Packing bag of features

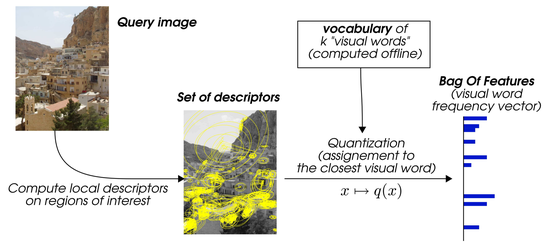
Hervé Jégou, Matthijs Douze, Cordelia Schmid
LEAR, INRIA/LJK, Grenoble, France

Context: large scale image search

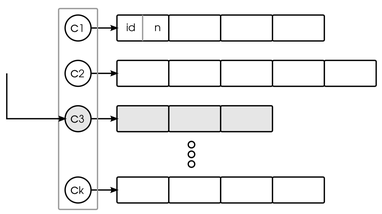
Same-scene search


Constraints:
✔ web-scale : index ~10-500 M images on a single server machine
✔ search in interactive time

Starting point: bag of features (BOF)



✔ TF-IDF weighting of histogram bins, L2-normalized
✔ comparison & ordering of results with L2 distance
✔ BOF is sparse: entries coded in an inverted file



Existing methods: compress inverted file
✔ binary BOF: discard number of occurrences
✔ min-hash: approximate histogram supports
✔ index compression: for each inverted list, RLE + arithmetic coder for image identifiers

MiniBOF

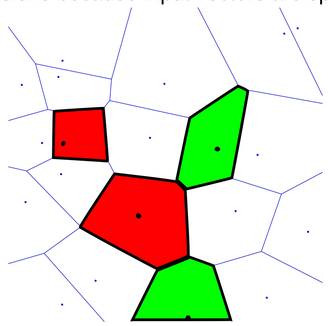
Problems with BOF vectors:
✔ high-dimensional
✔ sparse - there is no efficient approximate nearest neighbor method for sparse vectors

Our solution: transform BOF vectors into dense and shorter vectors = miniBOFs

Which transformation? A simple one: aggregator

$$A_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

✔ arbitrary choice of components to aggregate
✔ all aggregated components have same number of source components
✔ few collisions because input vectors are sparse



Search with L2 distance on Holidays (1000 visual words)
with BOF: mAP=0.403
with MiniBOF: mAP=0.319

→ a single MiniBOF is not enough!

Define **several** aggregators, with different random combinations

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Indexing MiniBOF

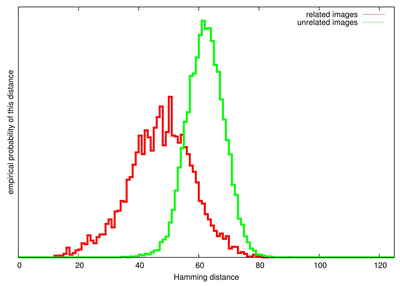
Why quantization:
✔ smaller memory footprint
✔ scan only fraction of dataset (via an inverted file)

2-level quantization and matching:
✔ nearest-neighbor quantizer trained with k-means.
Multiple assignment for the query descriptors

→ images can match if they share a quantization index

✔ binary signature computed with Hamming Embedding

→ matching distance is Hamming distance between signatures



MiniBOF combination

Combination = sum up Hamming distances

db image	15036	72221	14394	53058
MiniBOF 0	0	57	45	?
MiniBOF 1	0	48	53	50
MiniBOF 2	0	50	?	?
total	0	155	?	?

Missing distance measures: fill in with **Hamming distance expectation** = half of binary signature length

Results

✔ Kentucky object recognition benchmark: 2550 groups of 4 images that represent the same object

method	score	memory usage	image hits
BOF	2.92	6,662	9,928
compressed bin. BOF	3.02	946*	9,928
1 MiniBOF	2.07	20	94
8 MiniBOFs	2.72	160	383
64 MiniBOFs	2.93	1,280	1,078

* estimated memory usage

✔ INRIA Holidays dataset + 1 million Flickr images: seach 500 images of a scene among distractors

method	mAP	memory usage	query time
state-of-art	0.609	35,844	22656
BOF	0.315	8,885	2827
binary BOF	0.381	7,108	2562
1 MiniBOF	0.066	20	74
8 MiniBOFs	0.196	160	132
32 MiniBOFs	0.244	640	352

Example results: