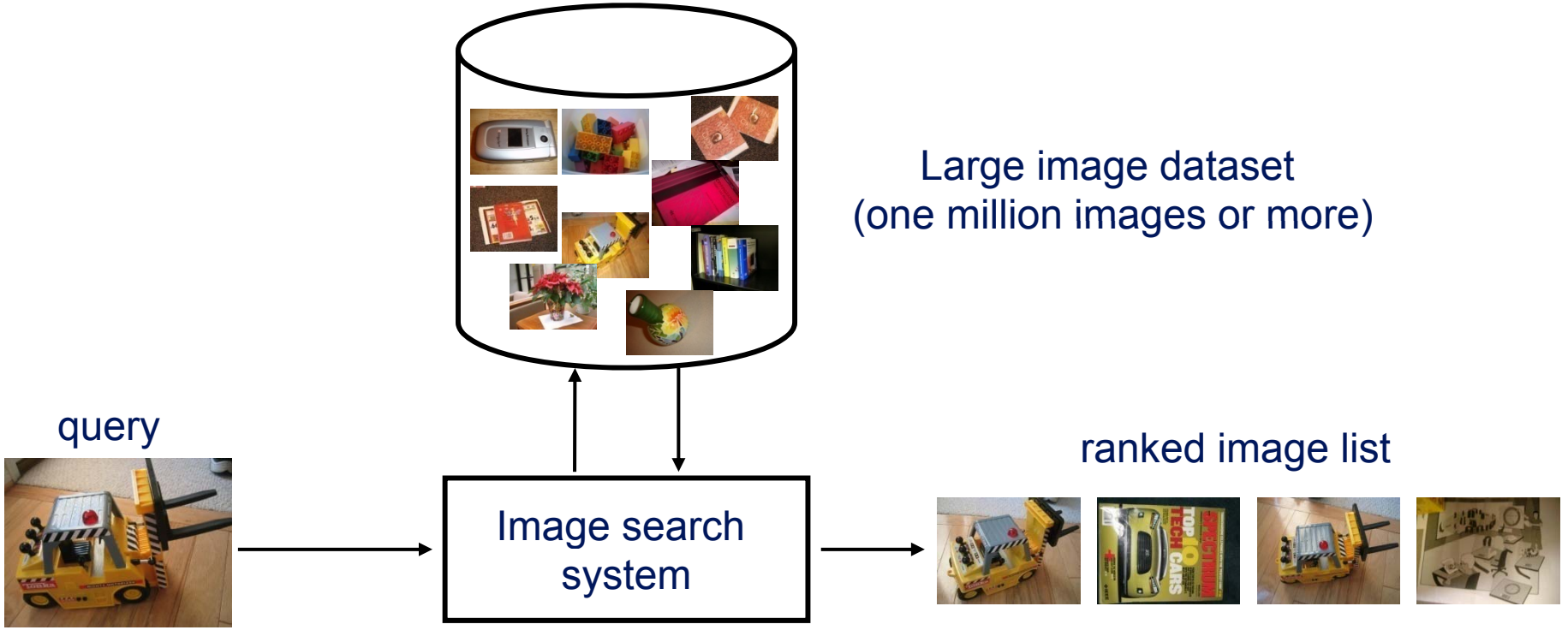
Overview

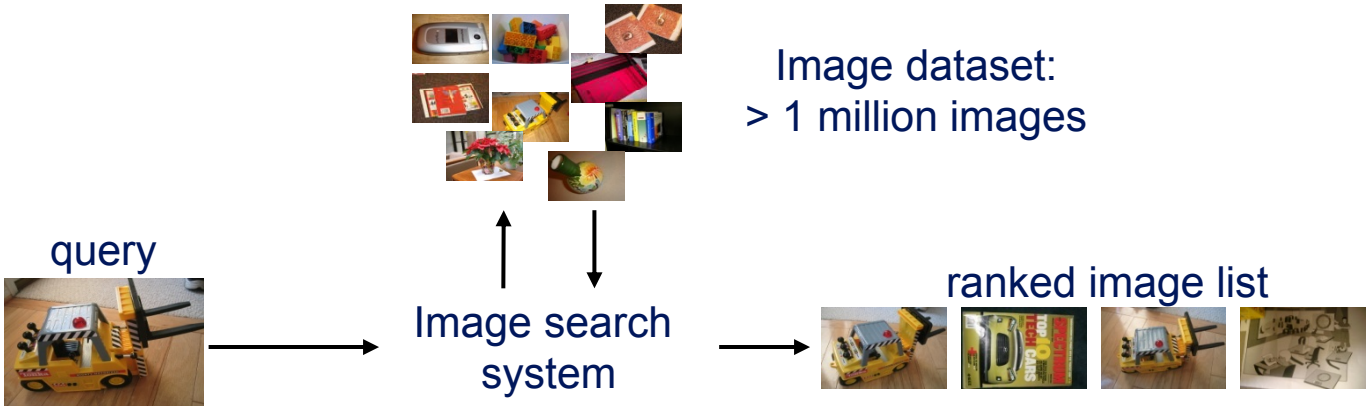
- Local invariant features (C. Schmid)
- Matching and recognition with local features (J. Sivic)
- Efficient visual search (J. Sivic)
- **Very large scale search** (C. Schmid)
- Practical session

Image search system for large datasets



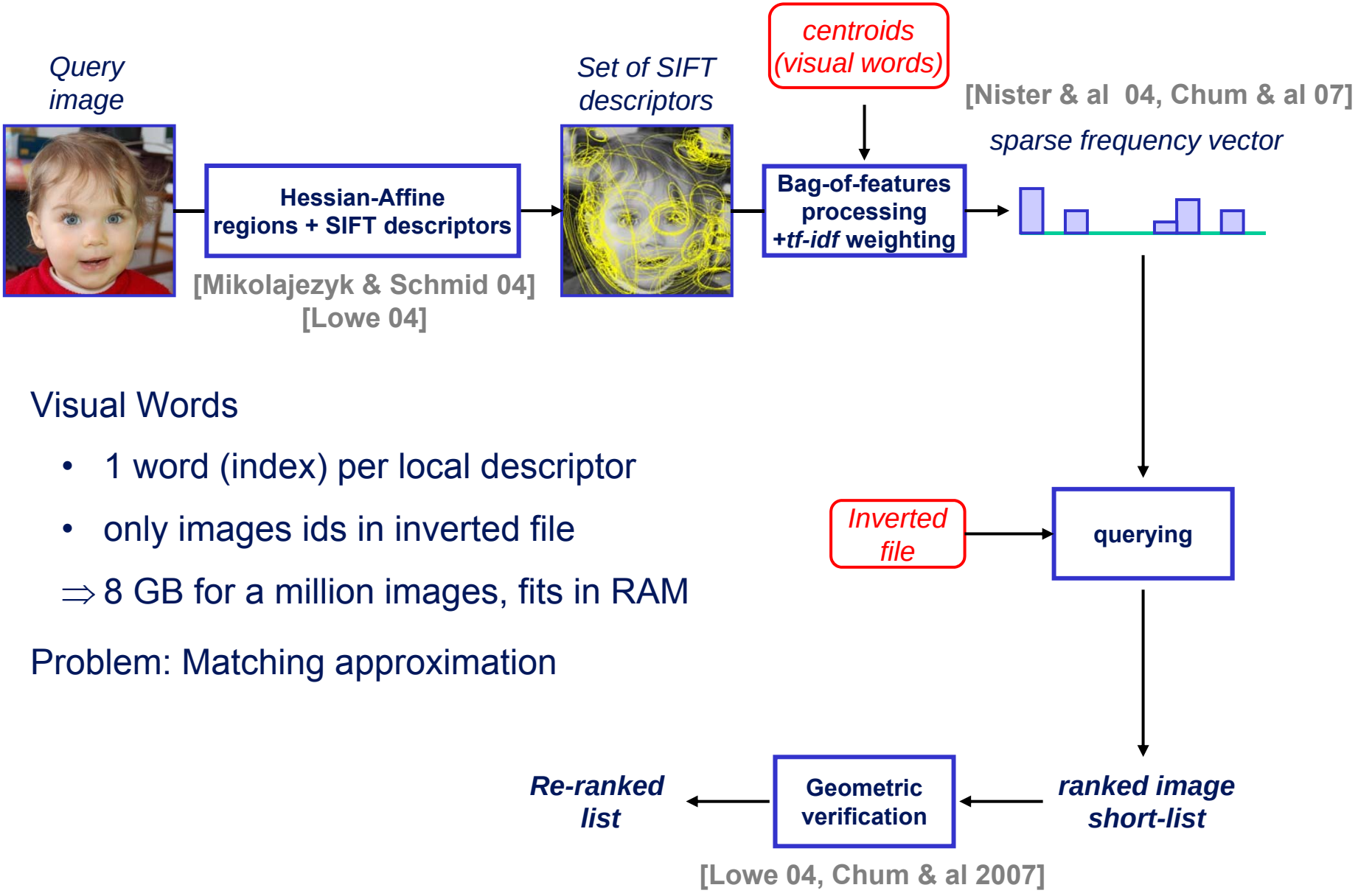
- **Issues** for very large databases
  - to reduce the query time
  - to reduce the storage requirements
  - with minimal loss in retrieval accuracy

Large scale object/scene recognition



- Each image described by approximately 2000 descriptors
  - $2 * 10^9$ descriptors to index for one million images!
- Database representation in RAM:
  - Size of descriptors : 1 TB, search+memory intractable

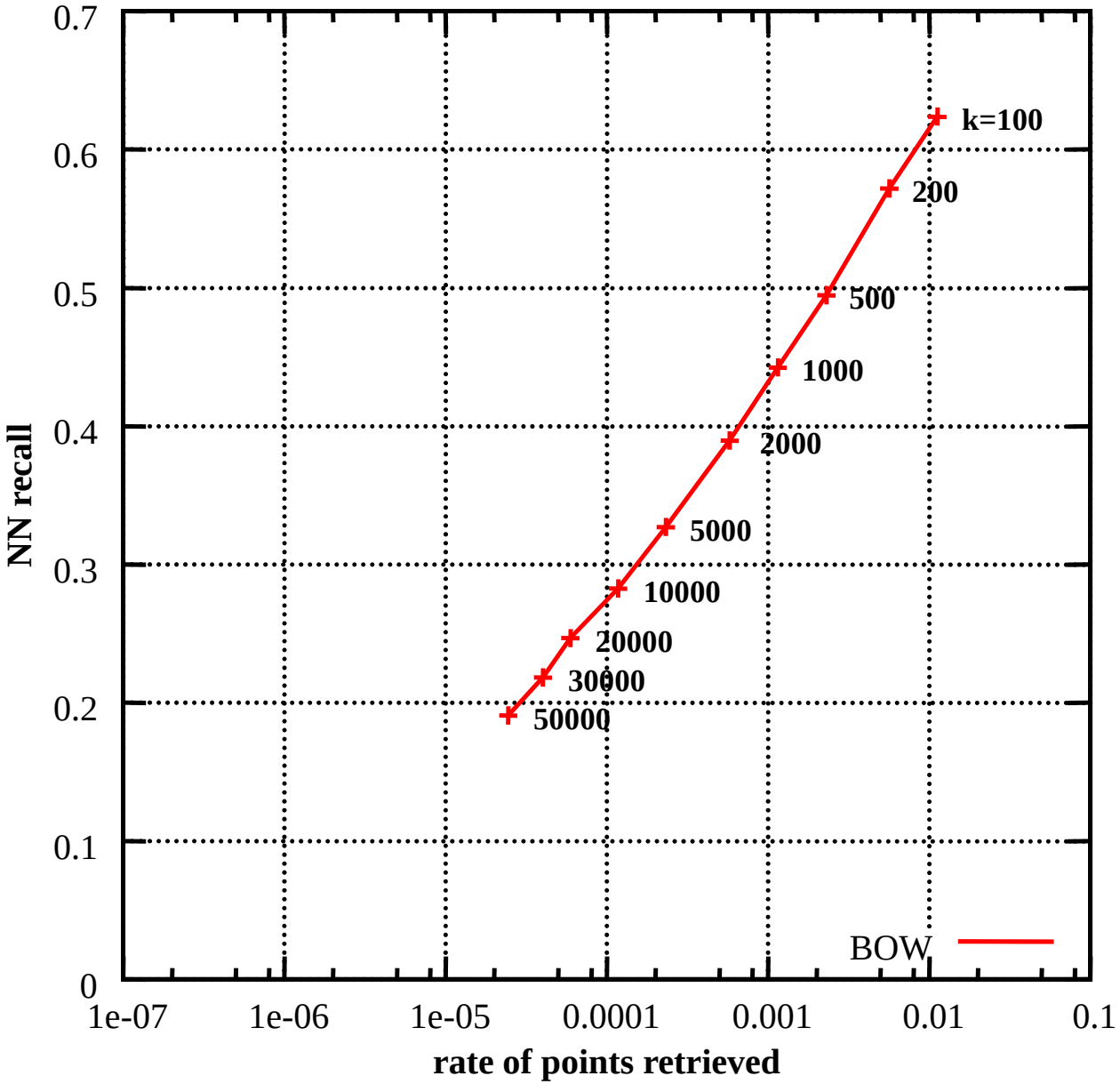
Bag-of-words [Sivic & Zisserman'03]



