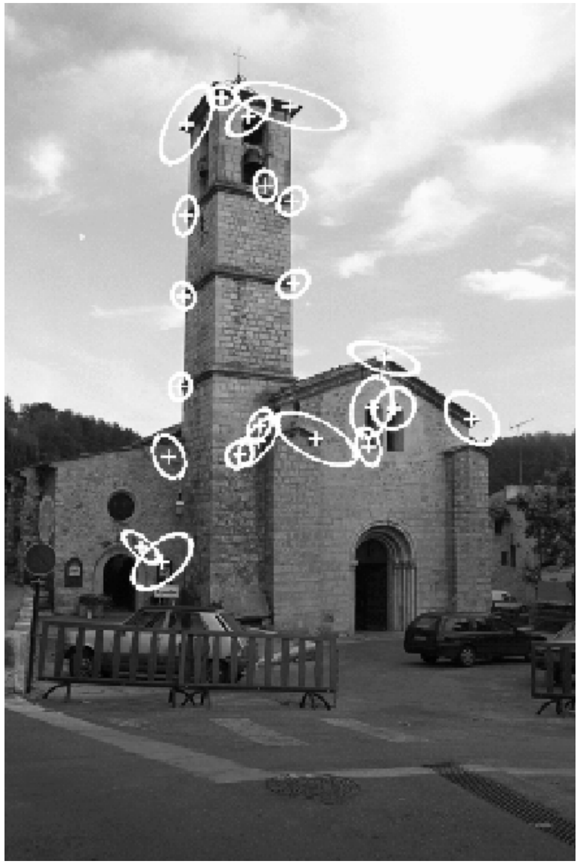
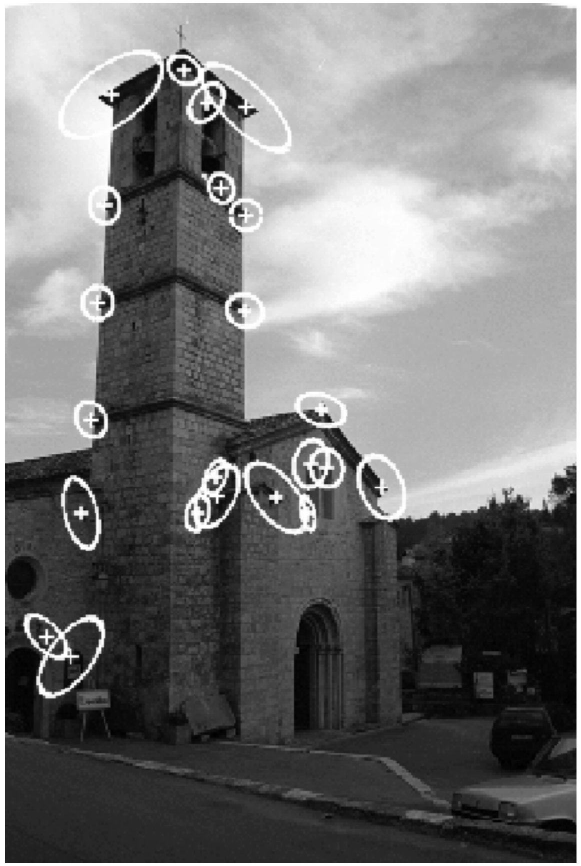
Efficient visual search of local features

Cordelia Schmid

Matching of descriptors





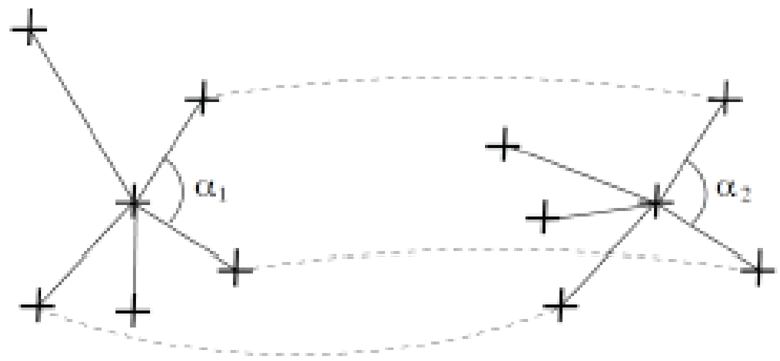
Matching of descriptors

- Find the nearest neighbor in the second image
- Pruning strategies
  - Ratio with respect to the second best match (d1/d2 << 1)

Matching of descriptors

- Find the nearest neighbor in the second image

- Pruning strategies
  - Ratio with respect to the second best match (d1/d2 << 1)
  - Local neighborhood constraints (semi-local constraints)



Neighbors of the point have to match and angles have to correspond.
Note that in practice not all neighbors have to be matched correctly.

