Overview

- Local invariant features (C. Schmid)
- Matching and recognition with local features (J. Sivic)
- Efficient visual search (J. Sivic)
- **Very large scale indexing** (C. Schmid)
- Practical session (J. Sivic)

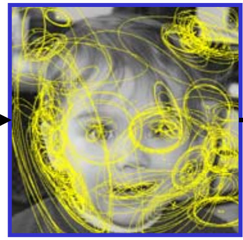
State-of-the-art: Bag-of-words [Sivic & Zisserman'03]

Query image

Hessian-Affine regions + SIFT descriptors

[Mikolajezyk & Schmid 04]
[Lowe 04]

Set of SIFT descriptors

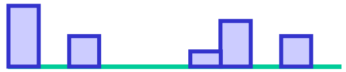


centroids (visual words)

Bag-of-features processing +*tf-idf* weighting

[Nister & al 04, Chum & al 07]

sparse frequency vector



Inverted file

querying

ranked image short-list

Geometric verification

[Lowe 04, Chum & al 2007]

Re-ranked list

Two issues:

- Matching approximation by visual words
- Still limited number of images

Bag-of-features as an ANN search algorithm

- Matching function of descriptors : *k*-nearest neighbors

$$f_{k\text{-NN}}(x,y) = \begin{cases} 1 & \text{if } x \text{ is a } k\text{-NN of } y \\ 0 & \text{otherwise} \end{cases}$$

- Bag-of-features matching function $f_q(x,y) = \delta_{q(x),q(y)}$

where q(x) is a quantizer, i.e., assignment to visual word and
$\delta_{a,b}$ is the Kronecker operator ($\delta_{a,b}$=1 iff a=b)

Approximate nearest neighbor search evaluation

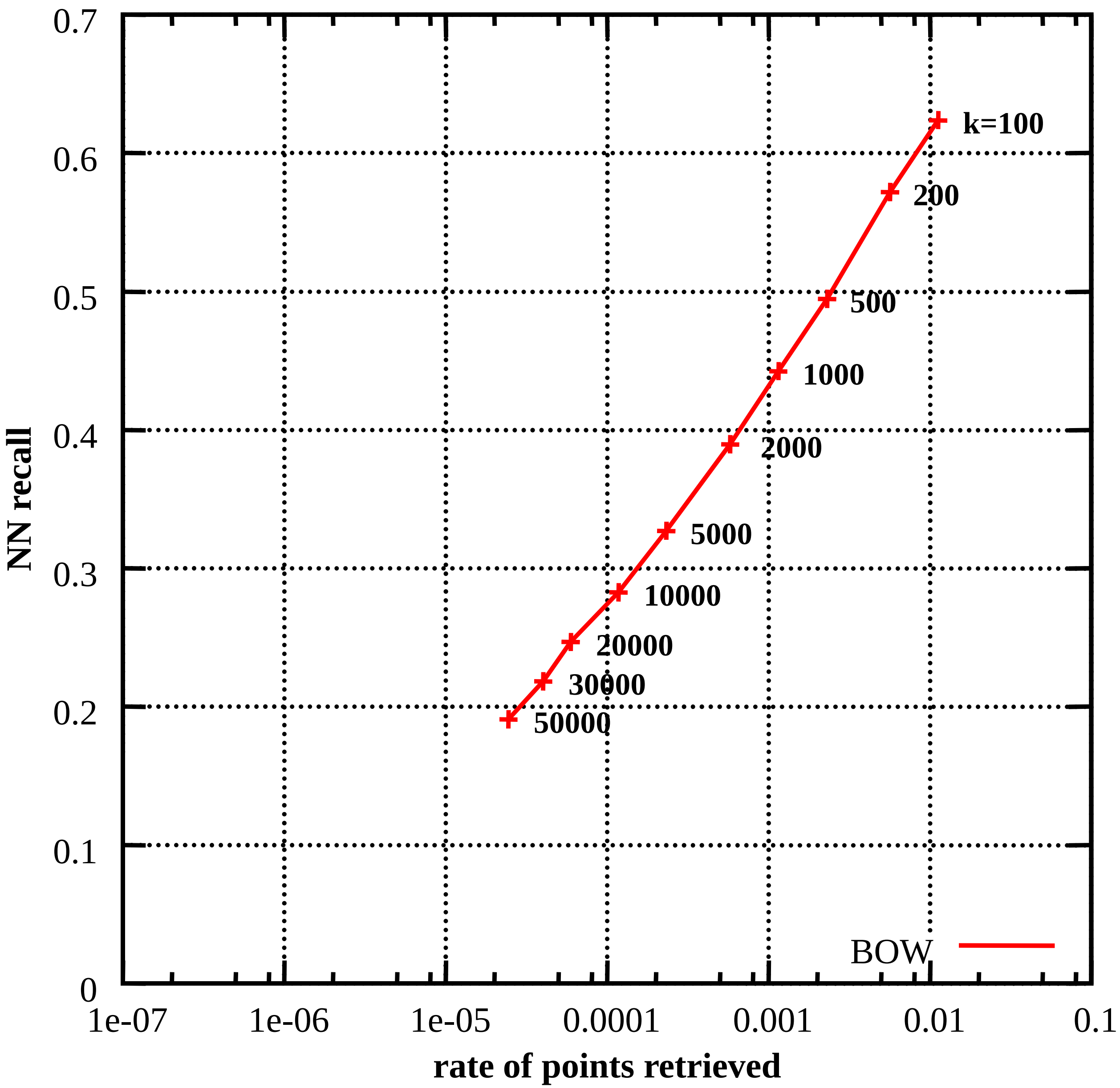
- ANN algorithms usually returns a short-list of nearest neighbors
  - this short-list is supposed to contain the NN with high probability
  - exact search may be performed to re-order this short-list

- Proposed quality evaluation of ANN search: trade-off between
  - **Accuracy**: **NN recall** = probability that *the* NN is in this list

    against

  - **Ambiguity removal** = proportion of vectors in the short-list
    - the lower this proportion, the more information we have about the vector
    - the lower this proportion, the lower the complexity if we perform exact search on the short-list
- ANN search algorithms usually have some parameters to handle this trade-off

ANN evaluation of bag-of-features



ANN algorithms returns a list of potential neighbors

Accuracy: **NN recall**
= probability that *the* NN is in this list

Ambiguity removal:
= proportion of vectors in the short-list

In BOF, this trade-off is managed by the number of clusters *k*

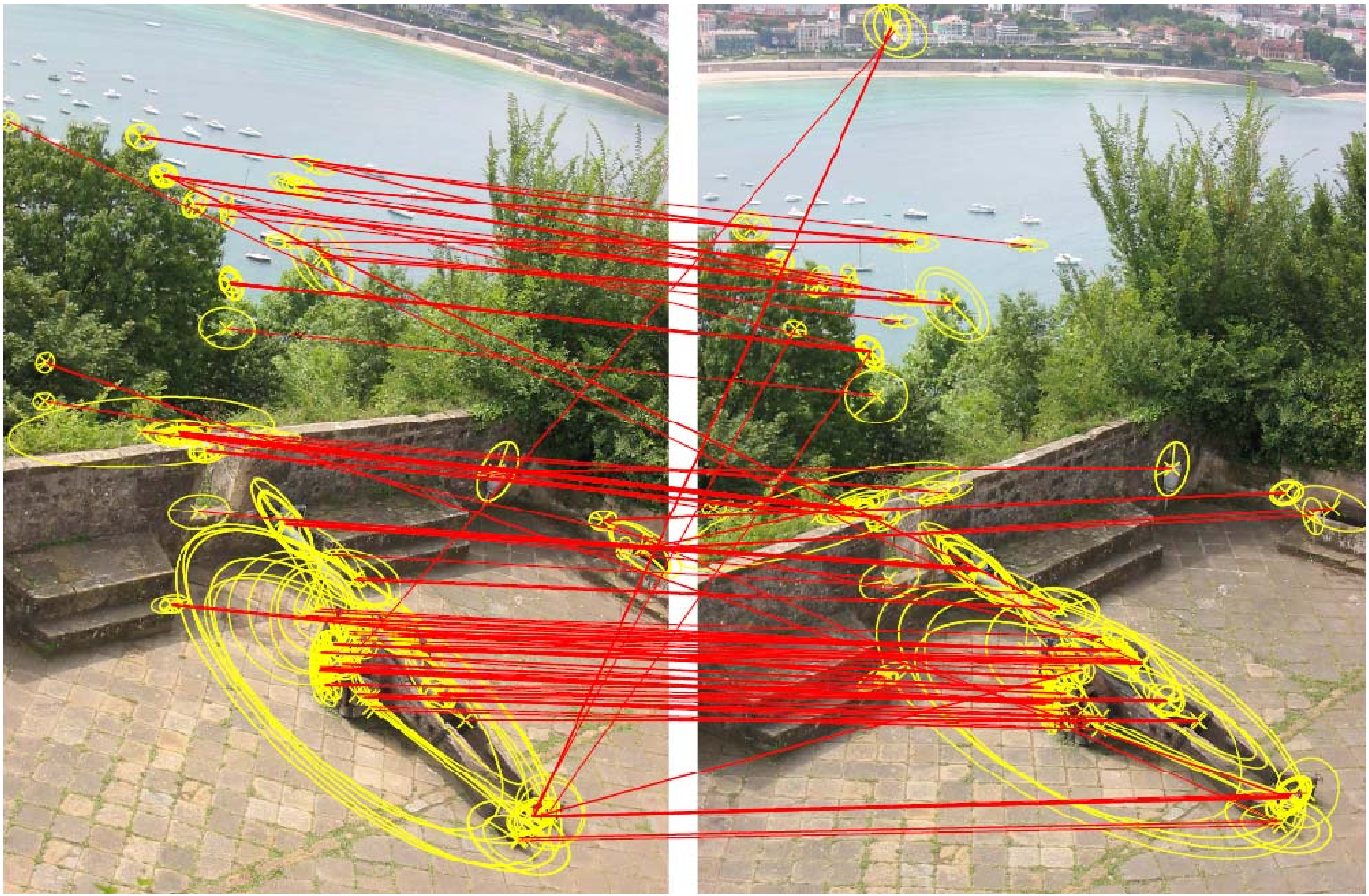
Problem with bag-of-features

- The intrinsic matching scheme performed by BOF is weak
  - for a “small” visual dictionary: too many false matches
  - for a “large” visual dictionary: many true matches are missed

