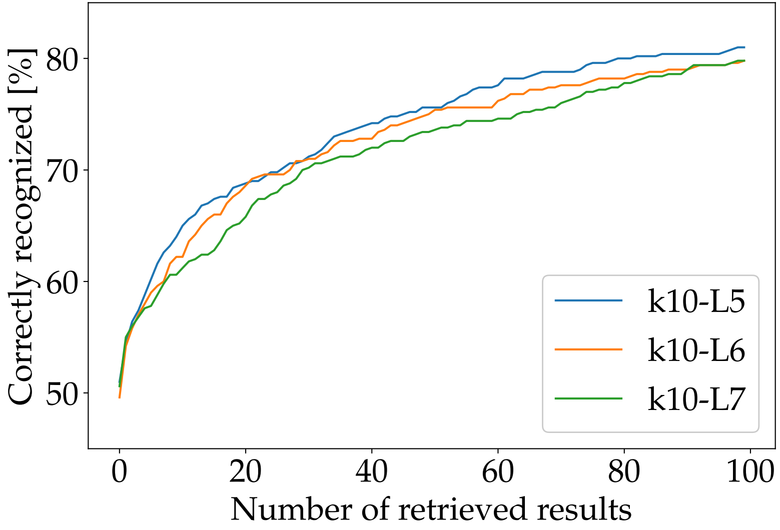
[5] M. Skarupke, *A new fast hash table in response to google's new fast hash table | probably dance*, May 28, 2018. [Online]. Available: https://probablydance.com/2018/05/28/a-new-fast-hash-table-in-response-to-googles-new-fasthash-table/ (visited on 2020-06-22).

Bonus Slides

Evaluation

Parameter Analysis: Vocabulary Tree Size / Structure (DBoW)





Evaluation

Parameter Analysis: Vocabulary Tree Size / Structure (DBoW)

Parameters	Training time
k = 4, L = 10	74 m 23 s
k = 7, L = 7	65 m 35 s
k = 32, L = 4	83 m 6 s

Parameters	Training time
k = 10, L = 5	64 m 41 s
k = 10, L = 6	74 m 12 s
k = 10, L = 7	78 m 35 s

Evaluation

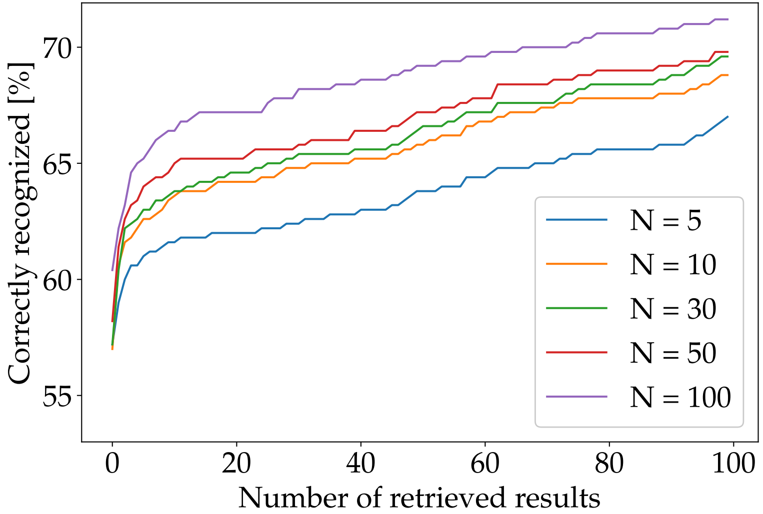
Parameter Analysis: Vocabulary Tree Size / Structure (DBoW)

Parameters	Add	Query
k = 4, L = 10	12.88 s	10.40 s
k = 7, L = 7	13.04 s	10.39 s
k = 32, L = 4	20.88 s	13.35 s

Parameters	Add	Query
k = 4, L = 10	9.95 s	19.00 s
k = 7, L = 7	14.33 s	10.42 s
k = 32, L = 4	17.74 s	10.80 s

Evaluation

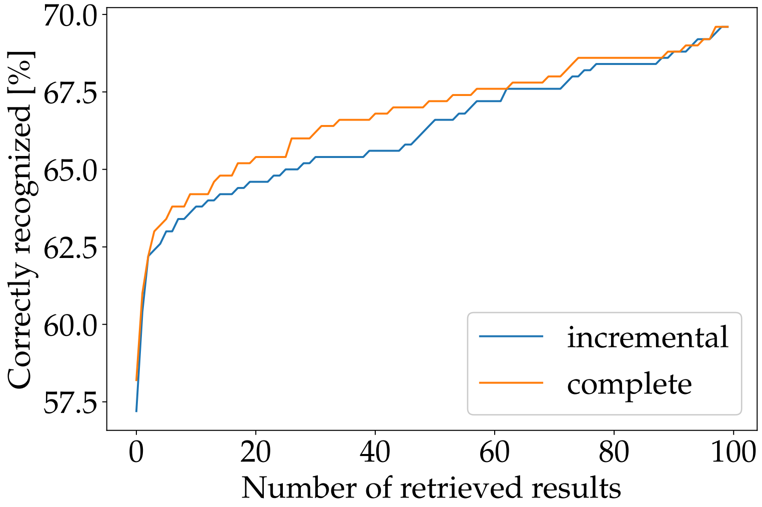
Parameter Analysis: Maximum Leaf Size (HBST)



Max. Leaf Size	Add	Query
5	9.69 s	2.10 s
10	4.58 s	2.25 s
30	5.15 s	2.71 s
50	6.20 s	3.22 s
100	8.85 s	4.61 s