- Visual Words
  - 1 word (index) per local descriptor
  - only images ids in inverted file

  $\Rightarrow$ 8 GB for a million images, fits in RAM
- Problem: Matching approximation

Approximate nearest neighbour (ANN) evaluation of bag-of-features



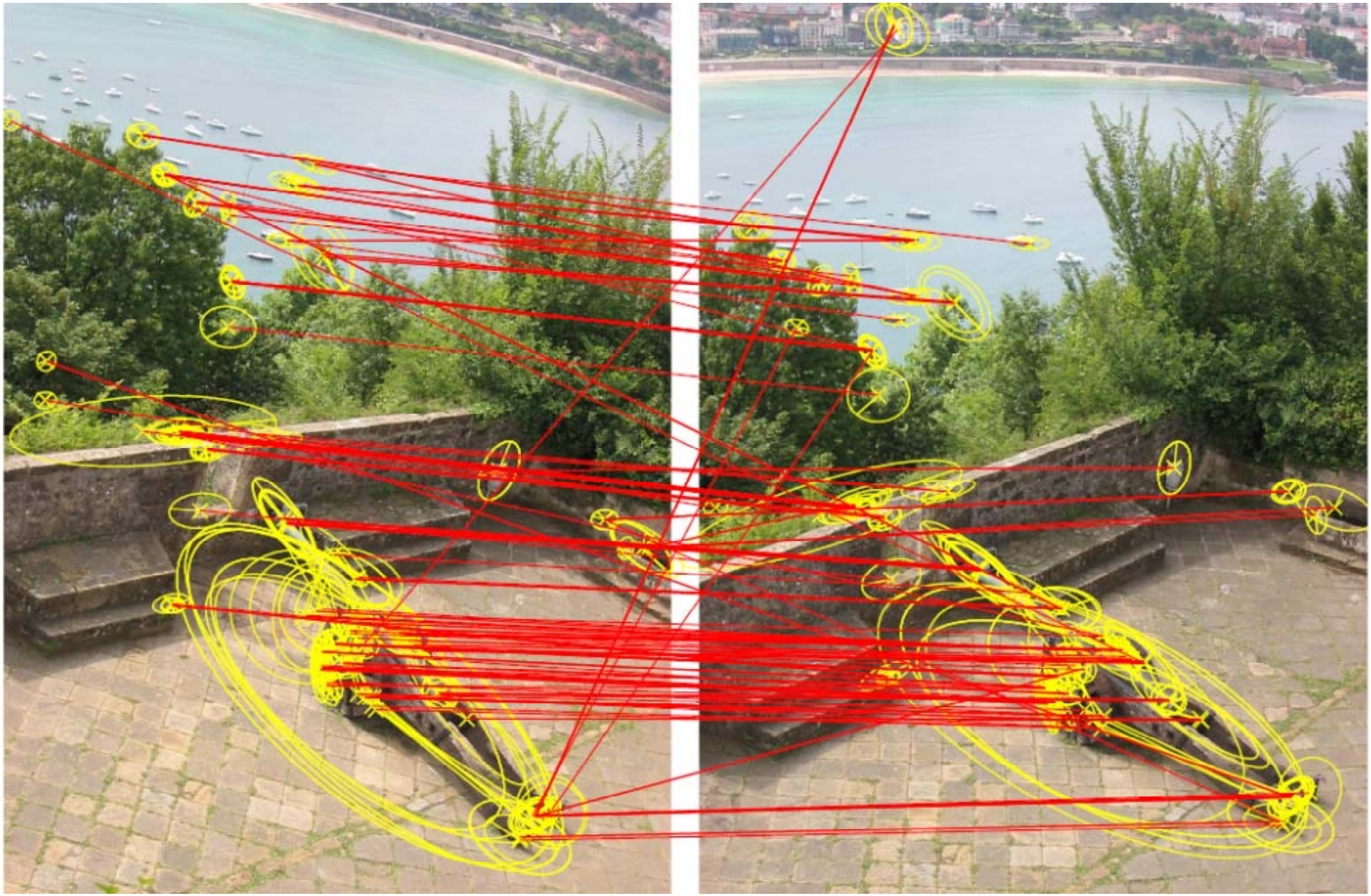
ANN algorithms returns a list of potential neighbors

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

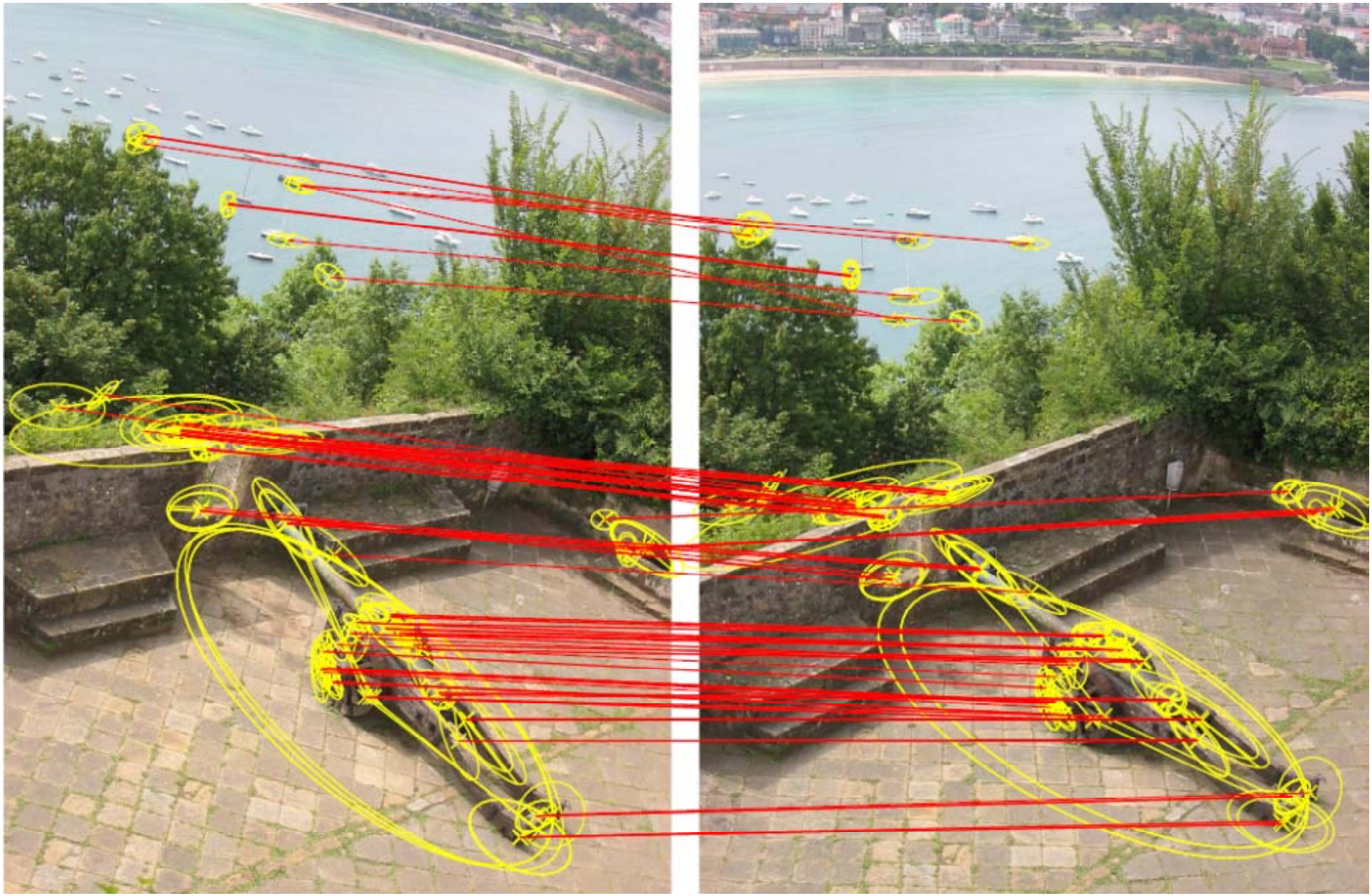
Ambiguity removal:
= proportion of vectors in the short-list

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

20K visual word: false matches



200K visual word: good matches missed



Problem with bag-of-features

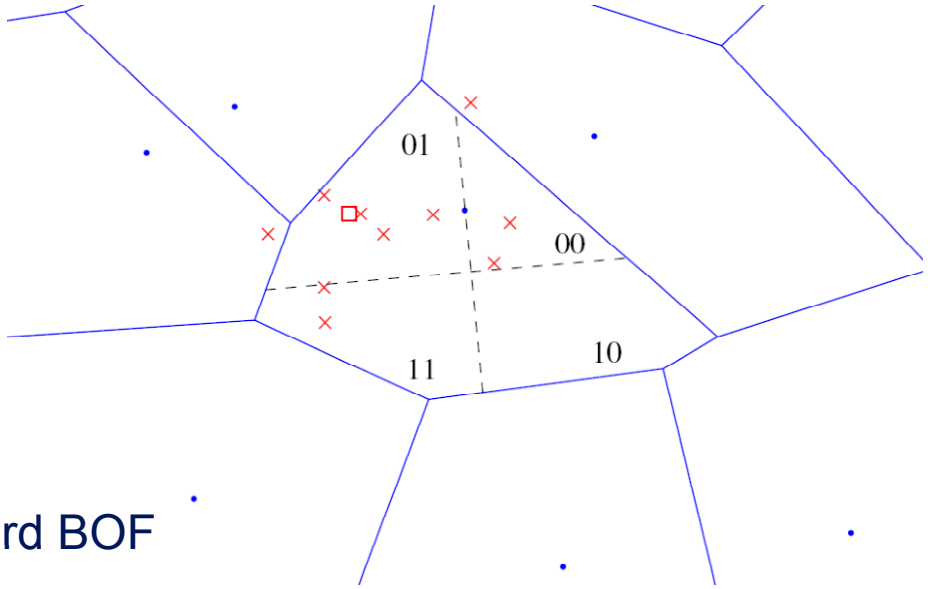
- The intrinsic matching scheme performed by BOF is weak
  - for a “small” visual dictionary: too many false matches
  - for a “large” visual dictionary: many true matches are missed

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

  → intrinsic approximate nearest neighbor search of BOF is not sufficient

  - Possible solutions
    - Soft assignment [Philbin et al. CVPR’08]
    - Additional short codes [Jegou et al. ECCV’08]

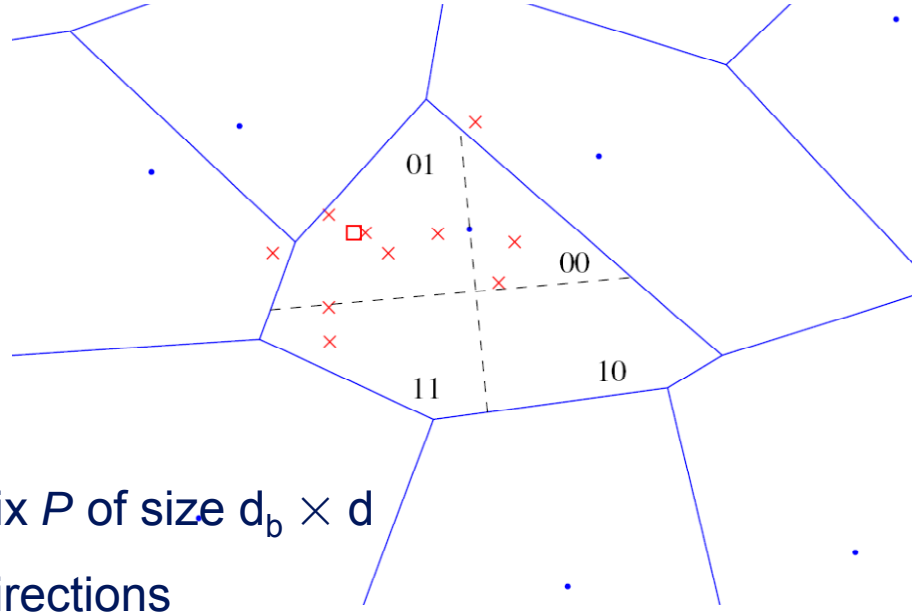
Hamming Embedding



- 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 Hamming 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