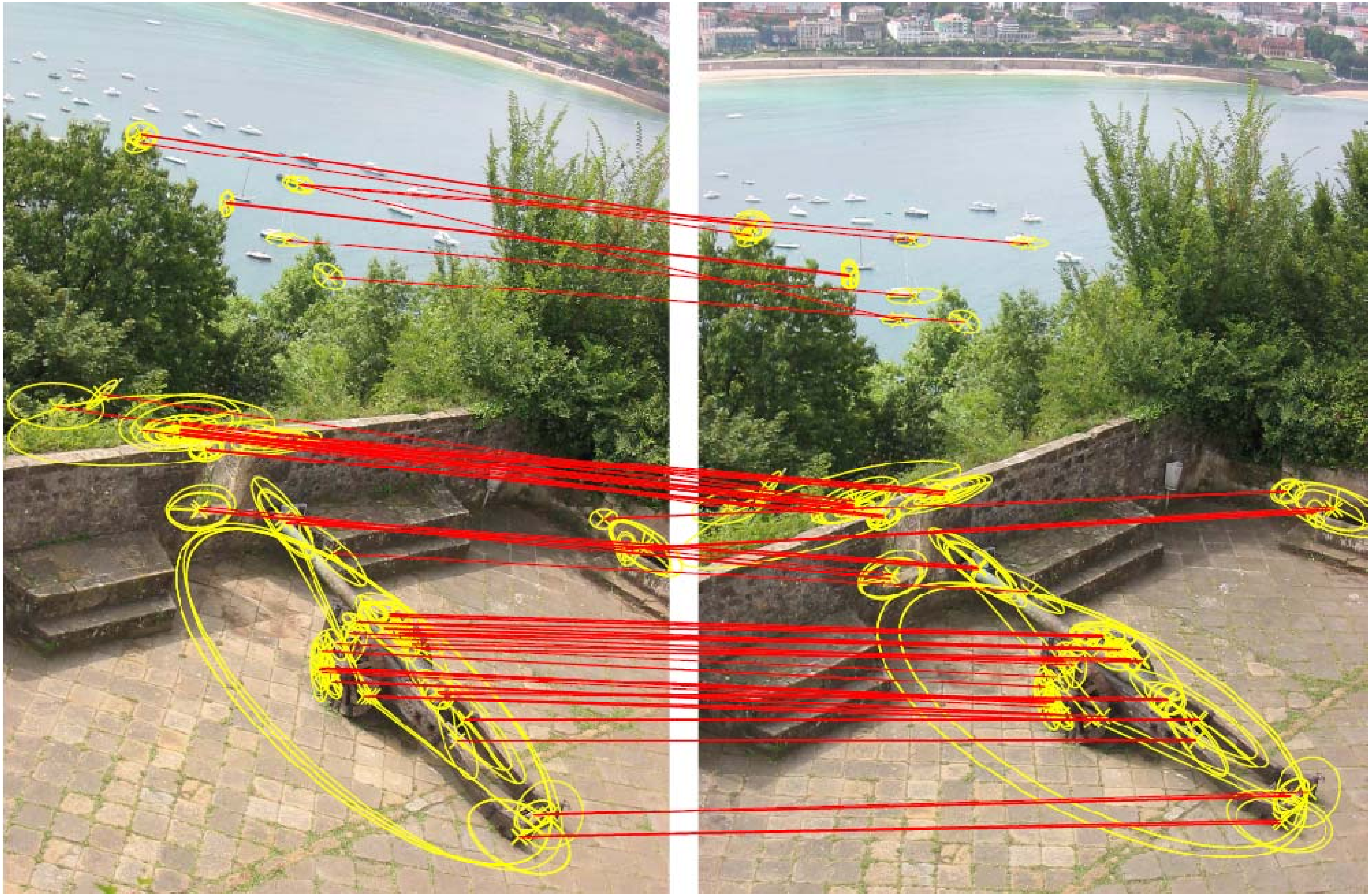
- No good trade-off between “small” and “large” !
  - either the Voronoi cells are too big
  - or these cells can’t absorb the descriptor noise

  $\rightarrow$ intrinsic approximate nearest neighbor search of BOF is not sufficient

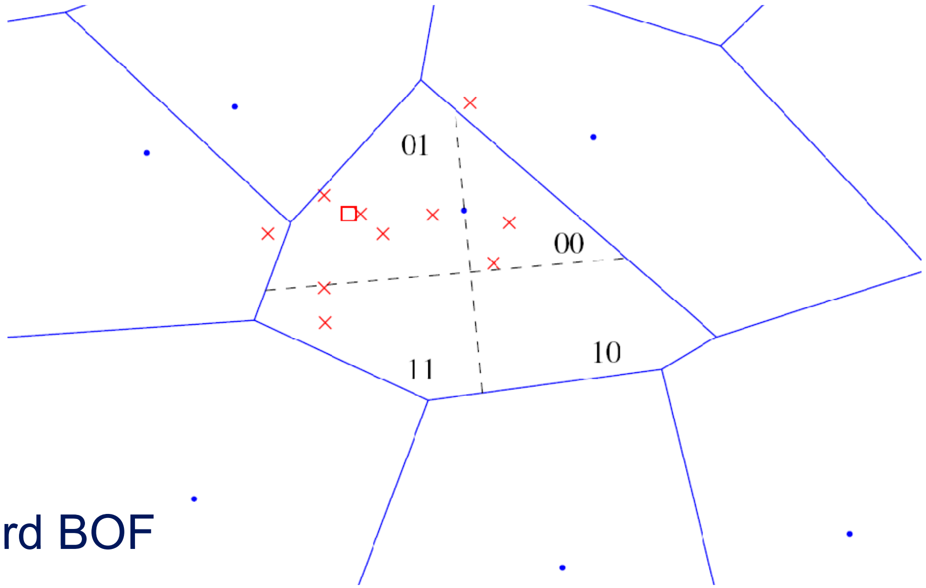
20K visual word: false matchs



200K visual word: good matches missed



Hamming Embedding



- Representation of a descriptor *x*
  - Vector-quantized to *q(x)* as in standard BOF
  - \+ short binary vector *b(x)* for an additional localization in the Voronoi cell

- Two descriptors x and y match iif

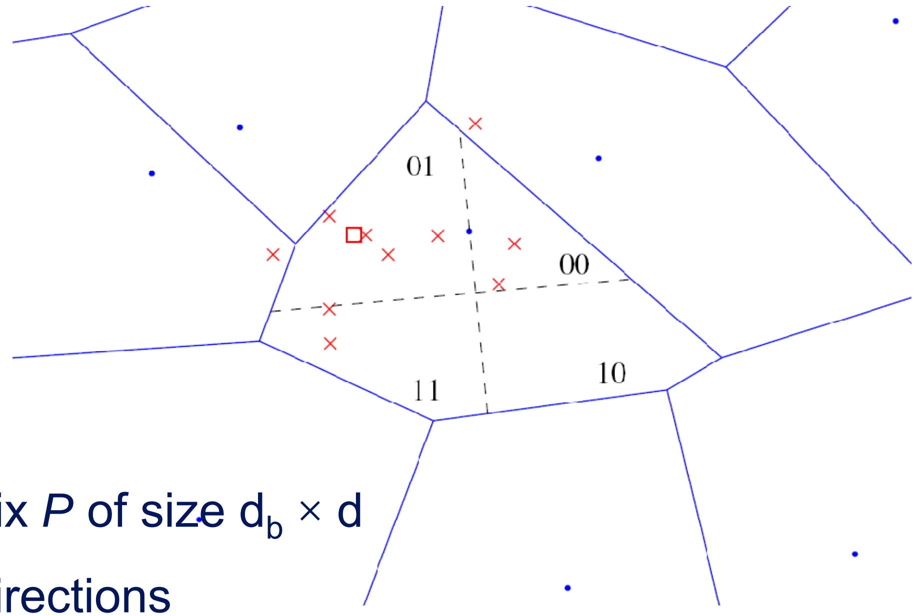
$$q(x) = q(y) \text{ and } h\left(b(x), b(y)\right) \leq h_t$$

where h(*a*,*b*) is the Hamming distance

- Nearest neighbors for Hammg distance ≈ the ones for Euclidean distance

- Efficiency
  - Hamming distance = very few operations
  - Fewer random memory accesses: 3 faster that BOF with same dictionary size!