Evaluation

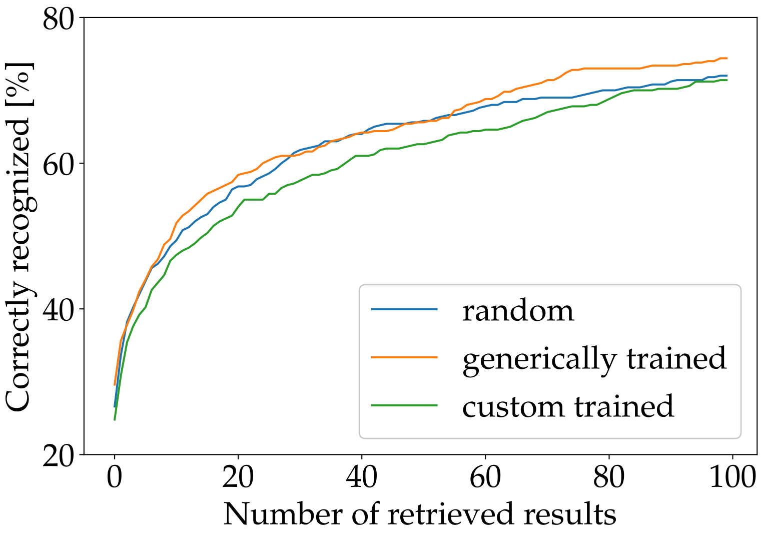
Tree Construction Strategy (HBST)

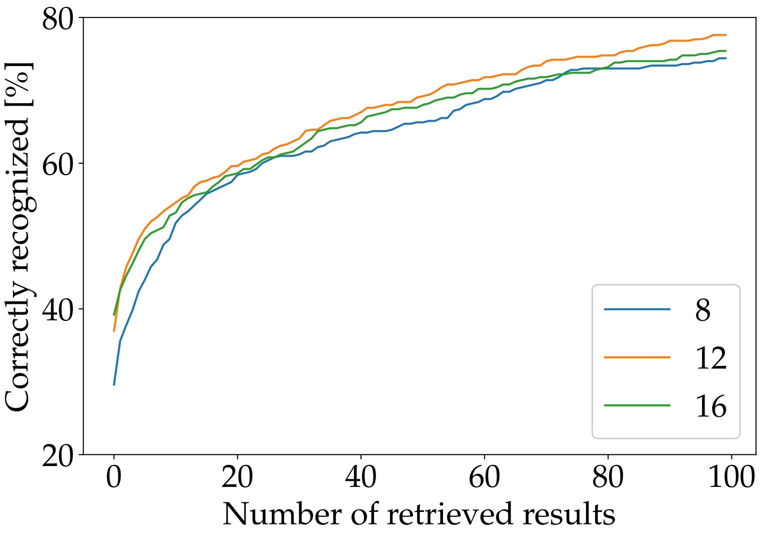


Construction	Add	Train
Incremental	5.17 s	-
Complete	0.02 s	103.27 s

Evaluation

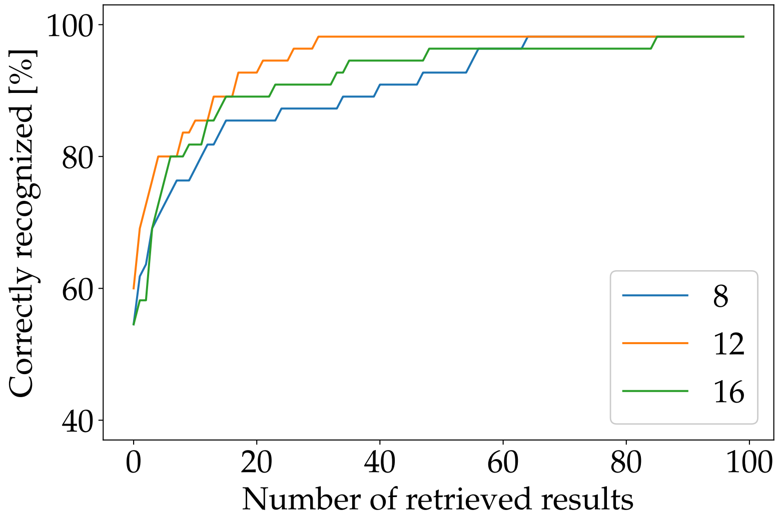
Training: HashBoW (Holidays)

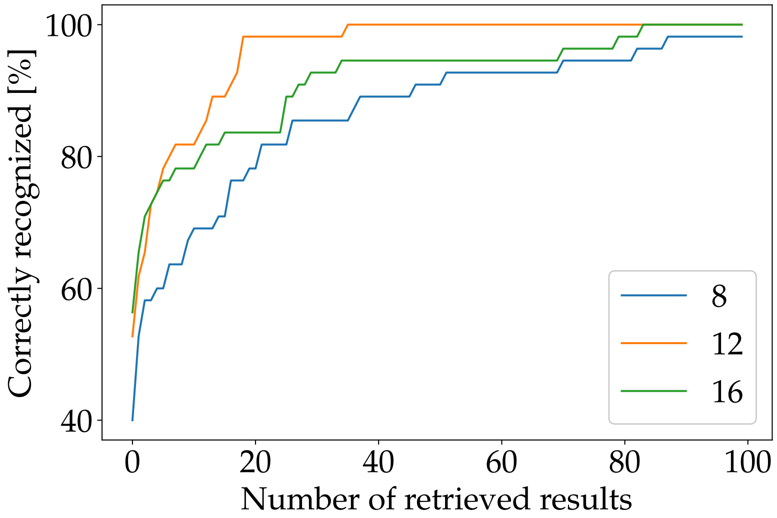




Evaluation

Training: HashBoW (Paris)





Evaluation

Influence of Feature Extractor: HashBoW