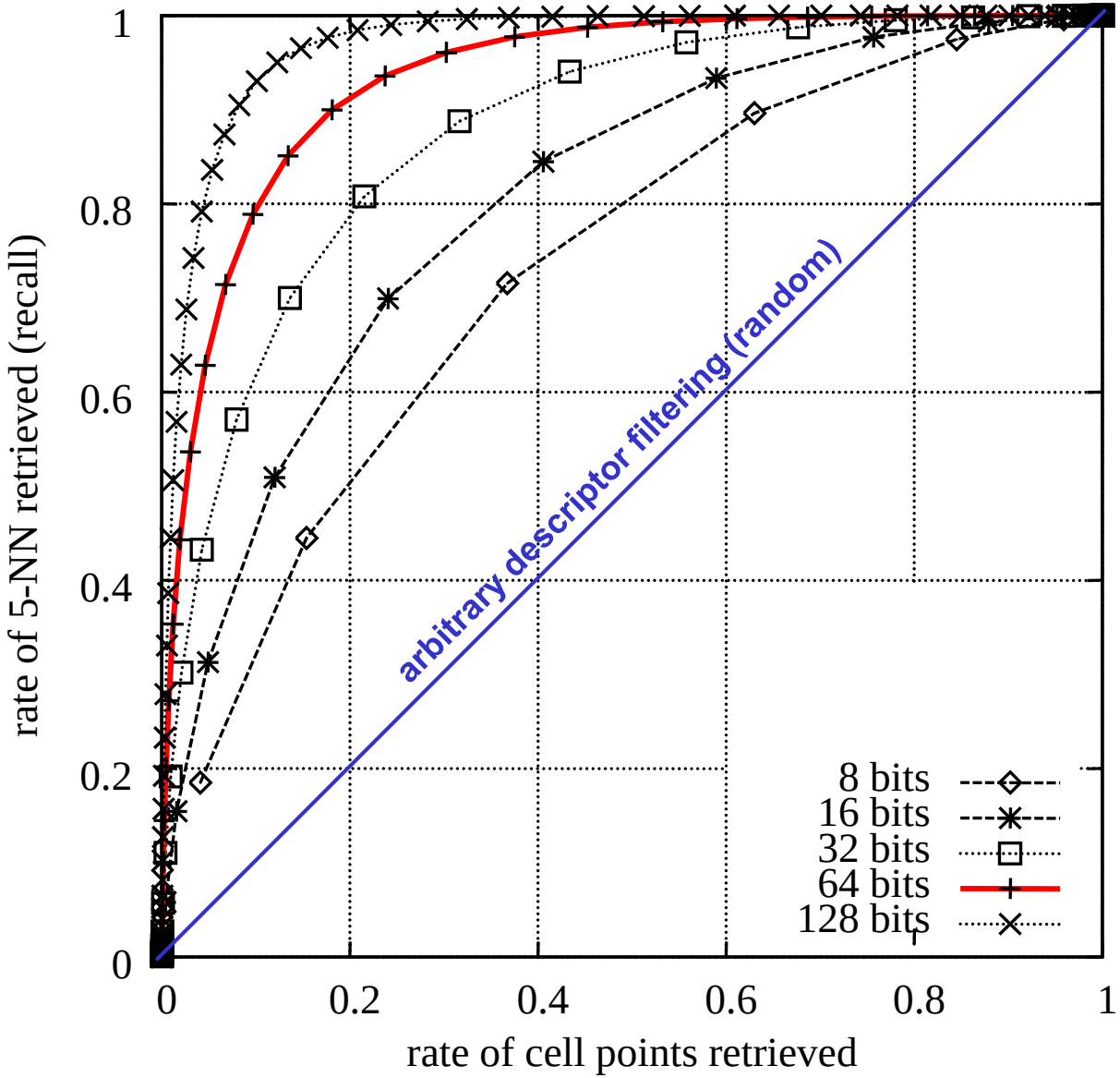
- **Off-line** (given a quantizer)
  - draw an orthogonal projection matrix P of size $d_b \times d$

    $\rightarrow$ 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

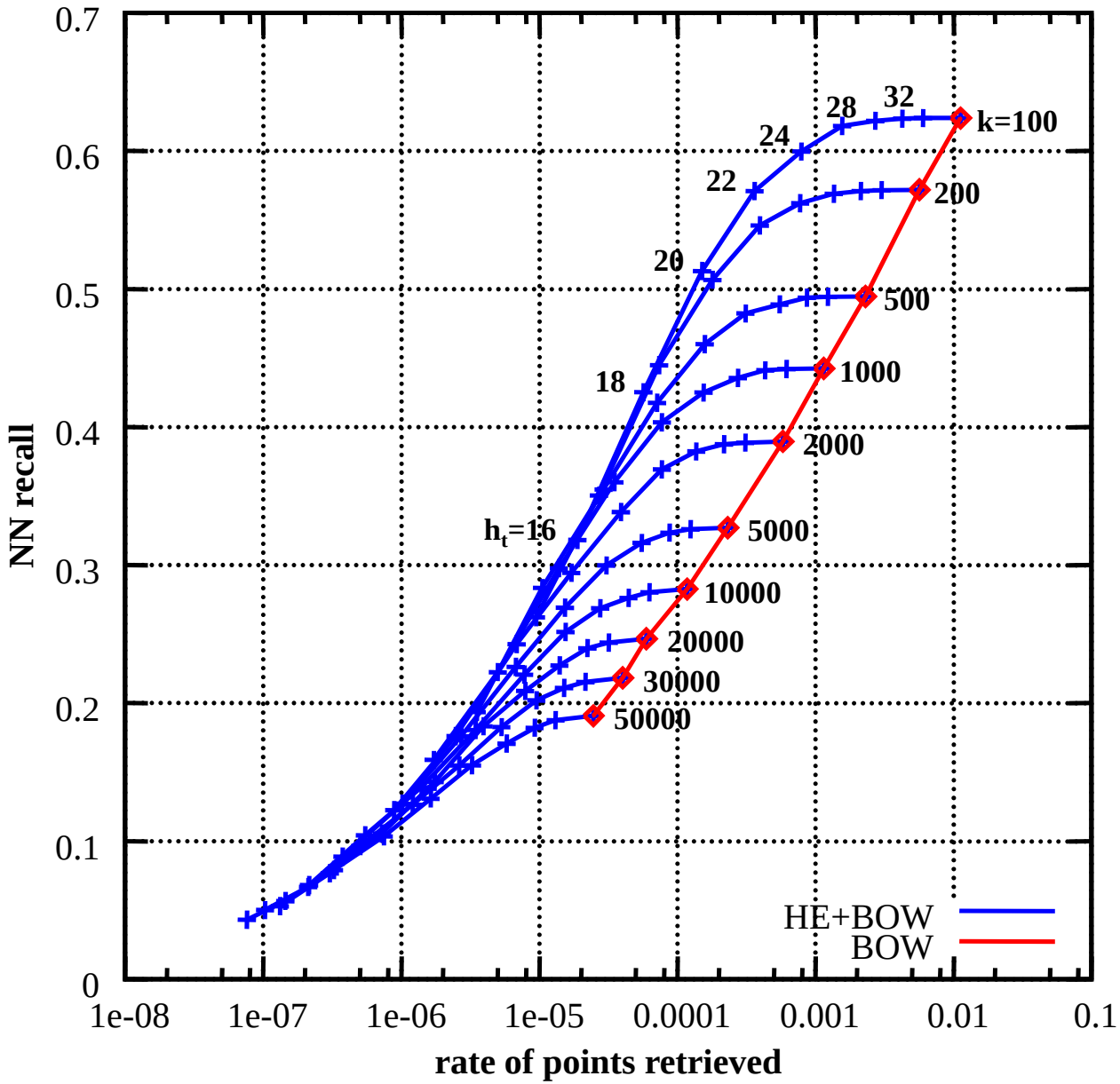


- trade-off between memory usage and accuracy

→ more bits yield higher accuracy

In practice 64 bits (8 bytes)

ANN evaluation of Hamming Embedding



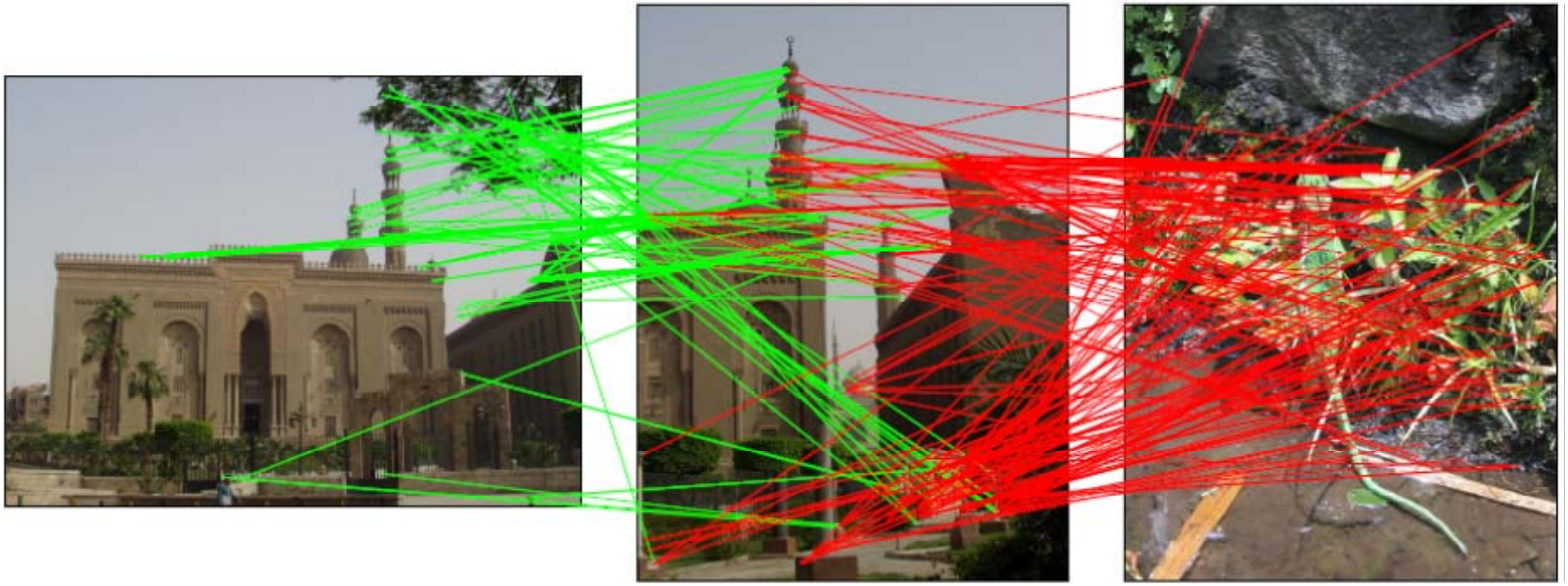
compared to BOW: at least 10 times less points in the short-list for the same level of accuracy

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

Matching points - 20k word vocabulary

201 matches

240 matches

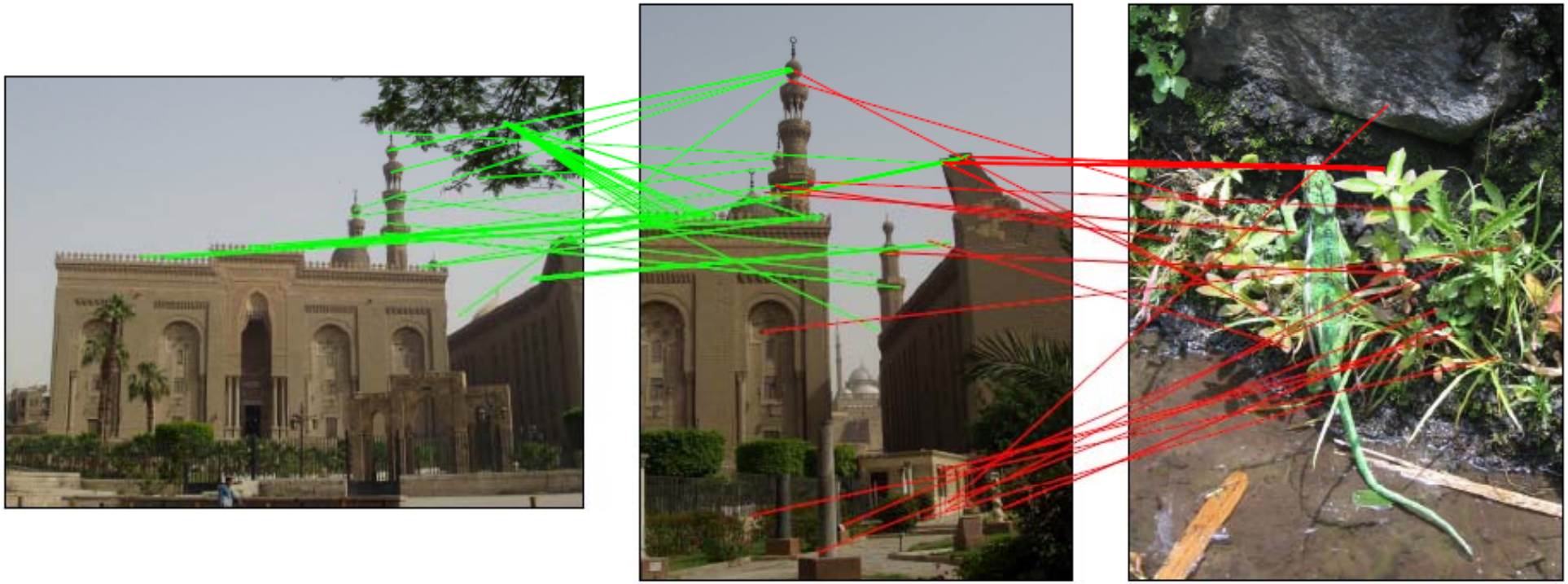


Many matches with the non-corresponding image!