Hamming Embedding



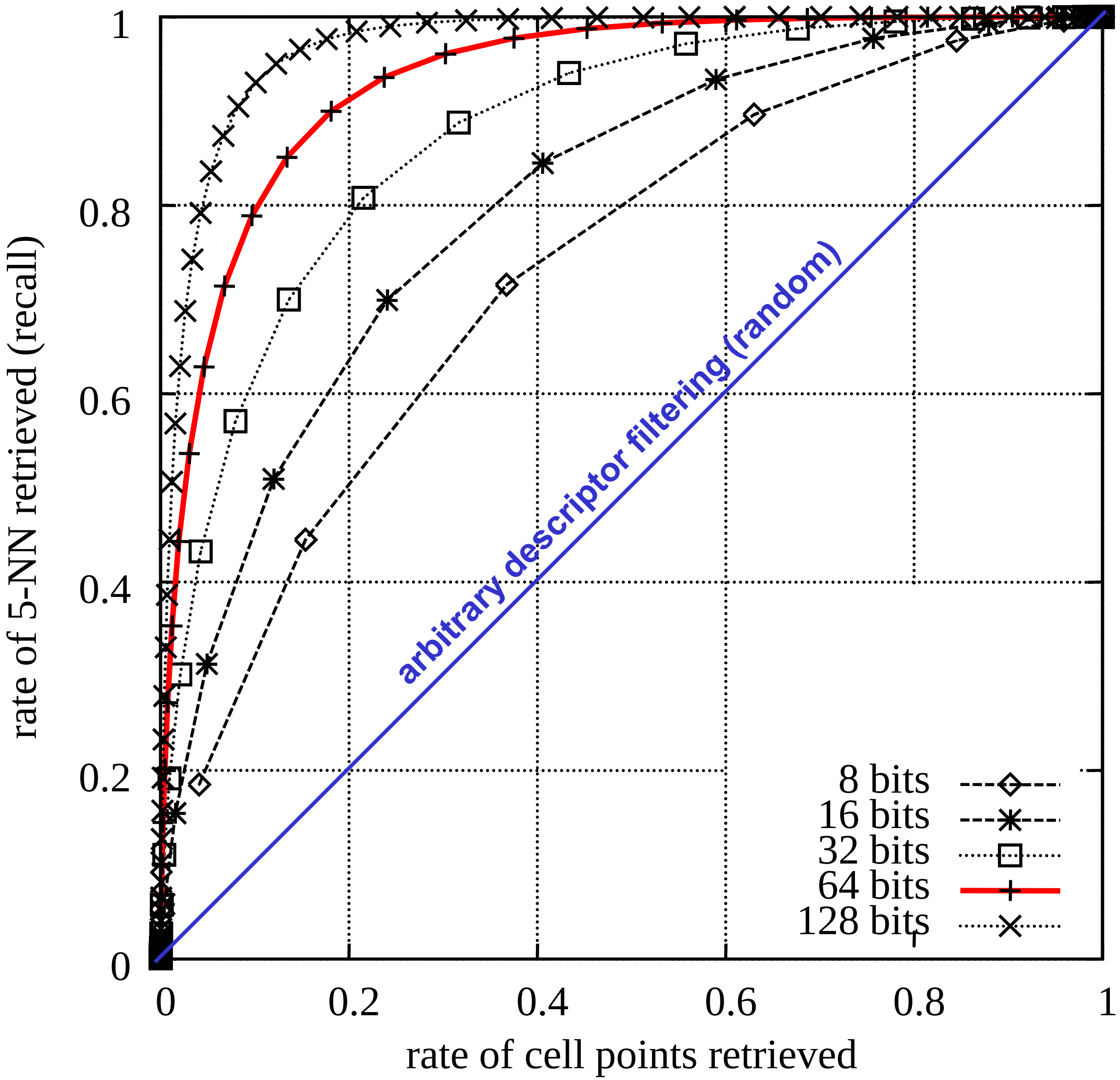
- **Off-line** (given a quantizer)
  - draw an orthogonal projection matrix P of size $d_b \times d$

  $\rightarrow$ this defines d_b random projection directions
  - for each Voronoi cell and projection direction, compute the median value from a learning set

- **On-line**: compute the binary signature b(x) of a given descriptor
  - project x onto the projection directions as $z(x) = (z_1, \ldots z_{db})$
  - $b_i(x) = 1$ if $z_i(x)$ is above the learned median value, otherwise 0

[H. Jegou et al., Improving bag of features for large scale image search, ICJV'10]

Hamming and Euclidean neighborhood

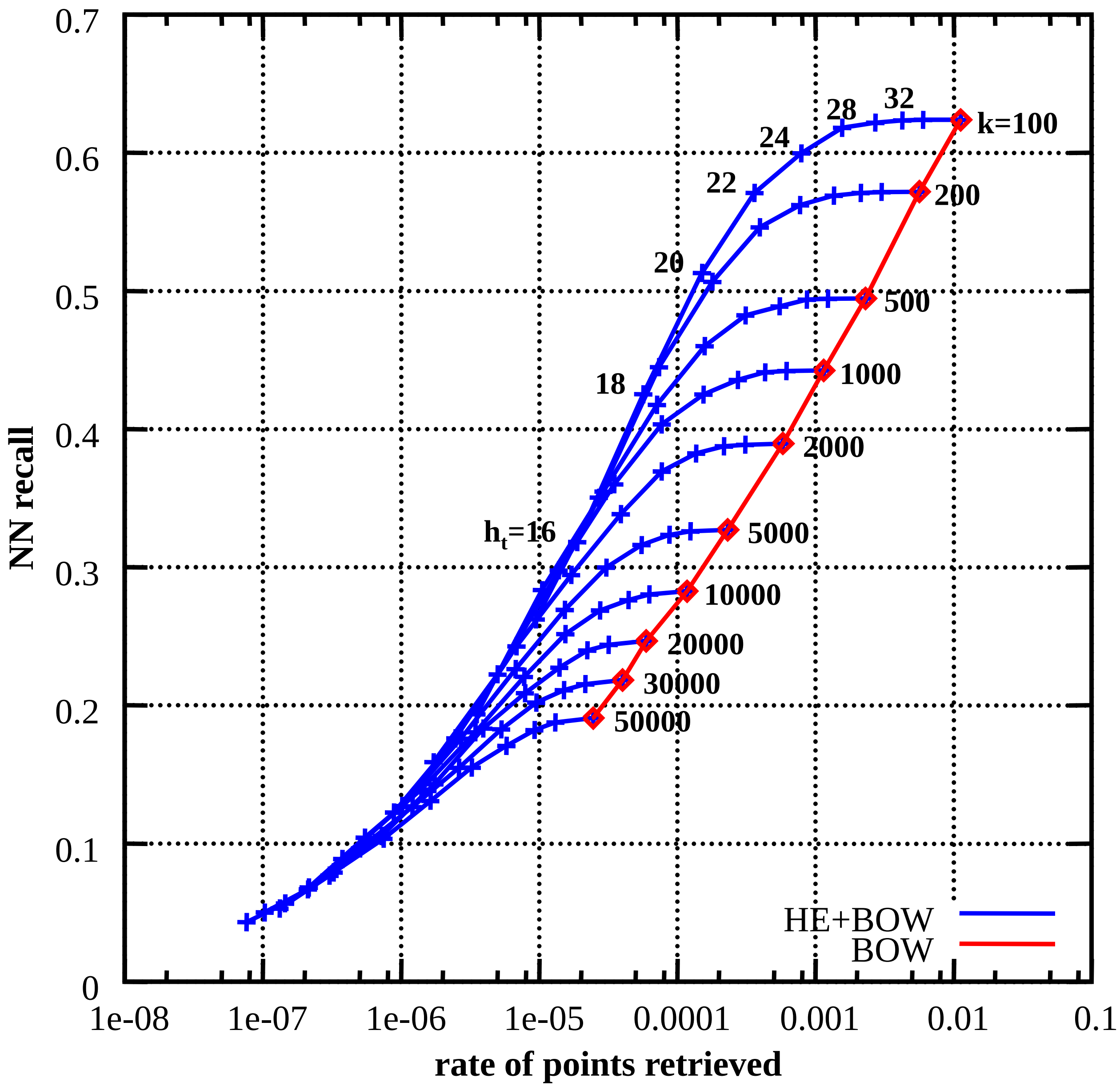


- trade-off between memory usage and accuracy

→ more bits yield higher accuracy

We used 64 bits (8 bytes)

ANN evaluation of Hamming Embedding



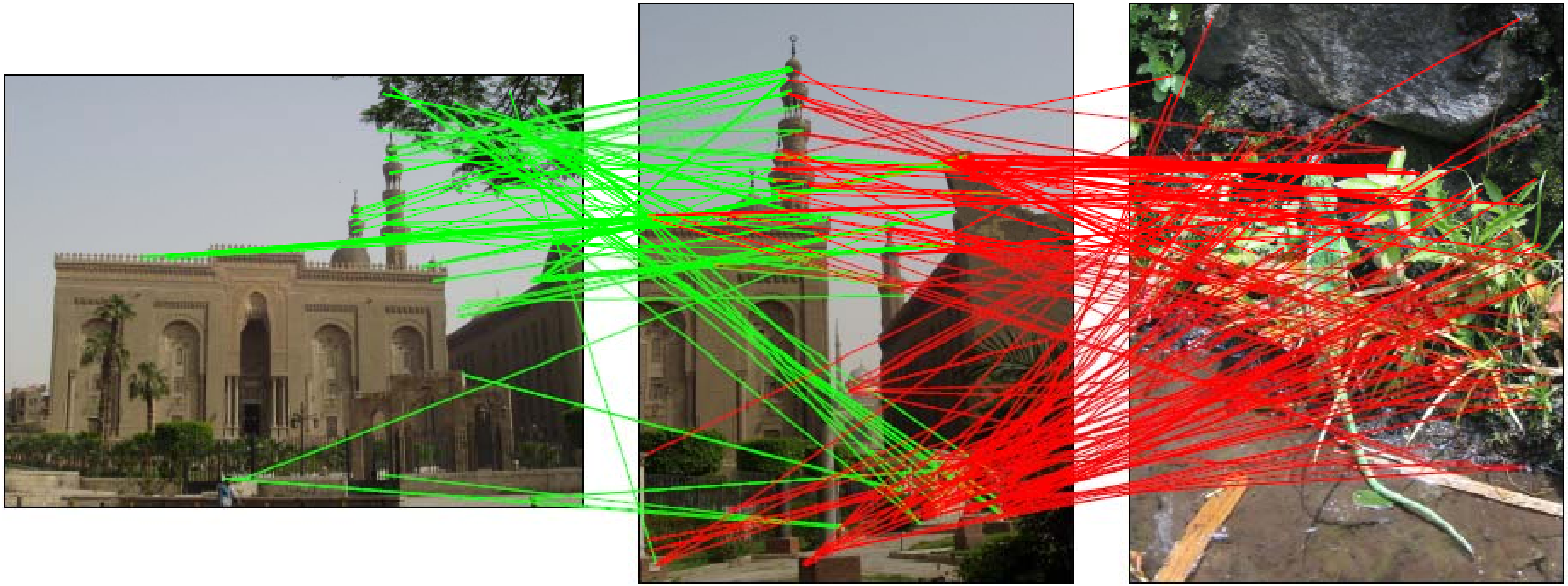
compared to BOW: at least 10 times less points in the short-list for the same level of accuracy

Hamming Embedding provides a much better trade-off between recall and ambiguity removal