Matching points - 200k word vocabulary

69 matches

35 matches

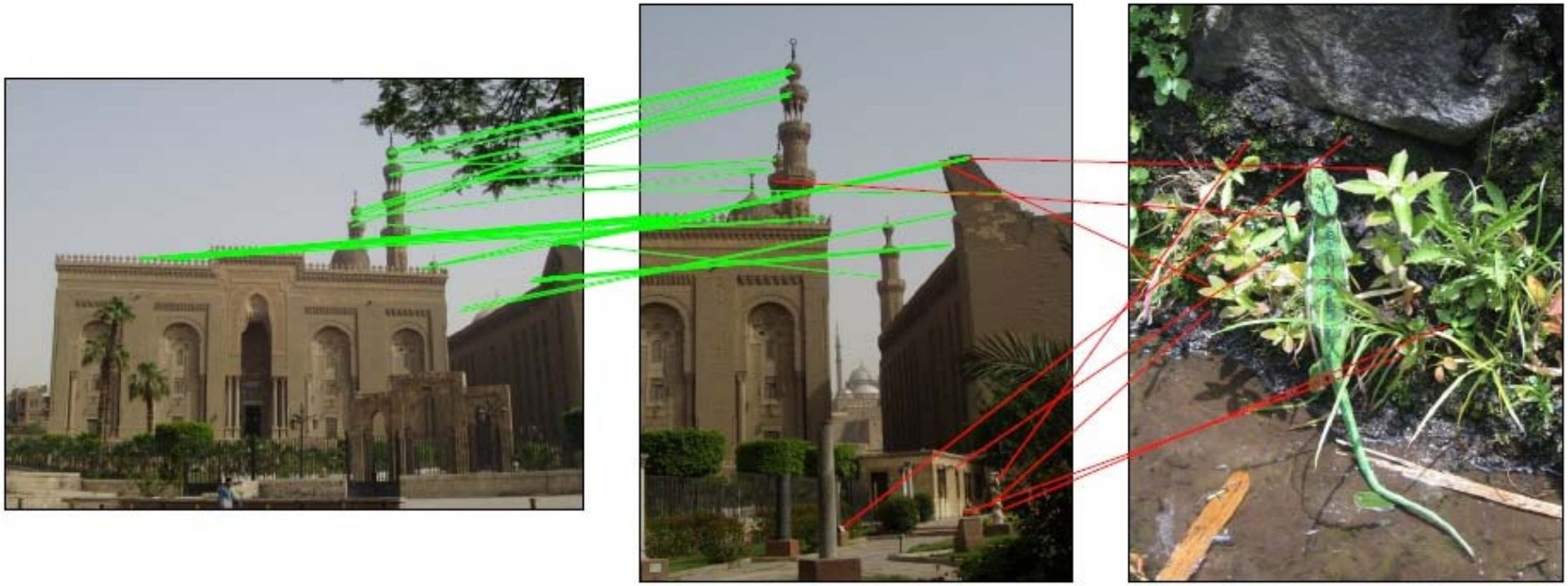


Still many matches with the non-corresponding one

Matching points - 20k word vocabulary + HE

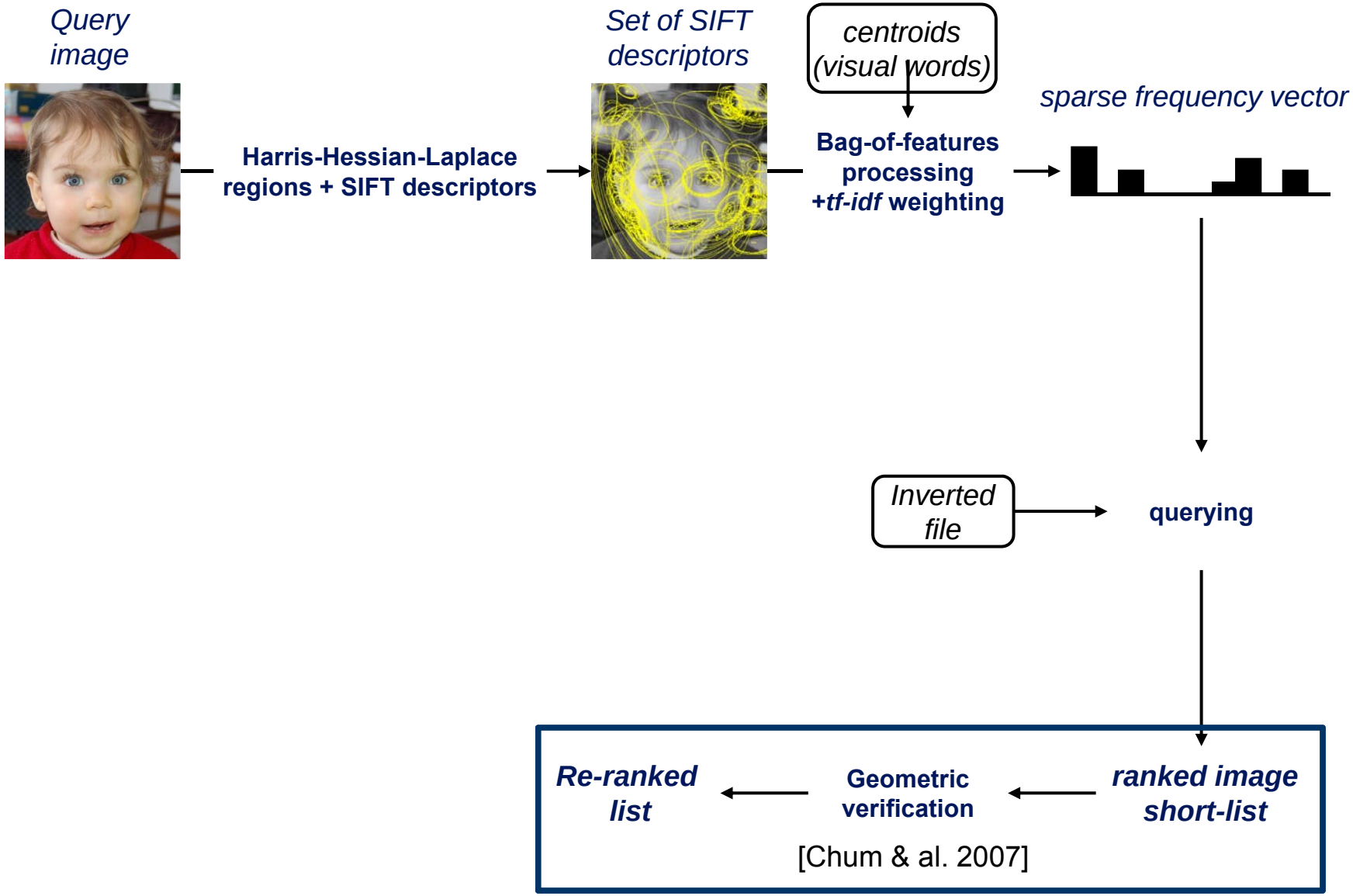
83 matches

8 matches



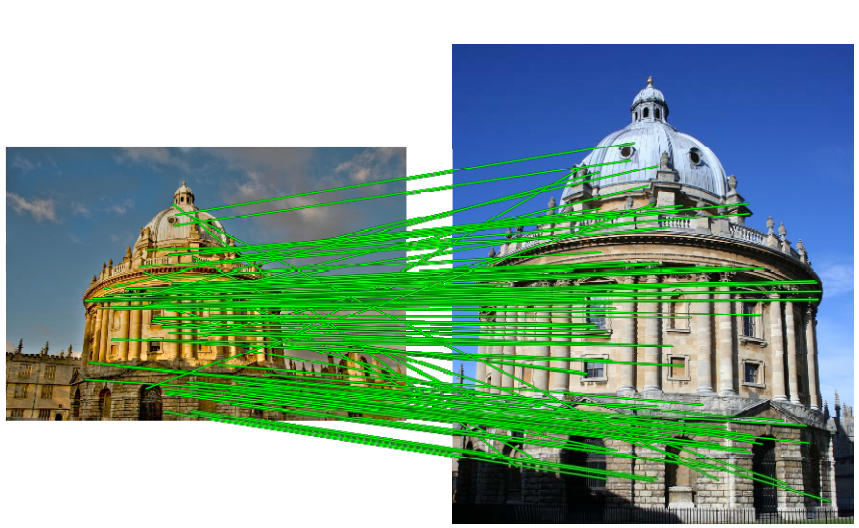
10x more matches with the corresponding image!

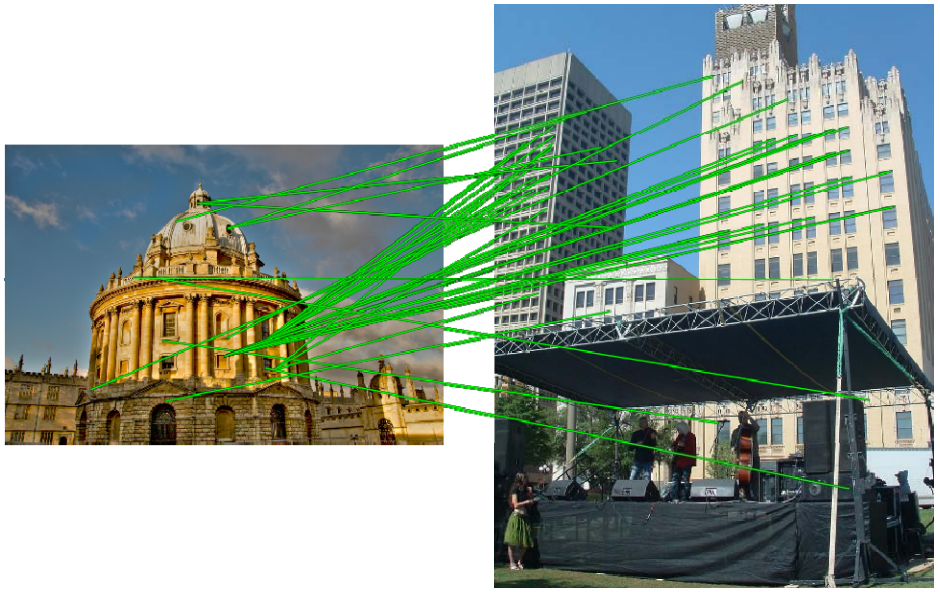
Bag-of-features [Sivic&Zisserman'03]



Geometric verification

Use the **position** and **shape** of the underlying features to improve retrieval quality

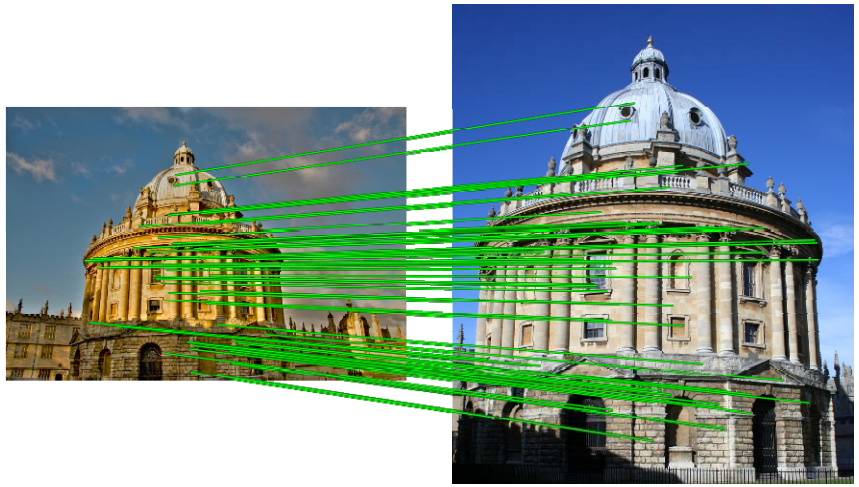




Both images have many matches – which is correct?

Geometric verification

We can measure **spatial consistency** between the query and each result to improve retrieval quality

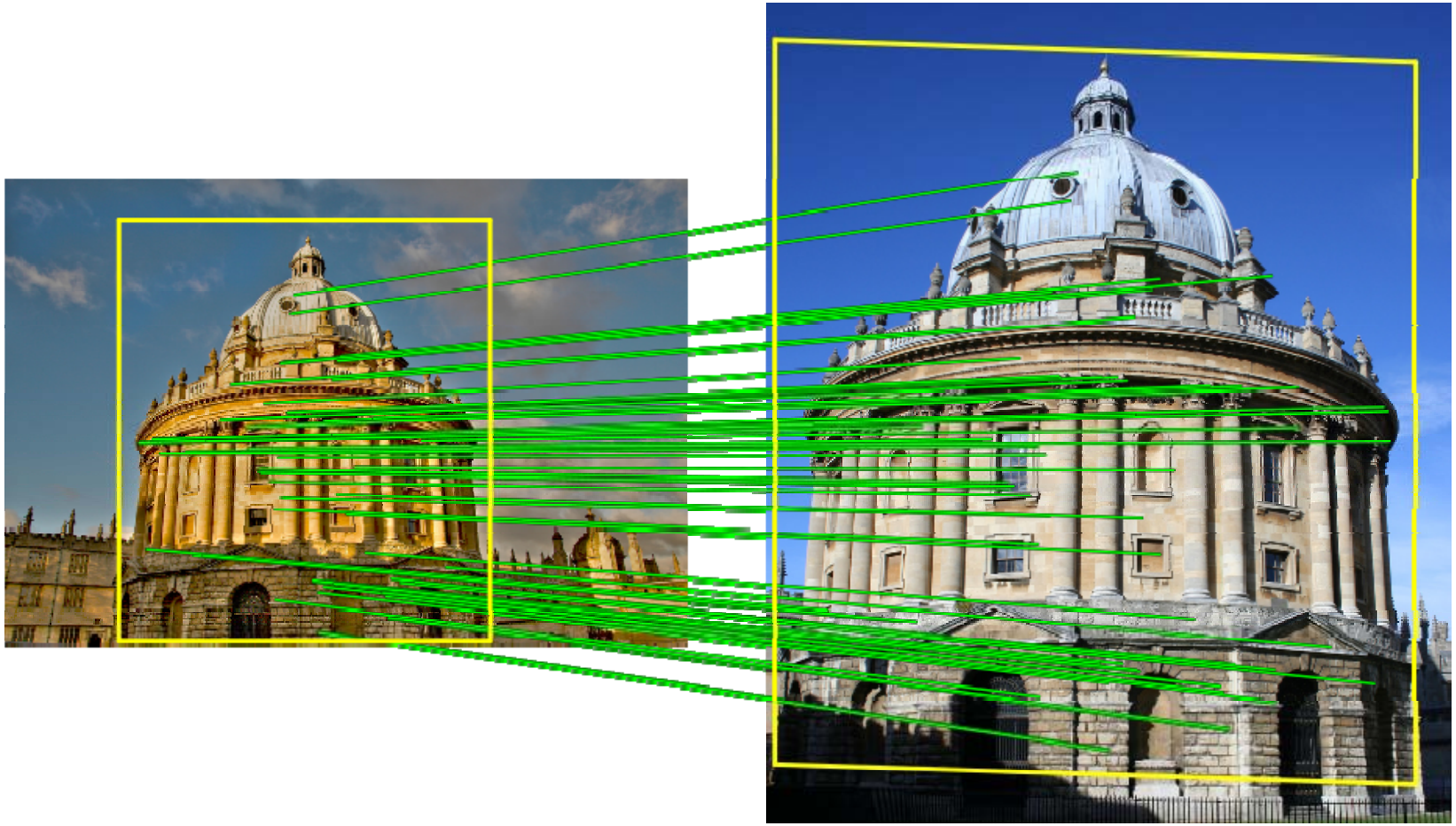


Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**

Geometric verification

Gives **localization** of the object

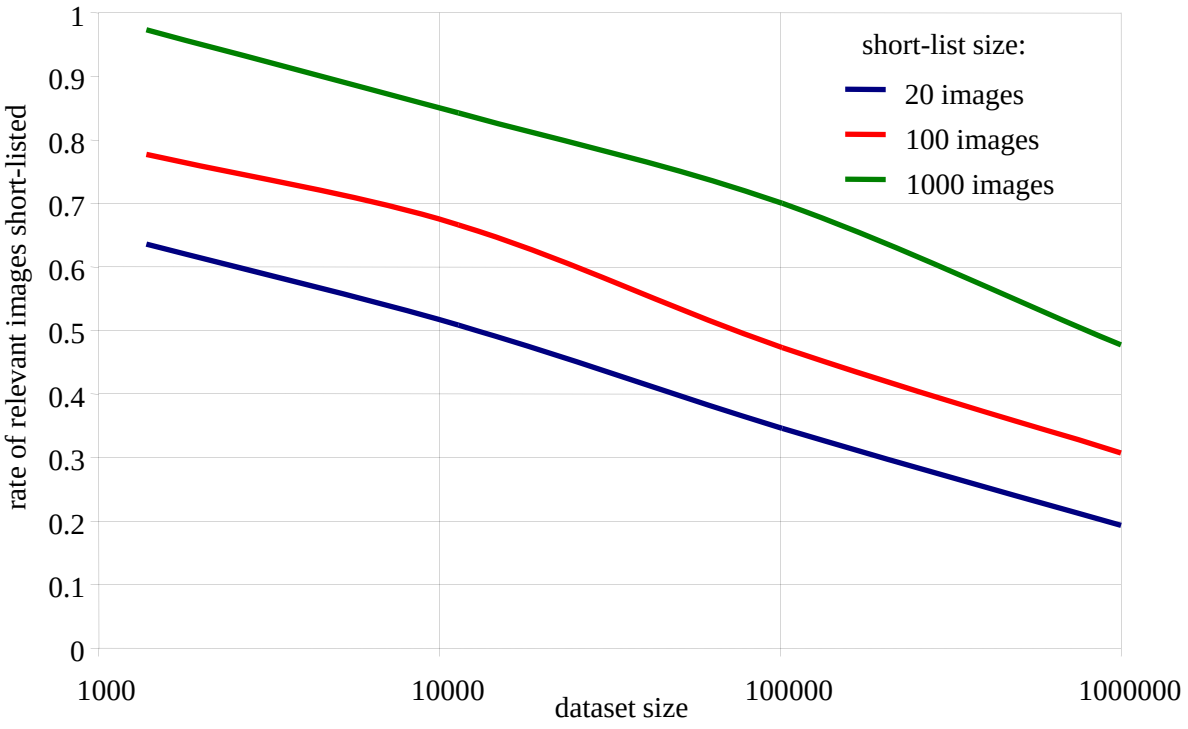


Re-ranking based on geometric verification

- works very well
- but performed on a short-list only (100 - 1000 images)

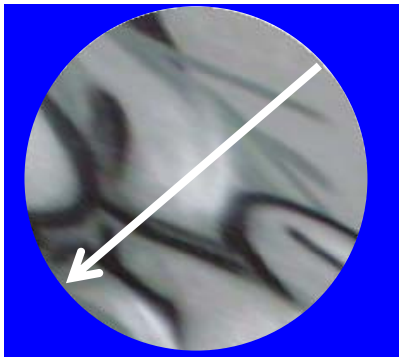
  → for very large datasets, the number of distracting images is so high that relevant images are not even short-listed!

  → weak geometry



Weak geometry consistency

- Weak geometric information used for **all** images (not only the short-list)
- Each invariant interest region detection has a scale and rotation angle associated, here characteristic scale and dominant gradient orientation

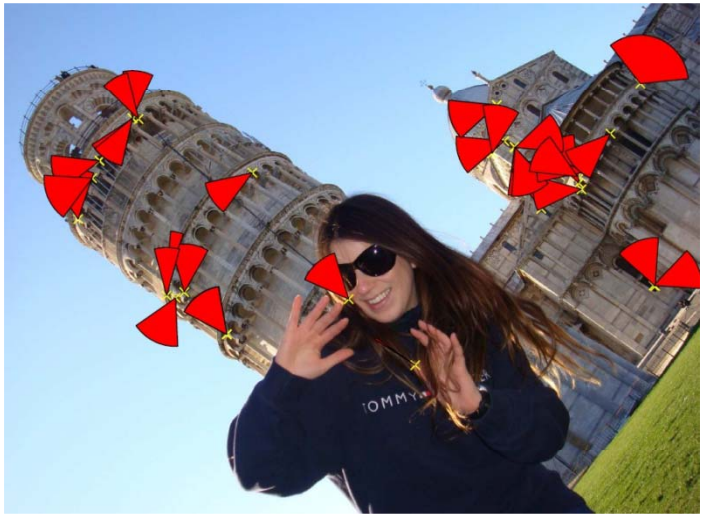


Scale change 2

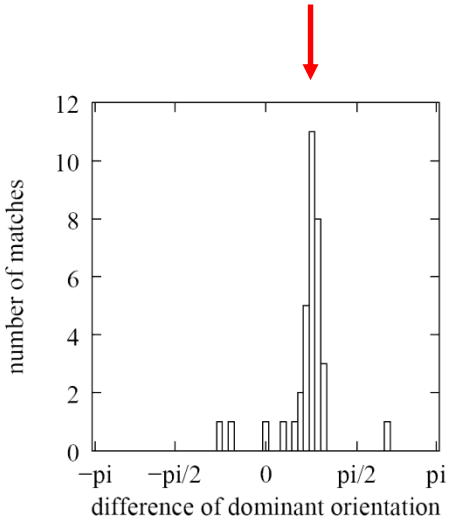
Rotation angle ca. 20 degrees

- Each matching pair results in a scale and angle difference
- For the global image scale and rotation changes are roughly consistent

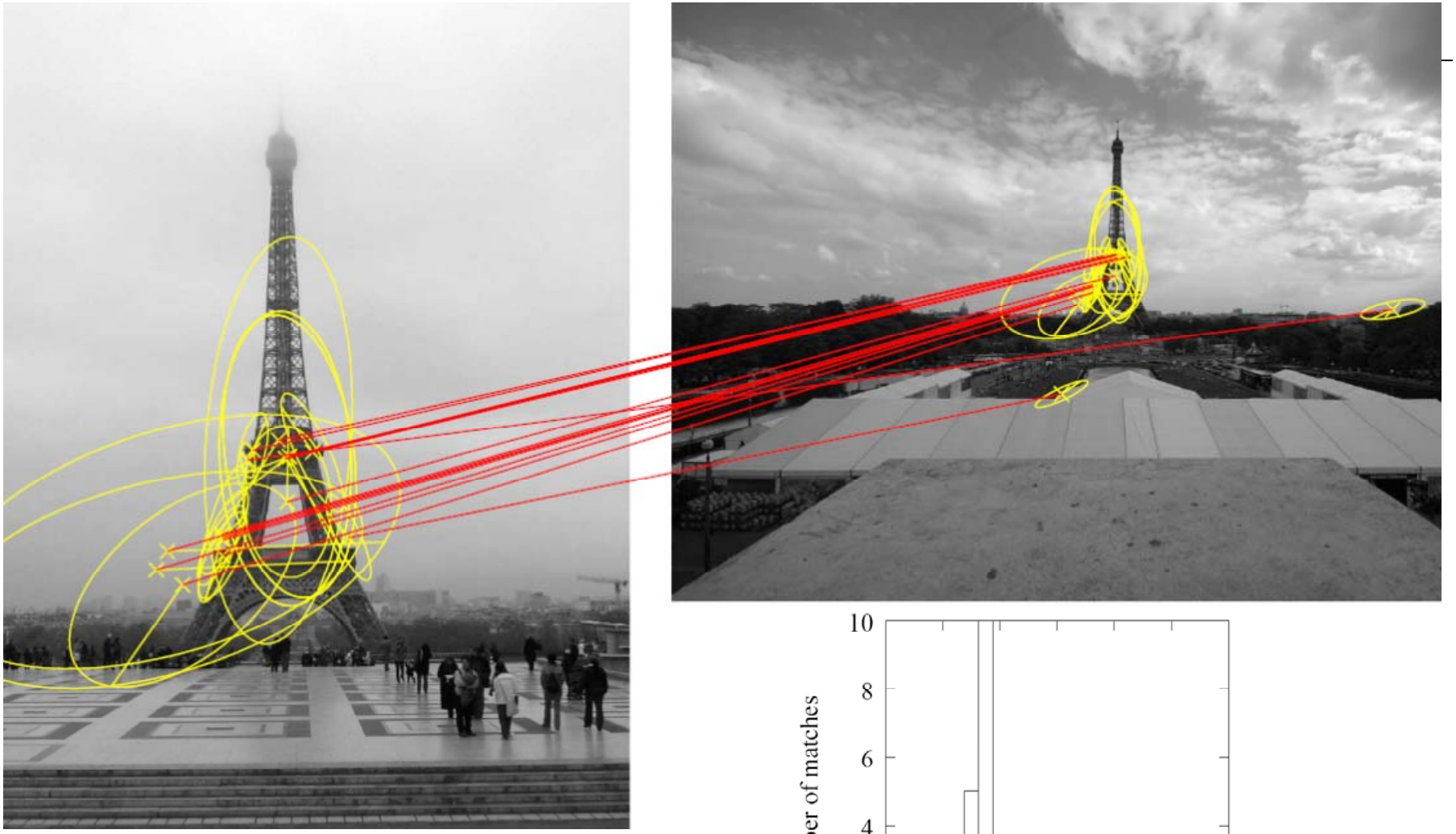
WGC: orientation consistency



Max = rotation angle between images



WGC: scale consistency



Weak geometry consistency

- Integration of the geometric verification into the BOF
  - votes for an image in two quantized subspaces, i.e. for angle & scale
  - these subspace are show to be roughly independent
  - final score: filtering for each parameter (angle and scale)

- Only matches that do agree with the main difference of orientation and scale will be taken into account in the final score