Matching points - 20k word vocabulary

201 matches

240 matches

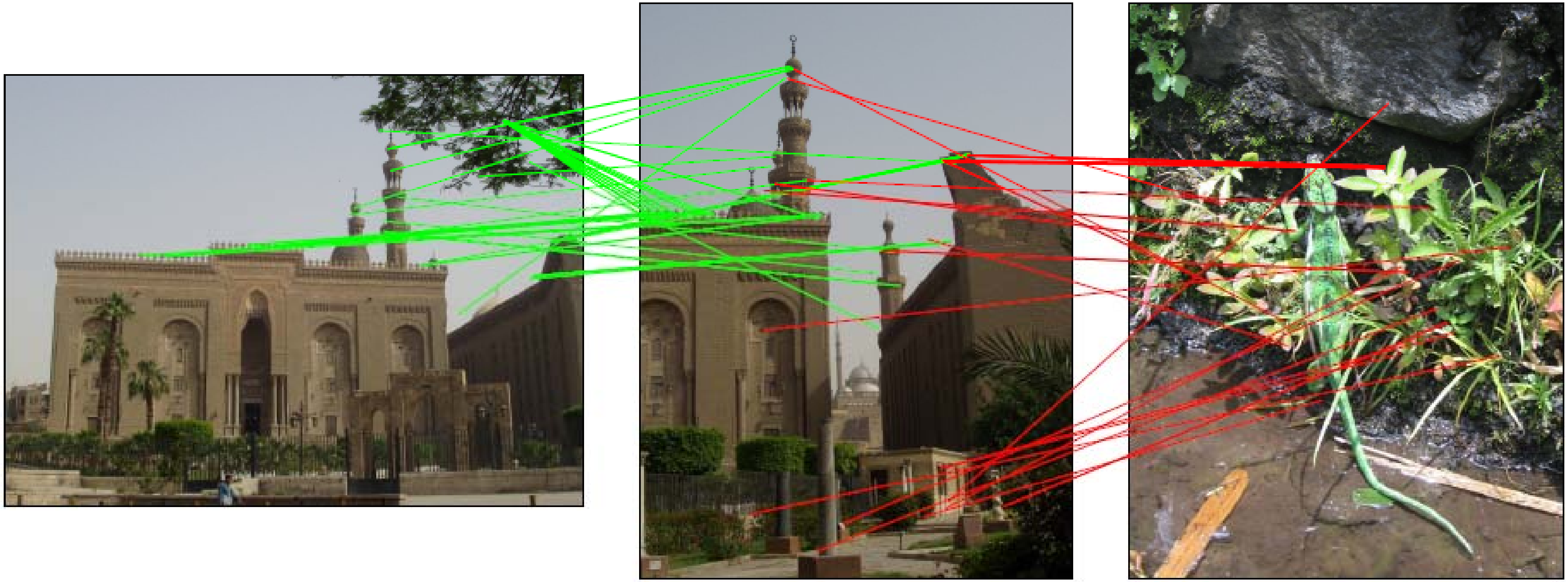


Many matches with the non-corresponding image!

Matching points - 200k word vocabulary

69 matches

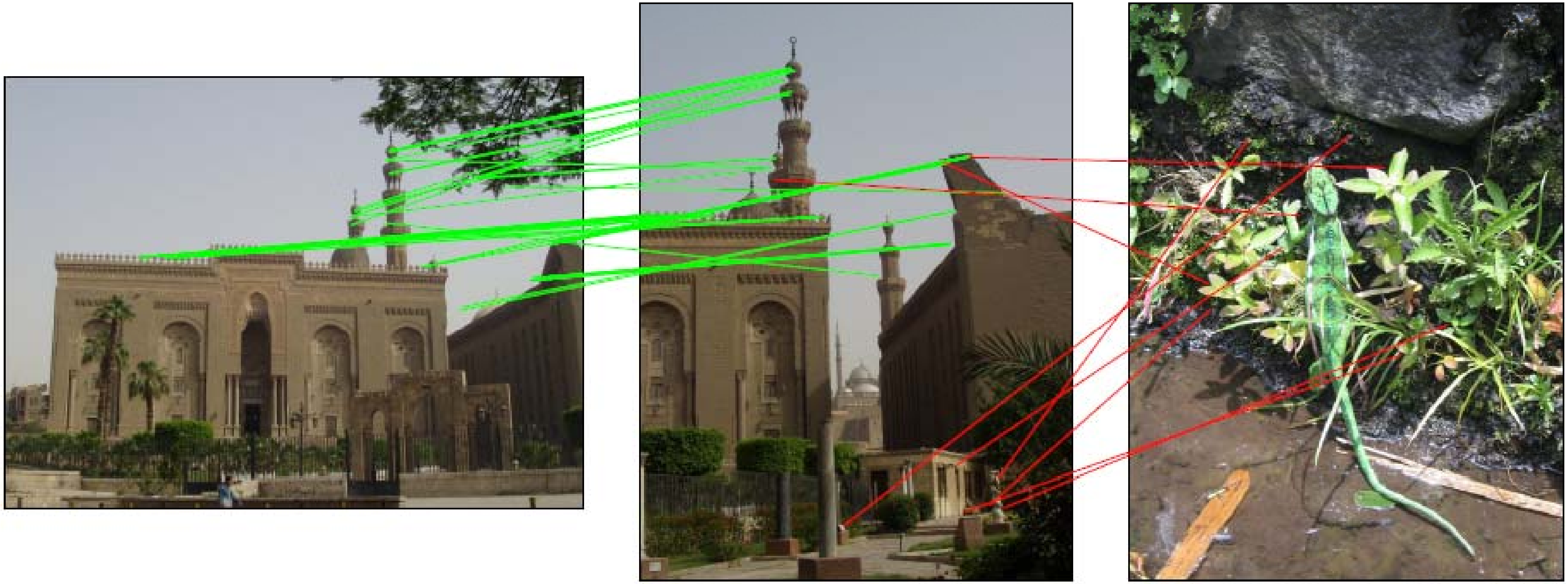
35 matches



Still many matches with the non-corresponding one

Matching points - 20k word vocabulary + HE

83 matches 8 matches



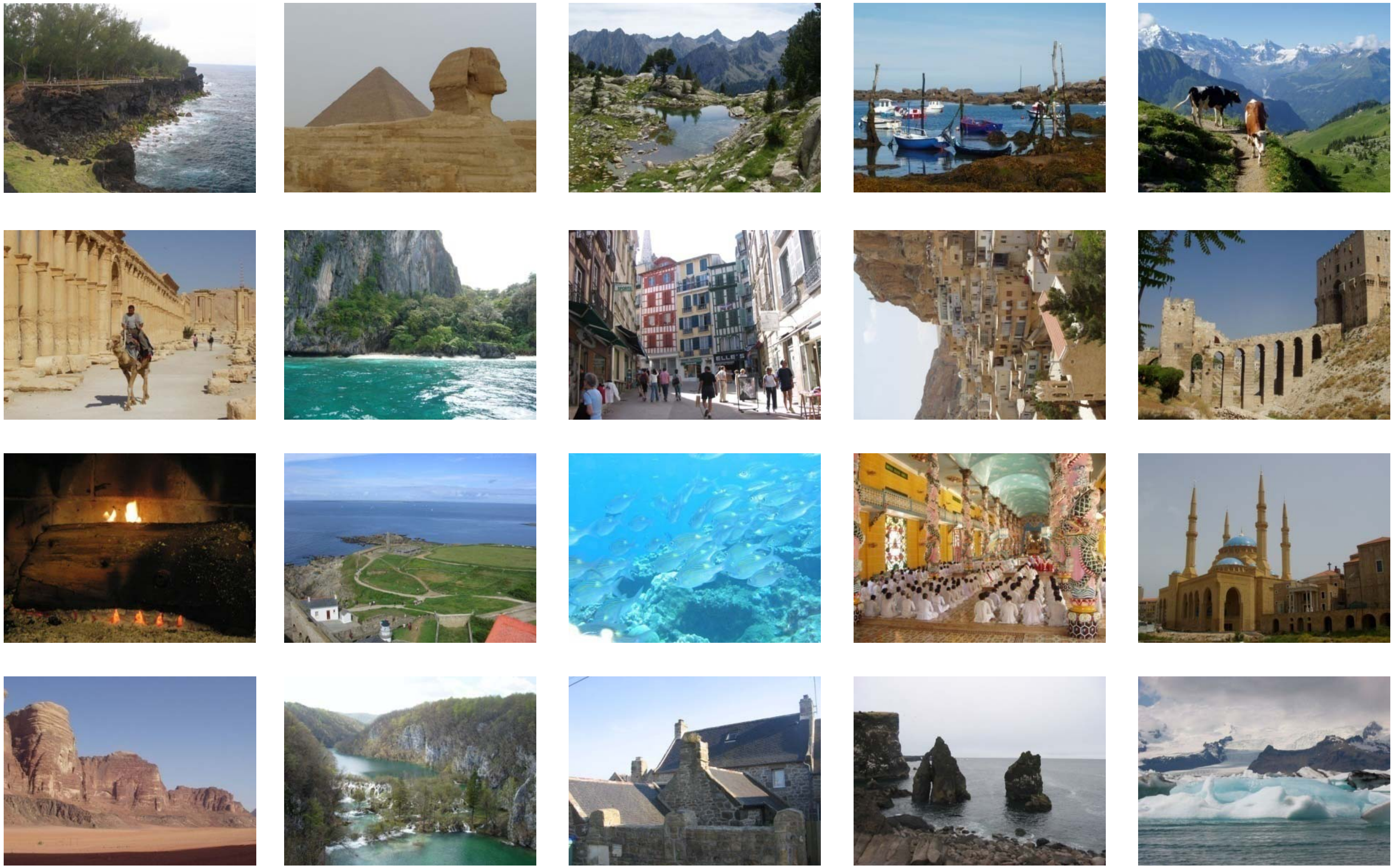
10x more matches with the corresponding image!