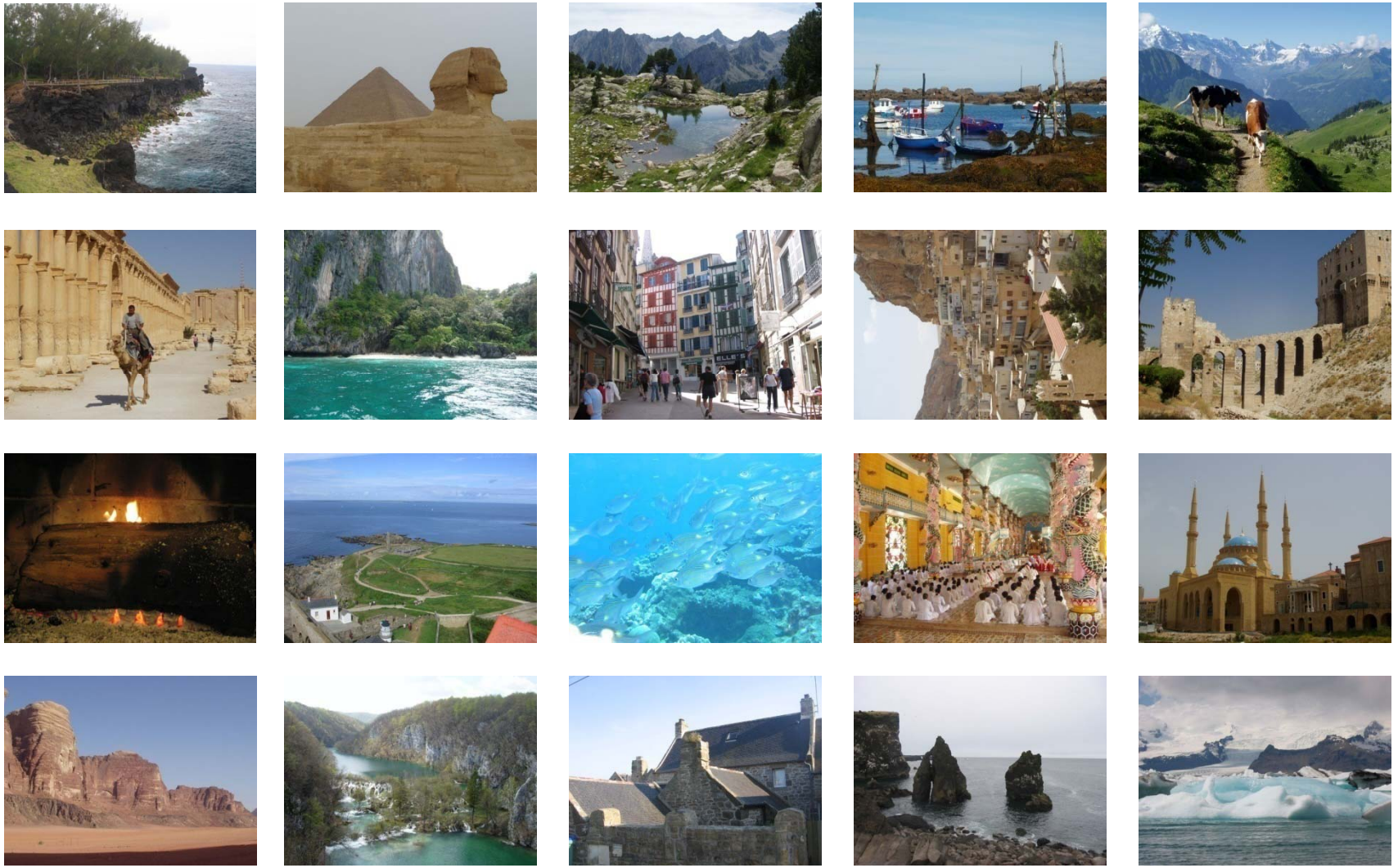
- Re-ranking using full geometric transformation still adds information in a final stage

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

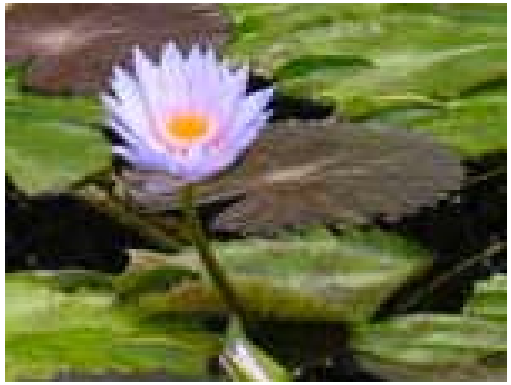
Example distractors - Flickr





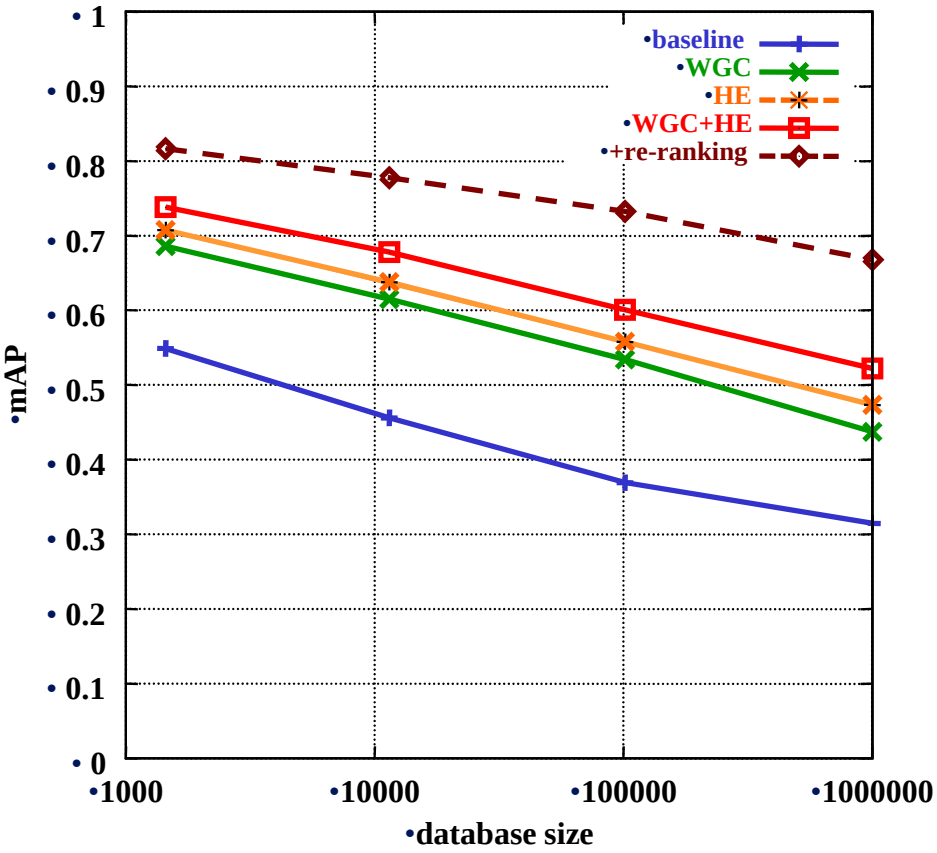






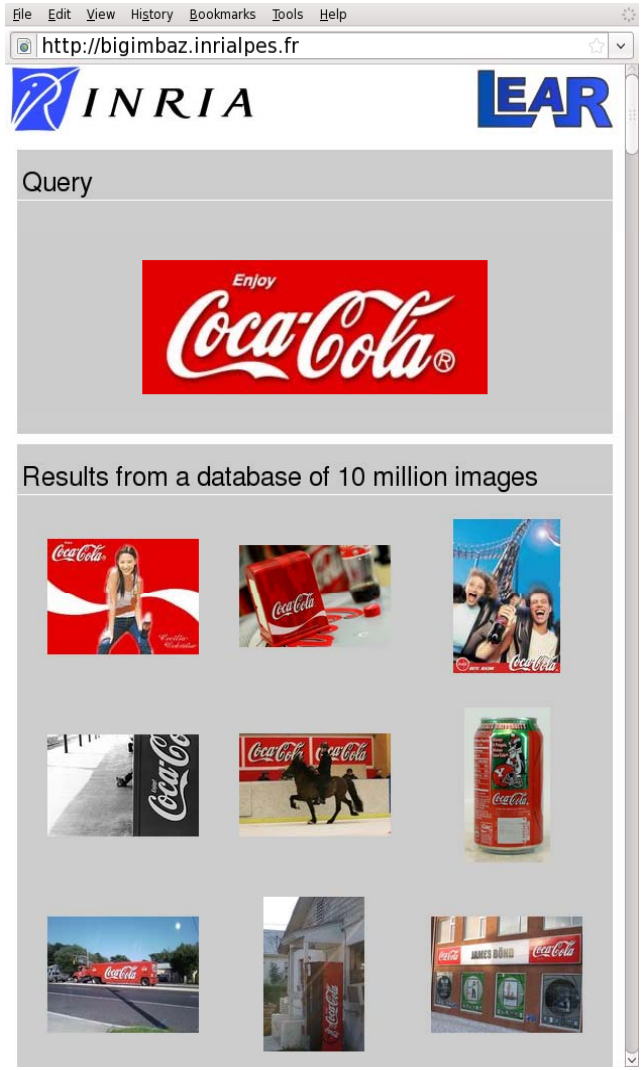

Experimental evaluation

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



Average query time (4 CPU cores)	
Compute descriptors	880 ms
Quantization	600 ms
Search – baseline	**620** ms
Search – WGC	**2110** ms
Search – HE	**200** ms
Search – HE+WGC	**650** ms

Results – Venice Channel

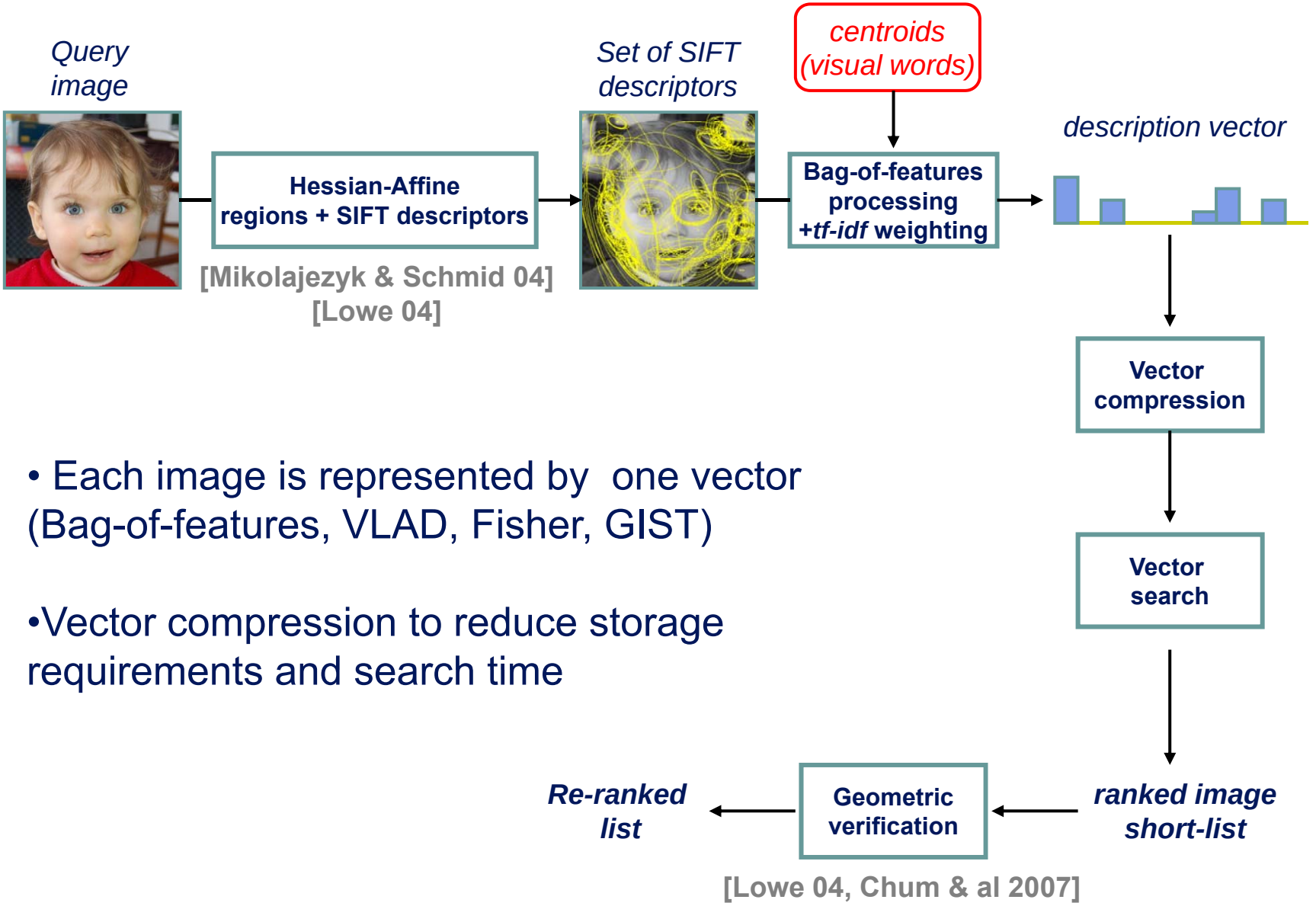


Demo at http://bigimbaz.inrialpes.fr

Towards large-scale image search

- BOF+inverted file can handle up to ~10 millions images
  - with a limited number of descriptors per image → RAM: 40GB
  - search: 2 seconds
- Web-scale = billions of images
  - with 100 M per machine → search: 20 seconds, RAM: 400 GB
  - not tractable
- Solution: represent each image by one compressed vector

Very large scale image search



• Each image is represented by one vector (Bag-of-features, VLAD, Fisher, GIST)

•Vector compression to reduce storage requirements and search time

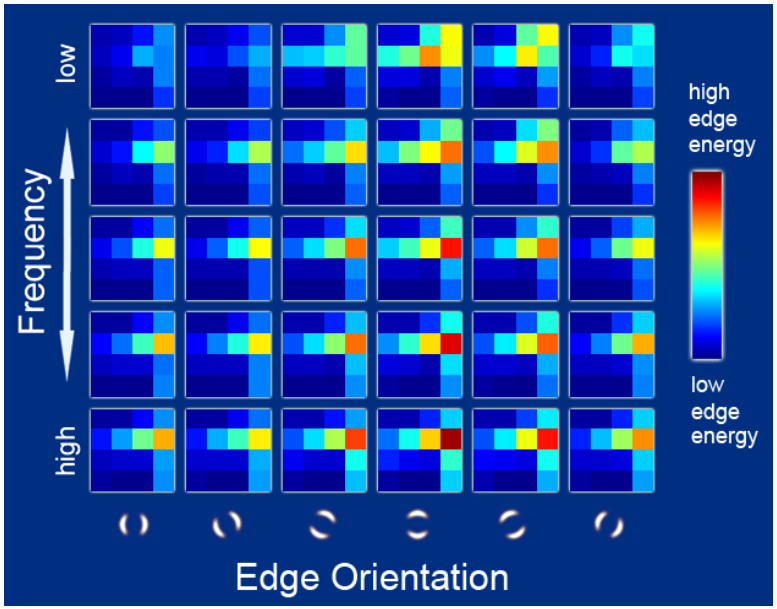
Related work on very large scale image search

- Min-hash and geometrical min-hash [Chum et al. 07-09]
- Compressing the BoF representation (miniBof) [Jegou et al. 09]
  → require hundreds of bytes to obtain a “reasonable quality”

- GIST descriptors with Spectral Hashing [Weiss et al.’08]
  → very limited invariance to scale/rotation/crop

Global scene context – GIST descriptor + spectral hashing

- The “GIST” of a scene: Oliva & Torralba (2001)



- 5 frequency bands and 6 orientations for each image location
- Tiling of the image (windowing)
- ~ 900 dimensions

- Spectral hashing produces binary codes, similar to spectral clustering

Related work on very large scale image search

- Min-hash and geometrical min-hash [Chum et al. 07-09]
- Compressing the BoF representation (miniBof) [Jegou et al. 09]
  → require hundreds of bytes to obtain a “reasonable quality”

- GIST descriptors with Spectral Hashing [Weiss et al.’08]
  → very limited invariance to scale/rotation/crop

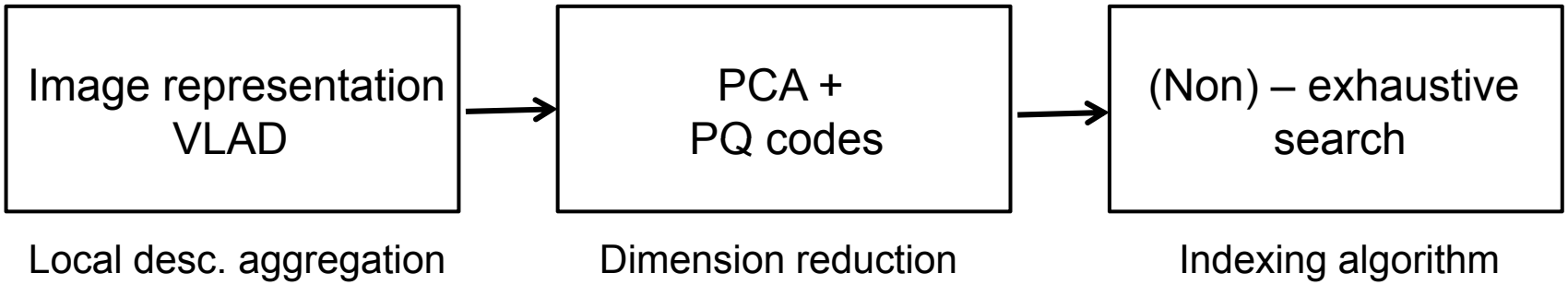
- Efficient object category recognition using classemes [Torresani et al.’10]

- Aggregating local descriptors into a compact image representation [Jegou&al.’10,‘12]

Aggregating local descriptors into a 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



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

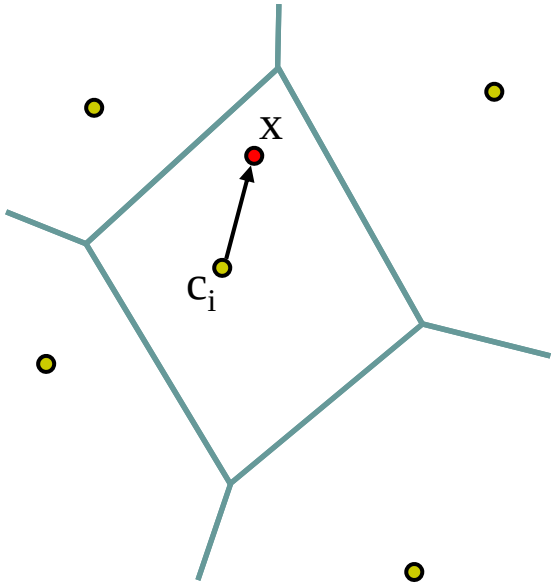
→ significant dimensionality reduction introduces loss

- Alternative: VLAD descriptor [VLAD = vector of locally aggregated descriptors]
  - non sparse vectors
  - excellent results with low dimensional vectors

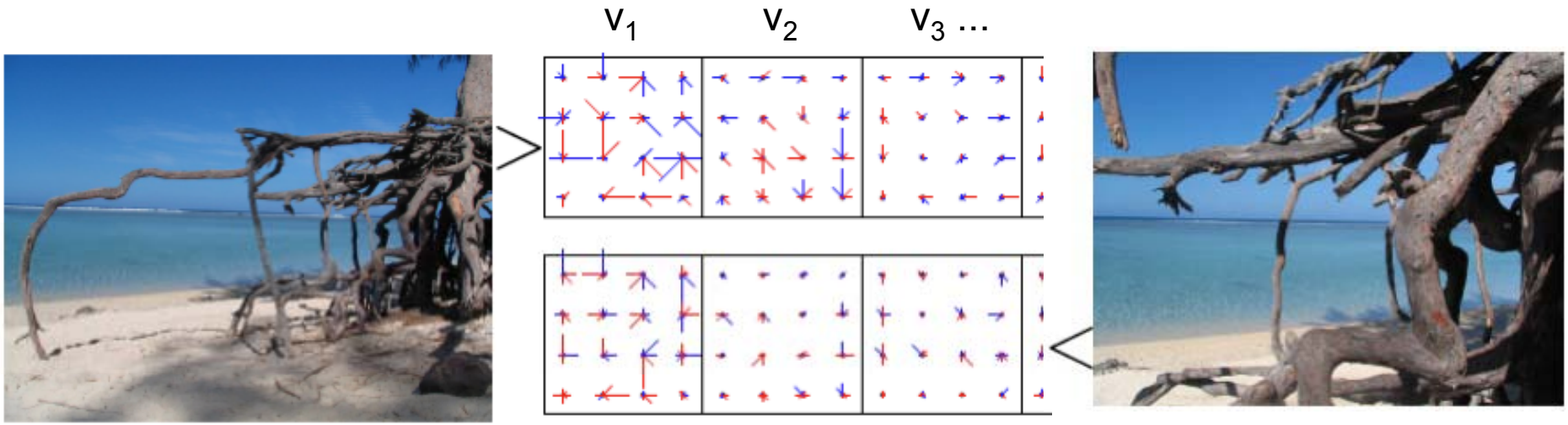
VLAD : vector of locally aggregated descriptors

- 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)

VLAD performance and dimensionality reduction

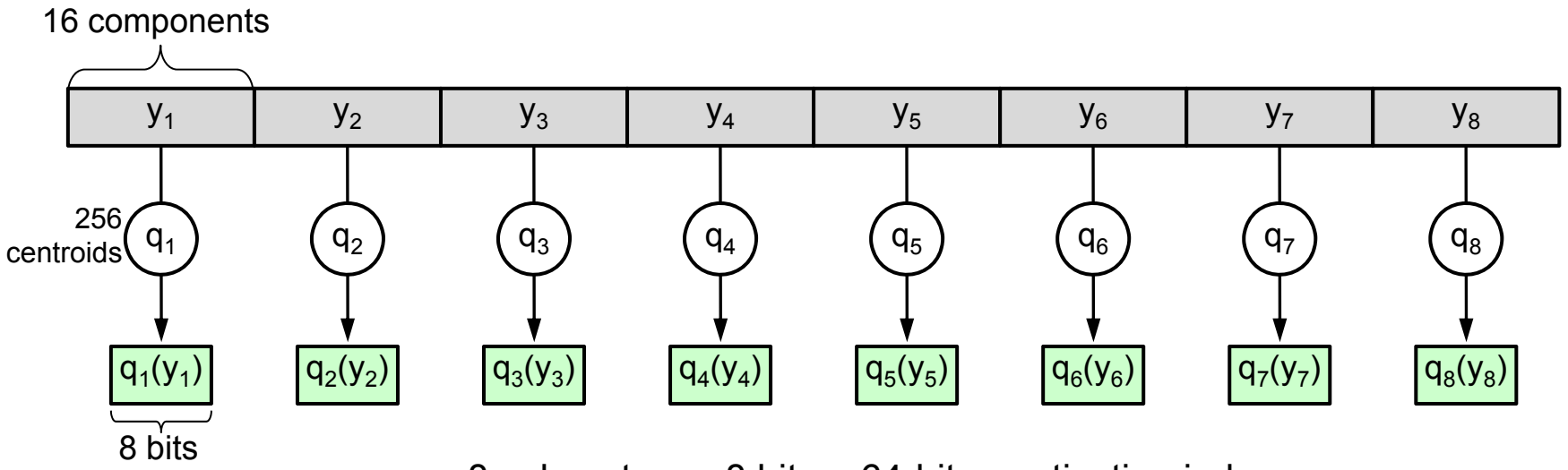
- We compare VLAD descriptors with BoF: INRIA Holidays Dataset (mAP,%)
- Dimension is reduced to from D to D' dimensions with PCA

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
  - 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

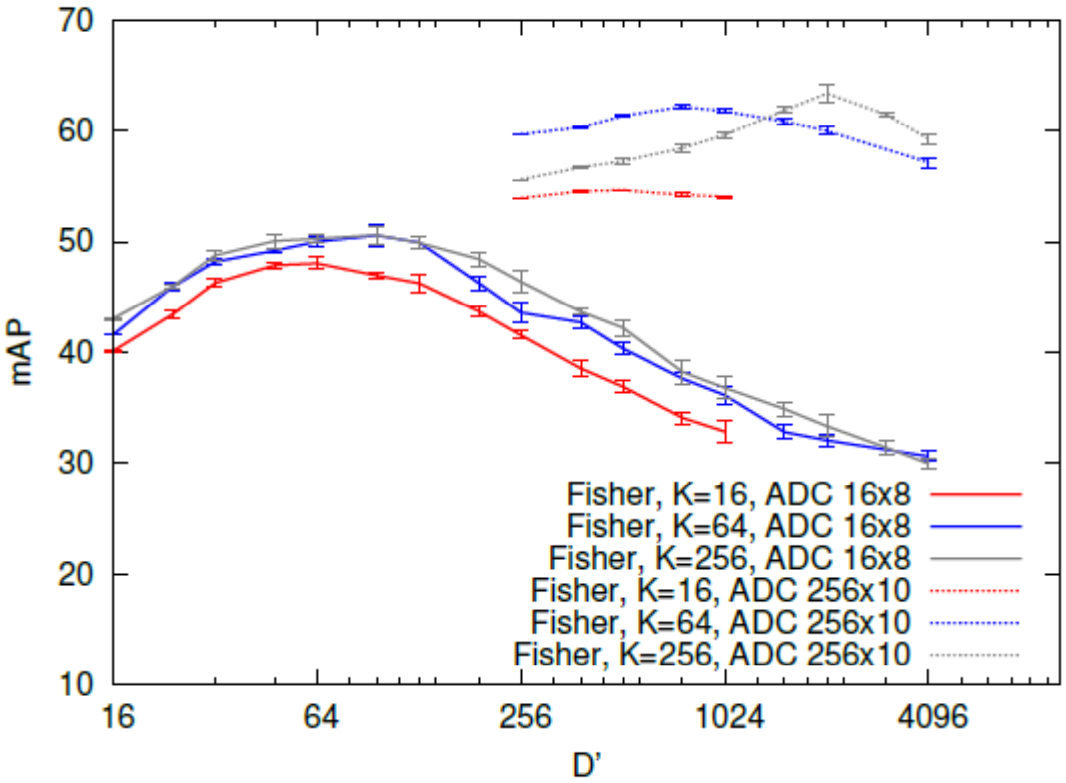


⇒ 8 subvectors x 8 bits = 64-bit quantization index

Optimizing the dimension reduction and quantization together

- VLAD vectors undergoes two approximations
  - mean square error from PCA projection
  - mean square error from quantization

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



Results on Holidays dataset:
- there exists an optimal D'
- 16 byte best results for k=64
- 320 byte best results for k=256