Experimental results

- Evaluation for the INRIA holidays dataset, 1491 images
  - 500 query images + 991 annotated true positives
  - Most images are holiday photos of friends and family
- 1 million & 10 million distractor images from Flickr
- Vocabulary construction on a different Flickr set
- Almost real-time search speed

- Evaluation metric: mean average precision (in [0,1], bigger = better)
  - Average over precision/recall curve

Holiday dataset – example queries



Dataset : Venice Channel

Dataset : San Marco square


Query


Base 1


Base 2


Base 3


Base 4


Base 5

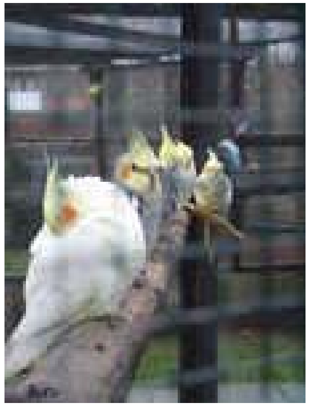

Base 6


Base 7


Base 8

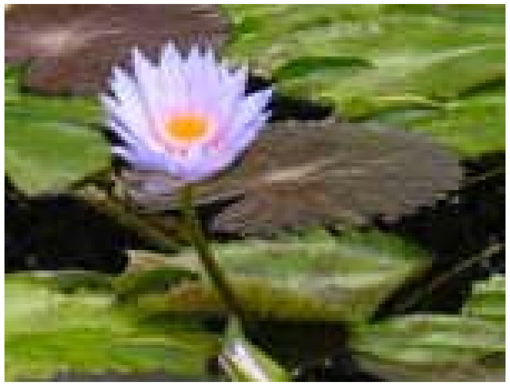

Base 9

Example distractors - Flickr

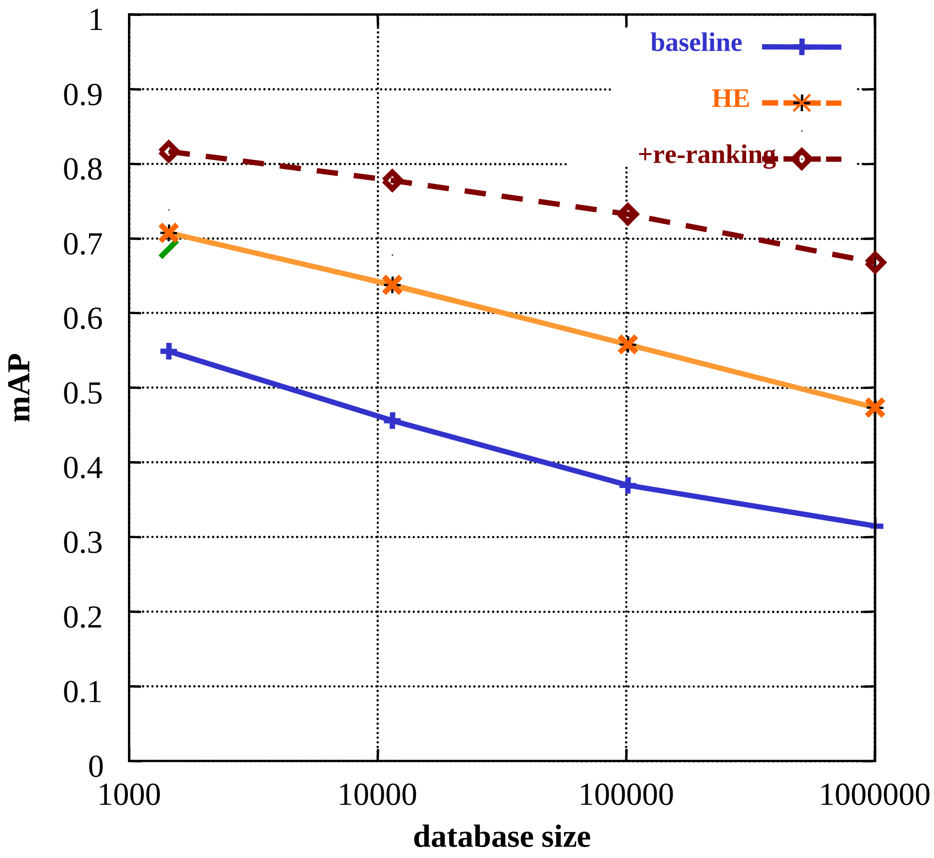





Experimental evaluation

- Evaluation on our holidays dataset, 500 query images, 1 million distracter images
- Metric: mean average precision (in [0,1], bigger = better)



BOF 2
Ours 1

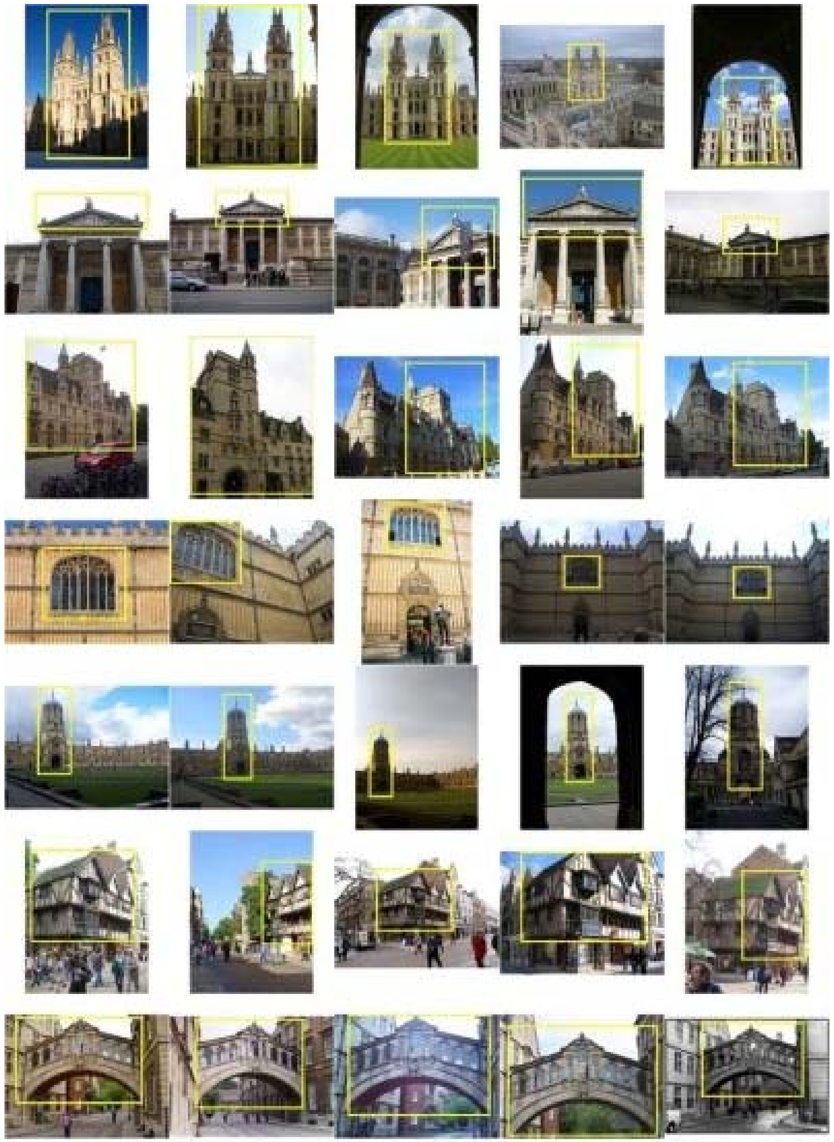
BOF 5890
Ours 4

BOF 43064
Ours 5

Results – Venice Channel

Demo at http://bigimbaz.inrialpes.fr

Comparison with the state of the art: Oxford dataset [Philbin et al. CVPR'07]



Evaluation measure:
Mean average precision (mAP)

Comparison with the state of the art: Kentucky dataset [Nister et al. CVPR'06]

4 images per object

Evaluation measure: among the 4 best retrieval results how many are correct (ranges from 1 to 4)

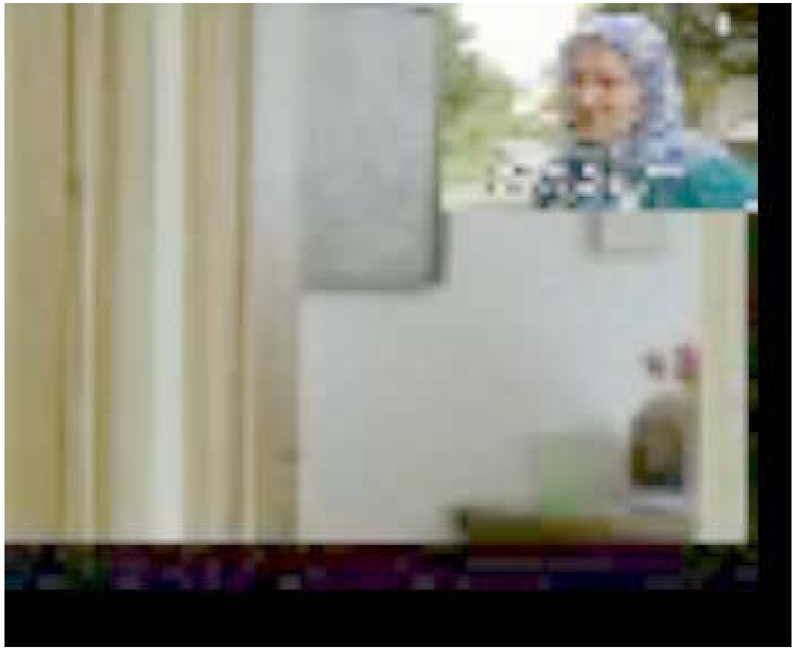
Comparison with the state of the art

dataset	Oxford		Kentucky	
distractors	0	100K	0	1M
soft assignment [14]	0.493	0.343		
ours	0.615	0.516		
soft + geometrical re-ranking [14]	0.598	0.480		
ours + geometrical re-ranking	0.667	0.591		
soft + query expansion [14]	0.718	0.605		
ours + query expansion	0.747	0.687		
hierarchical vocabulary [6]			3.19	
CDM [11]			3.61	2.93
ours			3.42	3.10
ours + geometrical re-ranking			3.55	3.40

[14] Philbin et al., CVPR'08; [6] Nister et al., CVPR'06; [10] Harzallah et al., CVPR'07]

Extension to videos: video copy detection

- Indexing individual sampled frames
- Addition of a spatio-temporal filter
- Excellent results in the TrecVid video copy detection competiton



Towards larger databases?