ADC = asymmetric distance computation

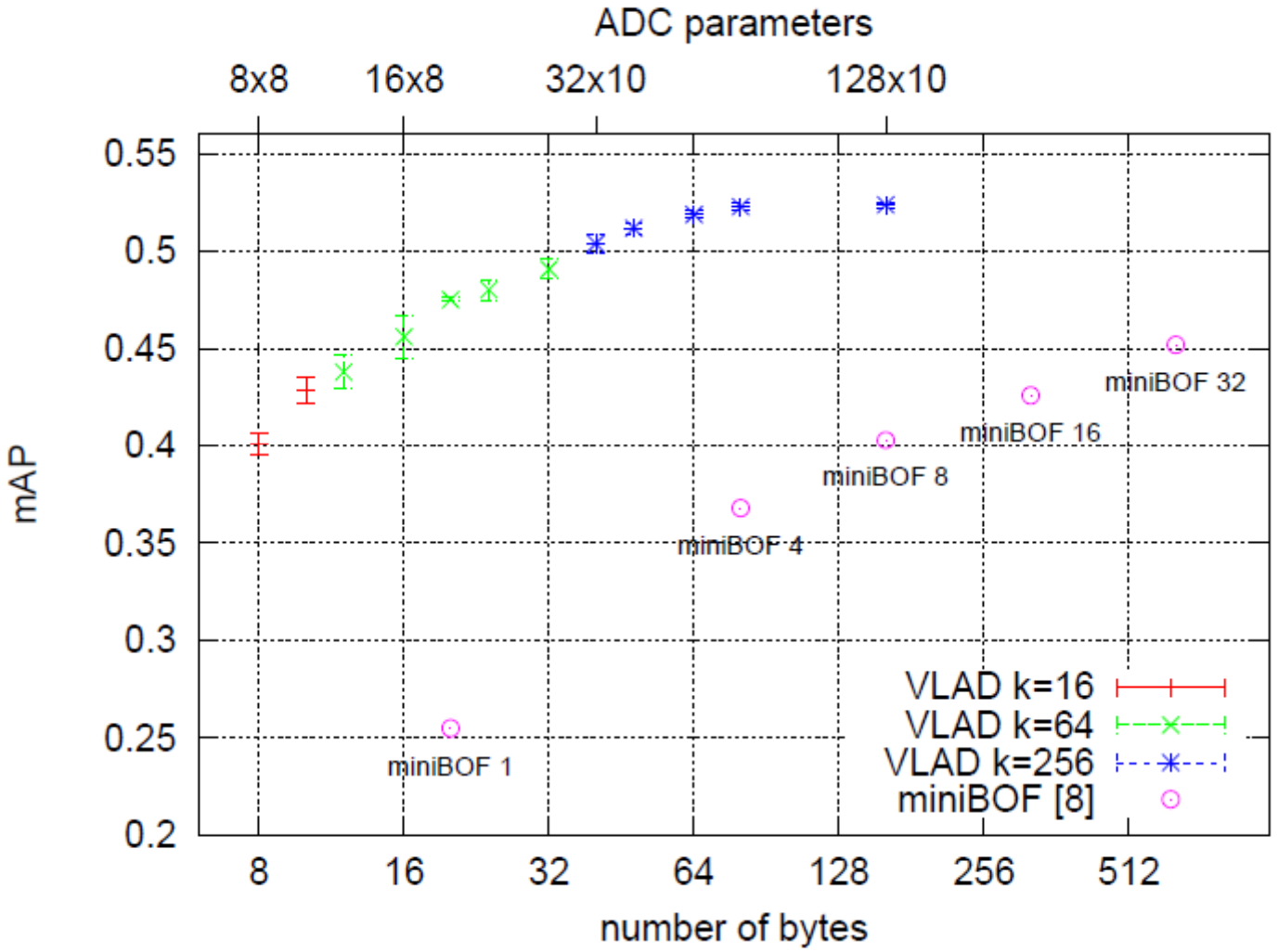
Joint optimization of VLAD and dimension reduction-indexing

- 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

➔ 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)

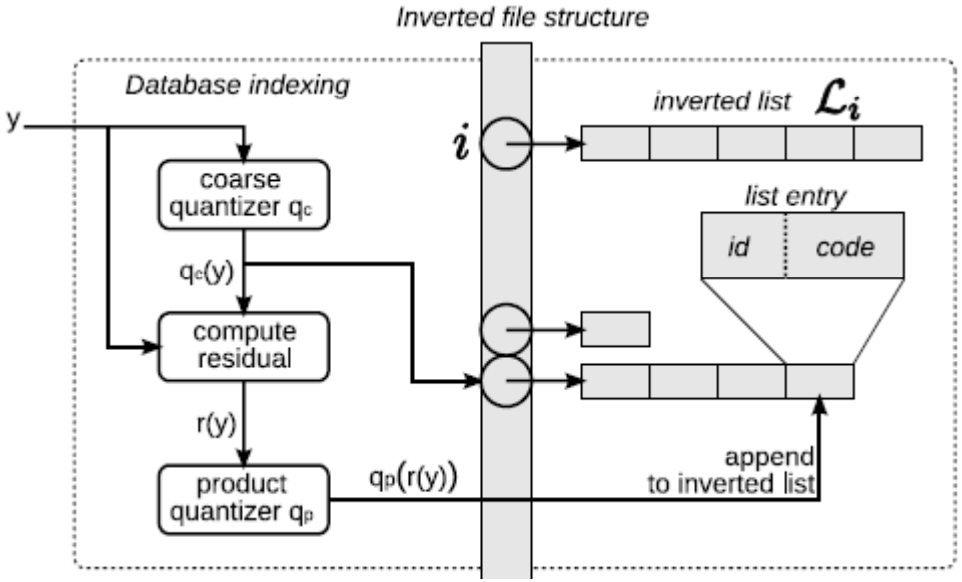
miniBOF: "Packing Bag-of-Features", ICCV'09

Large scale experiments (10 million images)

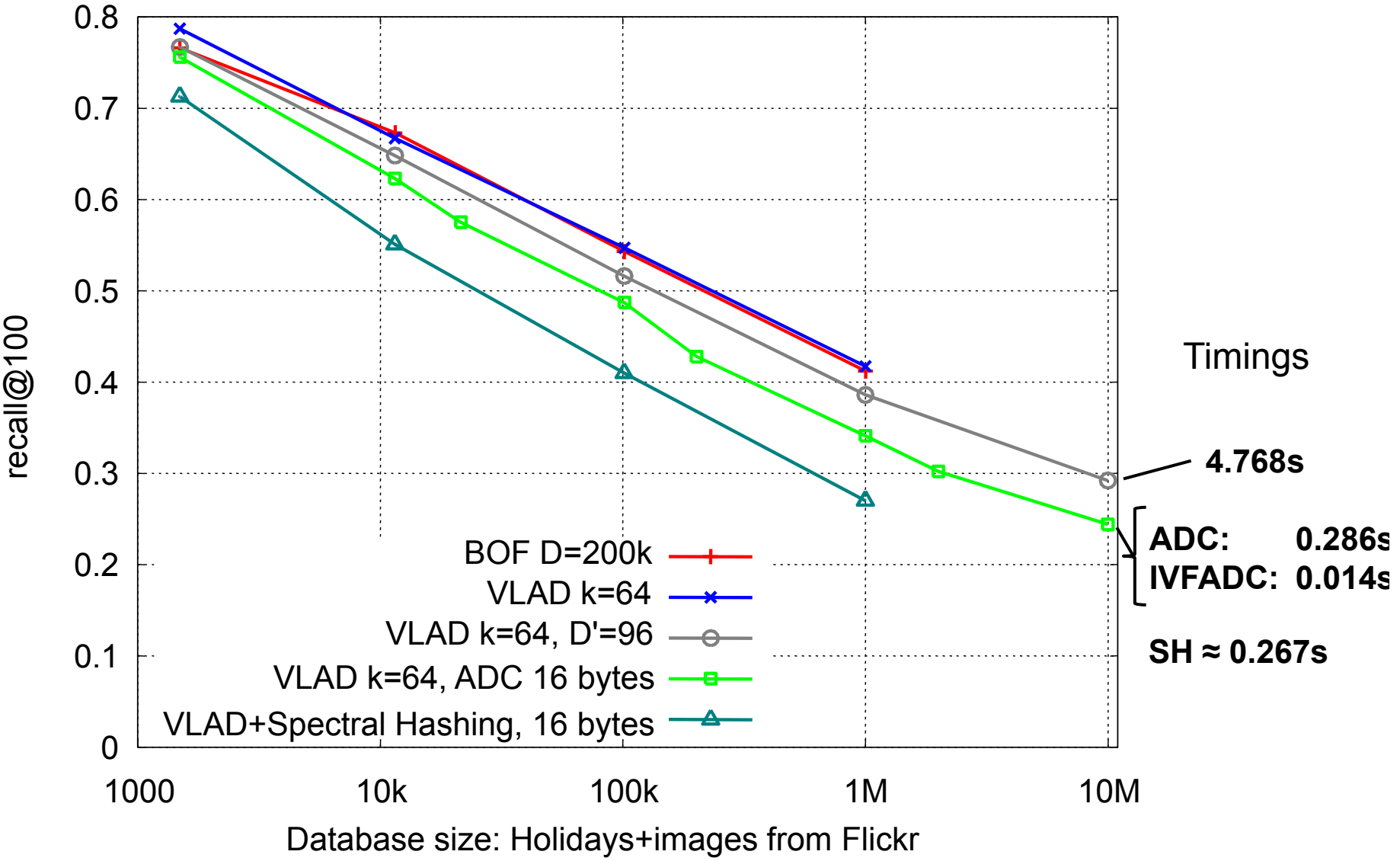
- Exhaustive search of VLADs, D'=64
  - 4.77s

- 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)

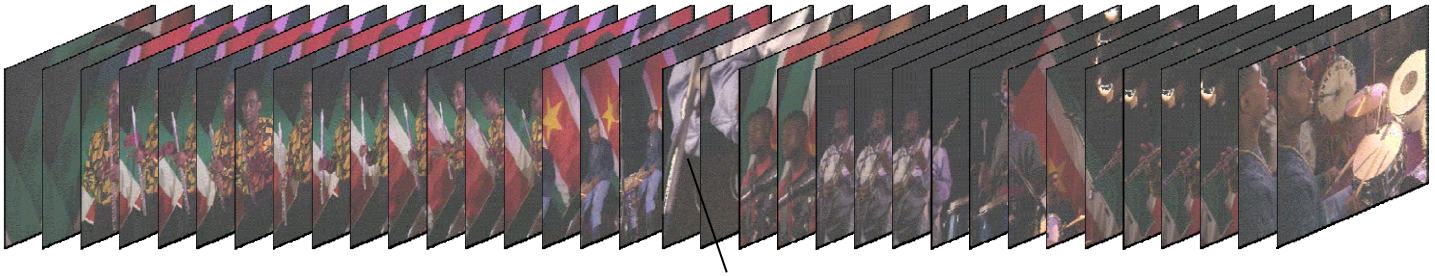


Conclusion & future work

- 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 frames
  - extrapolation for 1 billion images: 20GB RAM, query time < 1s on 8 cores
- On-line available: Matlab source code for product quantizer
- Alternative: using Fisher vectors instead of VLAD descriptors [Perronnin'10]
- Extension to video & more "semantic" search

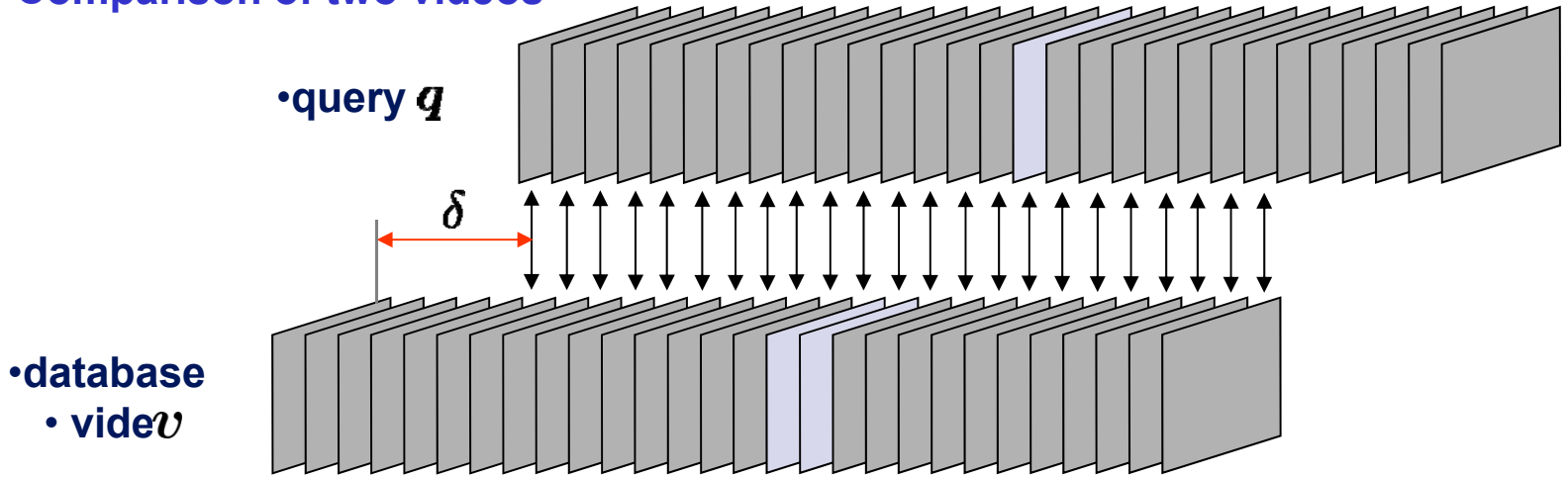
Event retrieval in large video collections [Revaud et al. 2013]

Video description



frame t → VLAD descriptor, reduced to 512D with PCA

Comparison of two videos



Fast calculation in the frequency domain + product quantization