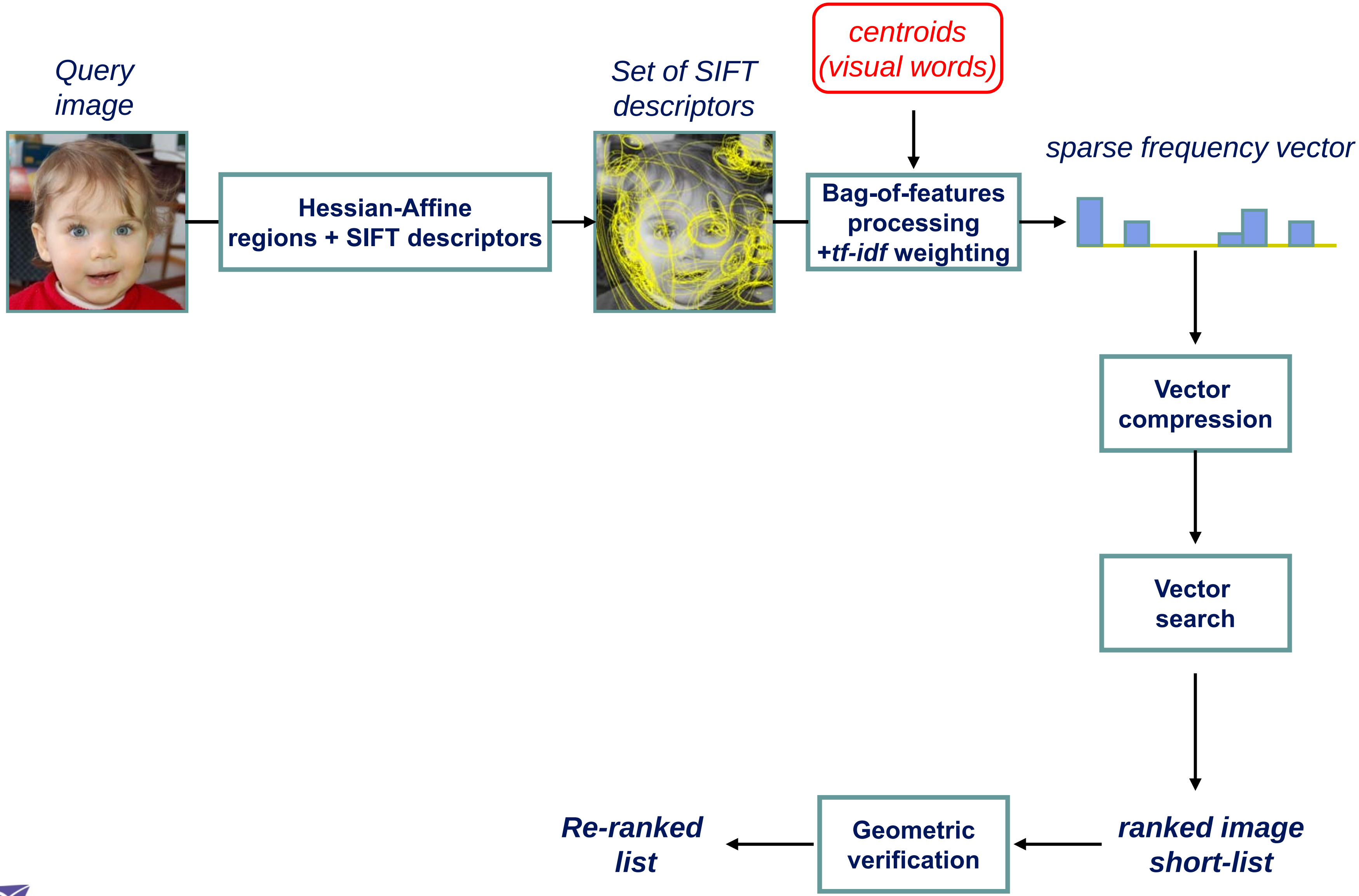
- BOF can handle up to ~10 M d'images
  - with a limited number of descriptors per image
  - 40 GB of RAM
  - search = 2 s

- Web-scale = billions of images
  - With 100 M per machine

    → search = 20 s, RAM = 400 GB

    → not tractable!

Recent approaches for very large scale indexing



Related work on very large scale image search

- Min-hash and geometrical min-hash [Chum et al. 07-09]
- GIST descriptors with Spectral Hashing [Torralba et al.'08]
- Compressing the BoF representation (miniBof) [Jégou et al. 09]
- Aggregating local desc into a compact image representation [Jegou et al. 10]
- Efficient object category recognition using classemes [Torresani et al.'10]

Compact image representation

- Aim: improving the tradeoff between
  - search speed
  - memory usage
  - search quality

- Approach: joint optimization of three stages
  - local descriptor aggregation
  - dimension reduction
  - indexing algorithm

Image representation VLAD → PCA + PQ codes → (Non) – exhaustive search

[H. Jegou et al., Aggregating local desc into a compact image representation, CVPR'10]

Aggregation of local descriptors

- Problem: represent an image by a single fixed-size vector:

  set of n local descriptors → 1 vector

- Most popular idea: BoF representation [Sivic & Zisserman 03]
  - sparse vector
  - highly dimensional

→ high dimensionality reduction introduces loss

- Alternative: Fisher Kernels [Perronnin et al 07]
  - non sparse vector
  - excellent results with a small vector dimensionality

  → our method (VLAD) the spirit of this representation

VLAD : vector of locally aggregated descriptors

- Simplification of Fisher kernels

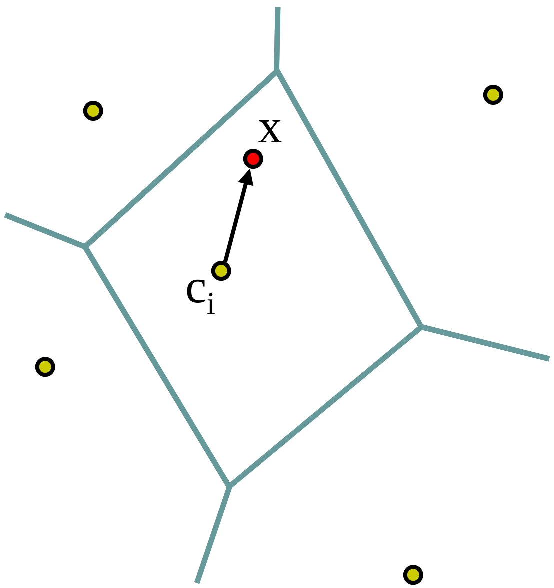
- Learning: a vector quantifier (*k*-means)
  - output: *k* centroids (visual words): $c_1,\ldots,c_i,\ldots c_k$
  - centroid c_i has dimension *d*

- For a given image
  - assign each descriptor to closest center c_i
  - accumulate (sum) descriptors per cell

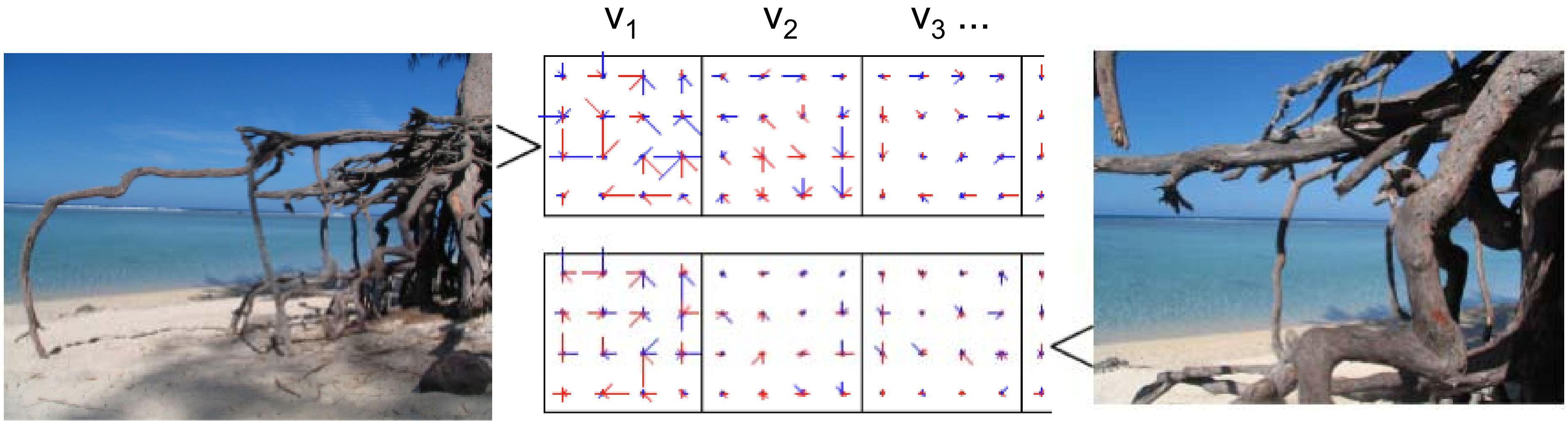
    $v_i := v_i + (x - c_i)$

- VLAD (dimension $D = k \times d$)

- The vector is L2-normalized



VLADs for corresponding images



SIFT-like representation per centroid (+ components: blue, - components: red)

- good coincidence of energy & orientations

VLAD performance and dimensionality reduction

- We compare VLAD descriptors with BoF: INRIA Holidays Dataset (mAP,%)
- Dimension is reduced to from D to D’ dimensions with PCA

Aggregator	**k**	**D**	**D’=D (no reduction)**	**D’=128**	**D’=64**
BoF	1,000	1,000	41.4	44.4	43.4
BoF	20,000	20,000	44.6	45.2	44.5
BoF	200,000	200,000	54.9	43.2	41.6
VLAD	16	2,048	49.6	49.5	**49.4**
VLAD	64	8,192	52.6	**51.0**	47.7
VLAD	256	32,768	**57.5**	50.8	47.6

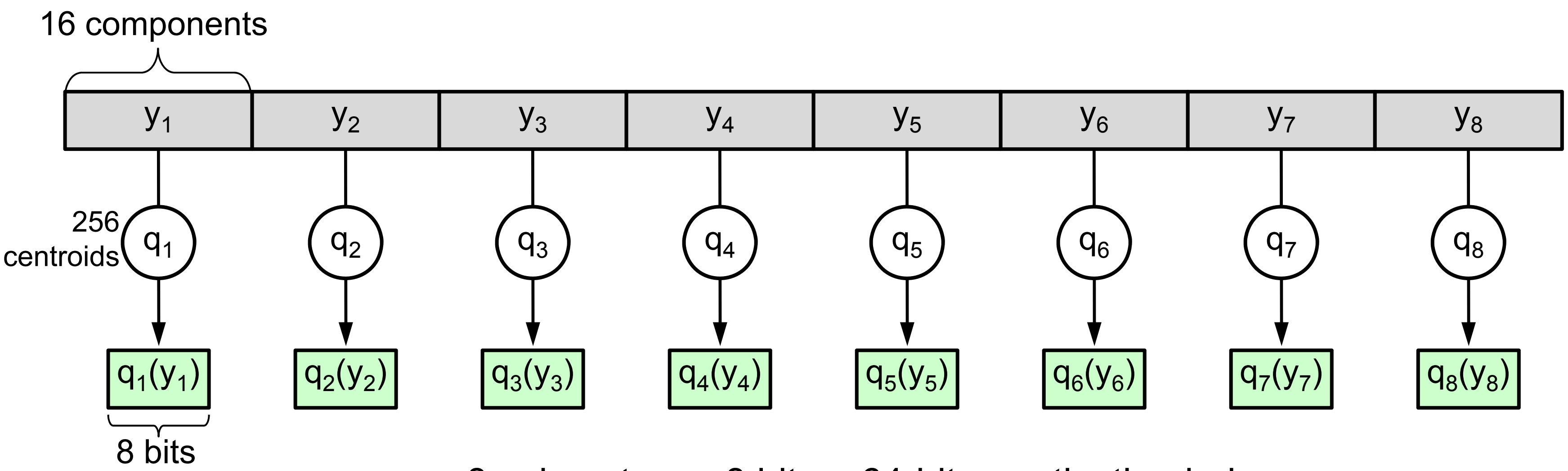
- Observations:
  - VLAD better than BoF for a given descriptor size
    → comparable to Fisher kernels for these operating points
  - Choose a small D if output dimension D’ is small

Product quantization for nearest neighbor search

- Vector split into *m* subvectors: $y \rightarrow [y_1 | \ldots | y_m]$

- Subvectors are quantized separately by quantizers $q(y) = [q_1(y_1) | \ldots | q_m(y_m)]$
  where each q_i is learned by *k*-means with a limited number of centroids

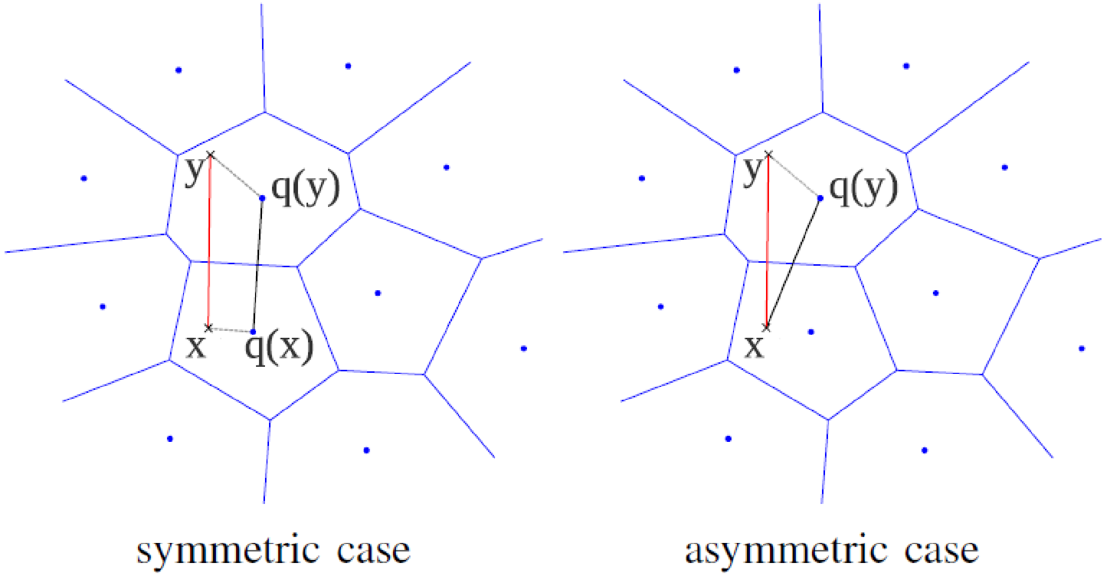
- Example: y = 128-dim vector split in 8 subvectors of dimension 16
  - each subvector is quantized with 256 centroids -> 8 bit
  - very large codebook 256^8 ~ 1.8x10^19



⇒ 8 subvectors x 8 bits = 64-bit quantization index

Product quantizer: distance computation

- Asymmetric distance computation (ADC)



- Sum of square distances with quantization centroids

Product quantizer: asymmetric distance computation (ADC)

- Compute the square distance approximation in the compressed domain

$$d(x,y)^2 \approx \sum_{i=1}^{m} d(x_i, q_i(y_i))^2$$

- To compute distance between query x and many codes
  - compute $d(x_i, c_{i,j})^2$ for each subvector x_i and all possible centroids

  → stored in look-up tables
  - for each database code: sum the elementary square distances

- Each 8x8=64-bits code requires only **m=8 additions per distance**!

- IVFADC: combination with an inverted file to avoid exhaustive search

Optimizing the dimension reduction and quantization together

- VLAD vectors suffer two approximations
  - mean square error from PCA projection: $e_p(D')$
  - mean square error from quantization: $e_q(D')$

- Given k and bytes/image, choose D' minimizing their sum

Ex, k=16, 16B:

D'	$e_p(D')$	$e_q(D')$	$e_p(D')+e_q(D')$
32	0.0632	0.0164	0.0796
48	0.0508	0.0248	0.0757
64	0.0434	0.0321	**0.0755**
80	0.0386	0.0458	0.0844

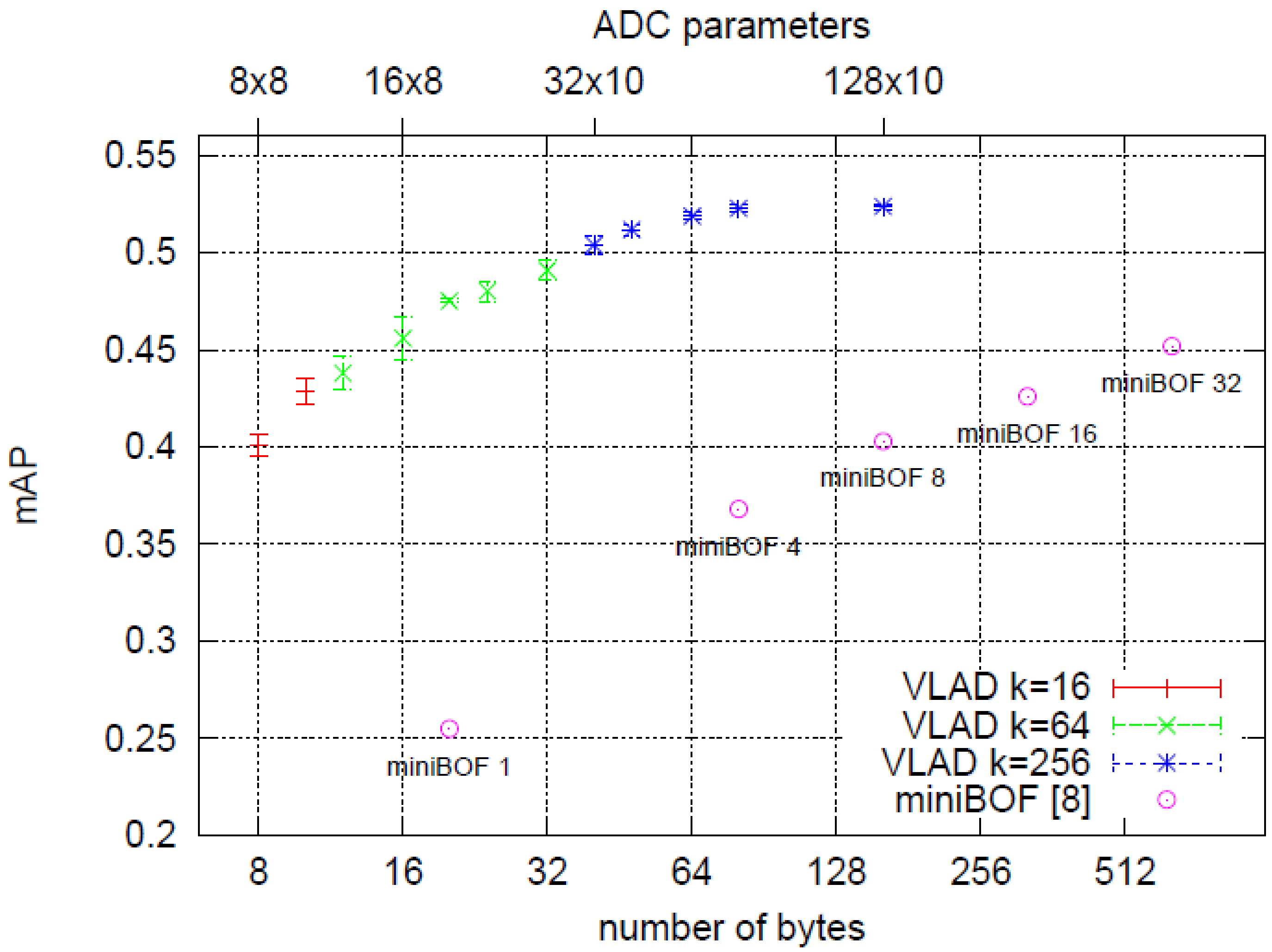
Joint optimization of VLAD and dimension reduction-indexing

- For VLAD
  - The larger *k*, the better the raw search performance
  - But large *k* produce large vectors, that are harder to index

- Optimization of the vocabulary size
  - Fixed output size (in bytes)
  - *D'* computed from *k* via the joint optimization of reduction/indexing
  - Only *k* has to be set

➔ end-to-end parameter optimization

Results on the Holidays dataset with various quantization parameters



Results on standard datasets

- Datasets
  - University of Kentucky benchmark — score: nb relevant images, max: 4
  - INRIA Holidays dataset — score: mAP (%)

Method	bytes	UKB	Holidays
BoF, k=20,000	10K	2.92	44.6
BoF, k=200,000	12K	3.06	54.9
miniBOF	20	2.07	25.5
miniBOF	160	2.72	40.3
VLAD k=16, ADC 16 x 8	**16**	**2.88**	**46.0**
VLAD k=64, ADC 32 x10	**40**	**3.10**	**49.5**

D' =64 for *k*=16 and *D'* =96 for *k*=64

ADC (subvectors) x (bits to encode each subvector)

miniBOF: “Packing Bag-of-Features”, ICCV’09

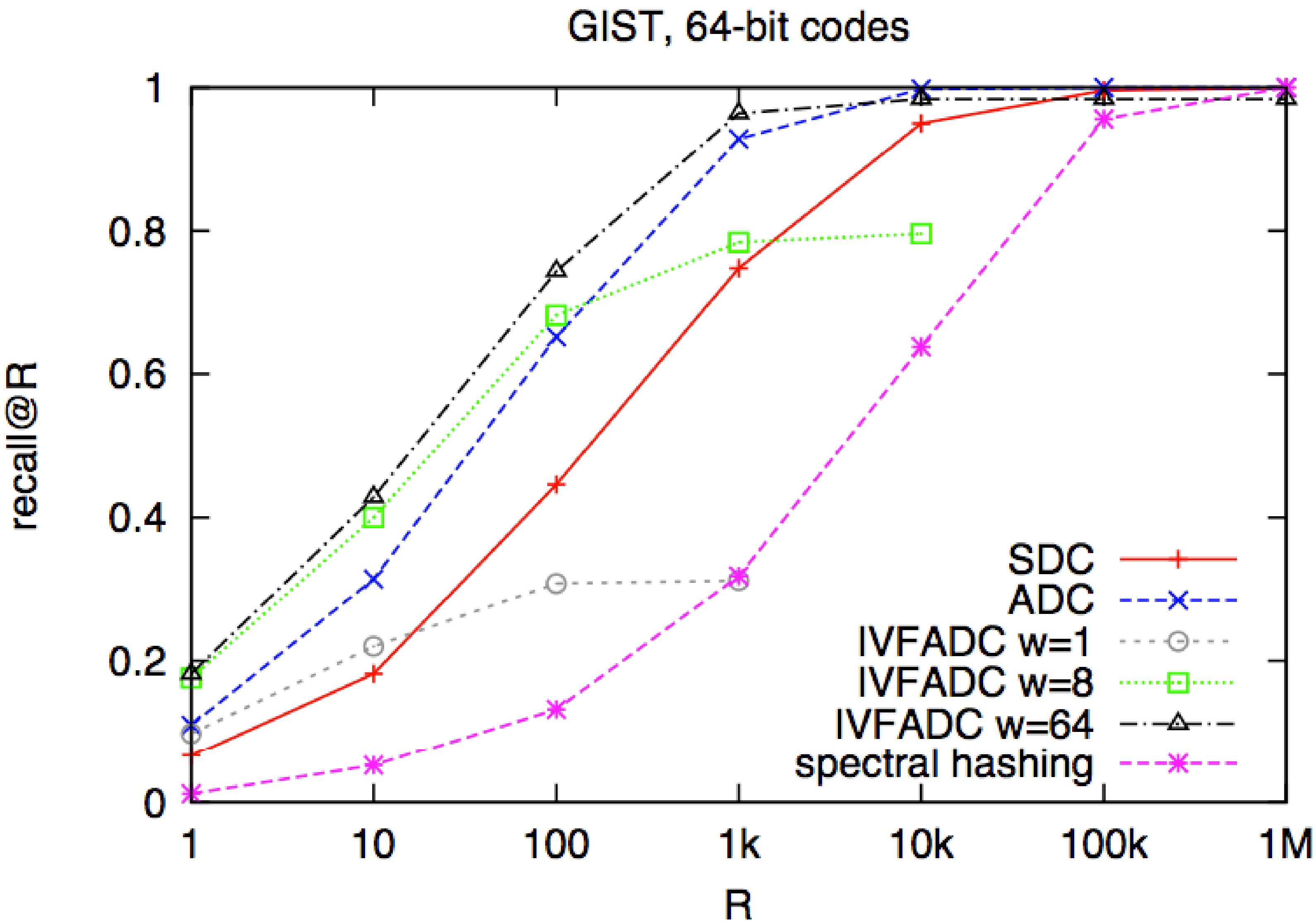
Comparison BOF / VLAD + ADC

- Datasets
  - INRIA Holidays dataset , score: mAP (%)

Method	Holidays
BOF, k=2048, D'= 64, ADC 16x8	42.5
VLAD k=16,D=2048, D' = 64, ADC 16 x 8	46.0
BOF, k=8192, D'= 128, AD16x8	41.9
VLAD k=64, D= 8192, D'=128, ADC 16X8	45.8

- VLAD improves results over BOF
- Product quantizer gives excellent results for BOF!

Searching with quantization: comparison with spectral Hashing

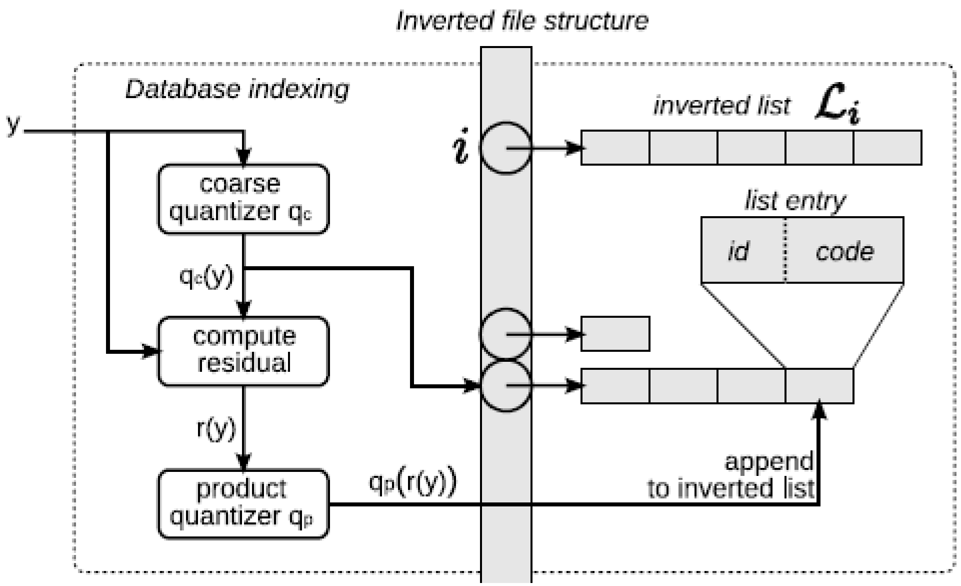


Large scale experiments (10 million images)

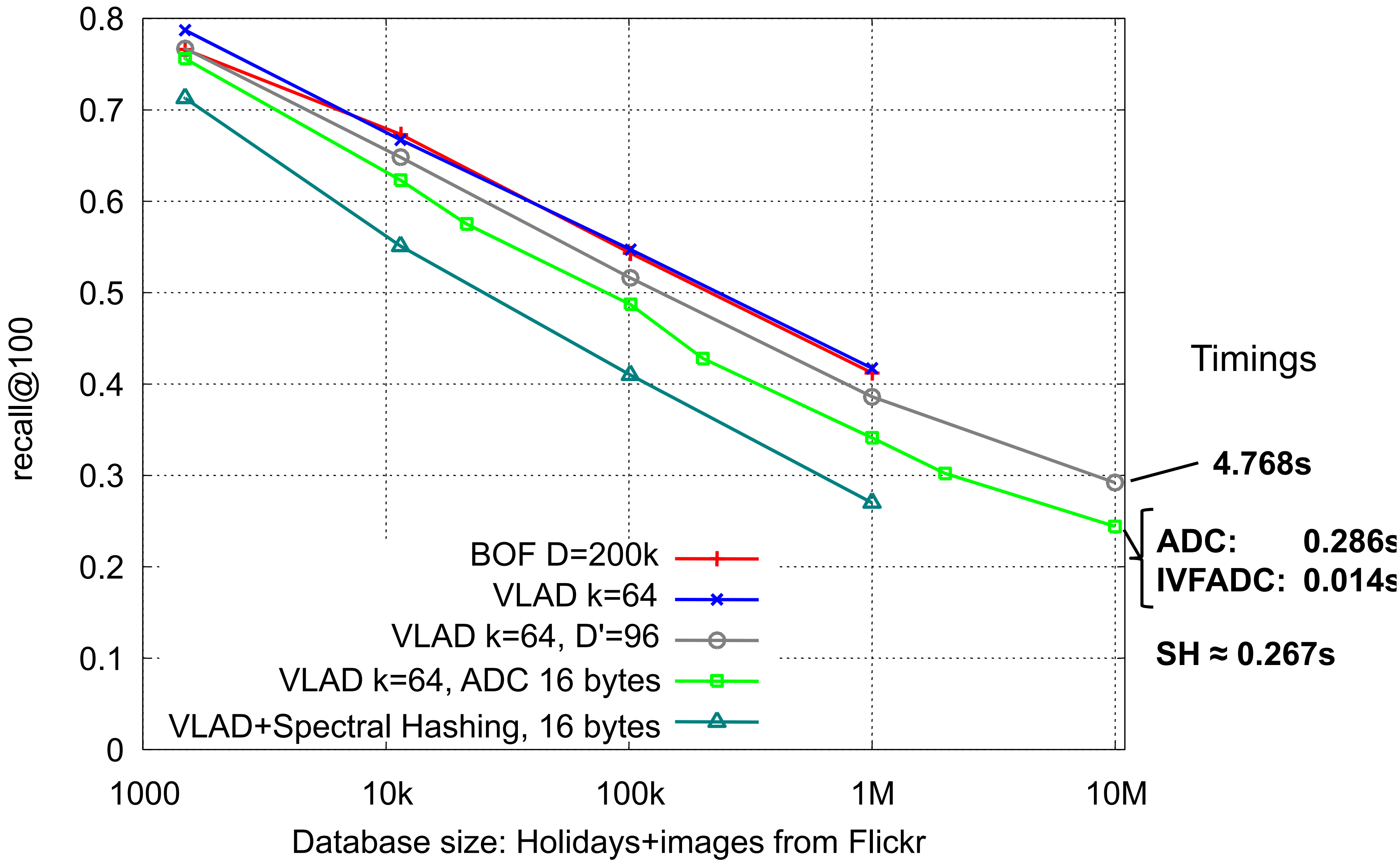
- Exhaustive search of VLADs, D'=64
  - 4.77s

- With the product quantizer
  - Exhaustive search with ADC: 0.29s
  - Non-exhaustive search with IVFADC: 0.014s

IVFADC -- Combination with an inverted file



Large scale experiments (10 million images)



Conclusion & future work

- Excellent search accuracy and speed in 10 million of images
- Each image is represented by very few bytes (20 – 40 bytes)
- Tested on up to 220 million video frame
  - extrapolation for 1 billion images: 20GB RAM, query < 1s on 8 cores
- On-line available:
  - Matlab source code of ADC
- Improved Fisher kernels by Perronnin et al., CVPR'2010
- Extension to video & more "semantic" search