Matching of descriptors

- Find the nearest neighbor in the second image

- Pruning strategies
  - Ratio with respect to the second best match (d1/d2 << 1)
  - Local neighborhood constraints (semi-local constraints)
  - Backwards matching (matches are NN in both directions)

Matching of descriptors

- Find the nearest neighbor in the second image

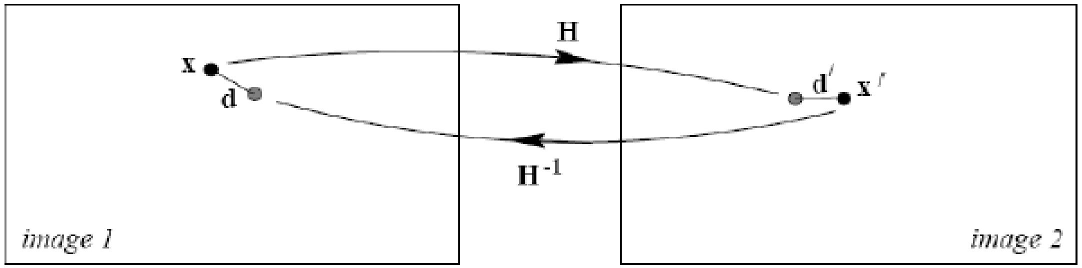
- Pruning strategies
  - Ratio with respect to the second best match (d1/d2 << 1)
  - Local neighborhood constraints (semi-local constraints)
  - Backwards matching (matches are NN in both directions)

- Geometric verification with global constraint
  - Hough transform [see for example Lowe'04, student presentation]
  - RANSAC (RANdom Sampling Consensus) [Fishler&Bolles'81]

Algorithm RANSAC

- Robust estimation with RANSAC of a homography
  - Repeat
    - Select 4 point matches
    - Compute 3x3 homography
    - Measure support (number of inliers within threshold, i.e. $d^2_{transfer} < t$)

$$d^2_{\text{transfer}} = d(\mathbf{x}, \mathtt{H}^{-1}\mathbf{x}')^2 + d(\mathbf{x}', \mathtt{H}\mathbf{x})^2$$



  - Choose (H with the largest number of inliers)
  - Re-estimate H with all inliers

Comparison

Hough Transform

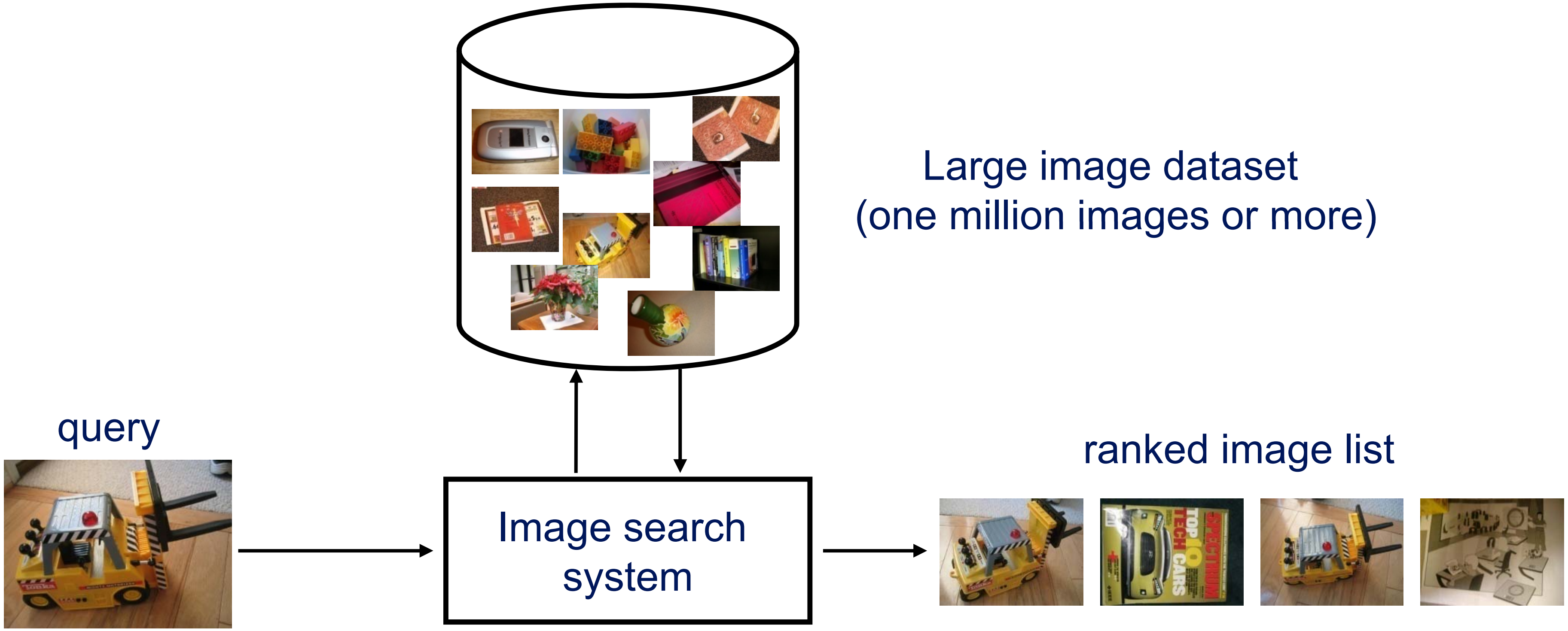
- Advantages
  - Can handle high percentage of outliers (>95%)
  - Extracts groupings from clutter in linear time
- Disadvantages
  - Quantization issues
  - Only practical for small number of dimensions (up to 4)
- Improvements available
  - Probabilistic Extensions
  - Continuous Voting Space
  - Can be generalized to arbitrary shapes and objects

RANSAC

- Advantages
  - General method suited to large range of problems
  - Easy to implement
  - “Independent” of number of dimensions
- Disadvantages
  - Basic version only handles moderate number of outliers (<50%)
- Many variants available, e.g.
  - PROSAC: Progressive RANSAC [Chum05]
  - Preemptive RANSAC [Nister05]

Visual search

Image search system for large datasets



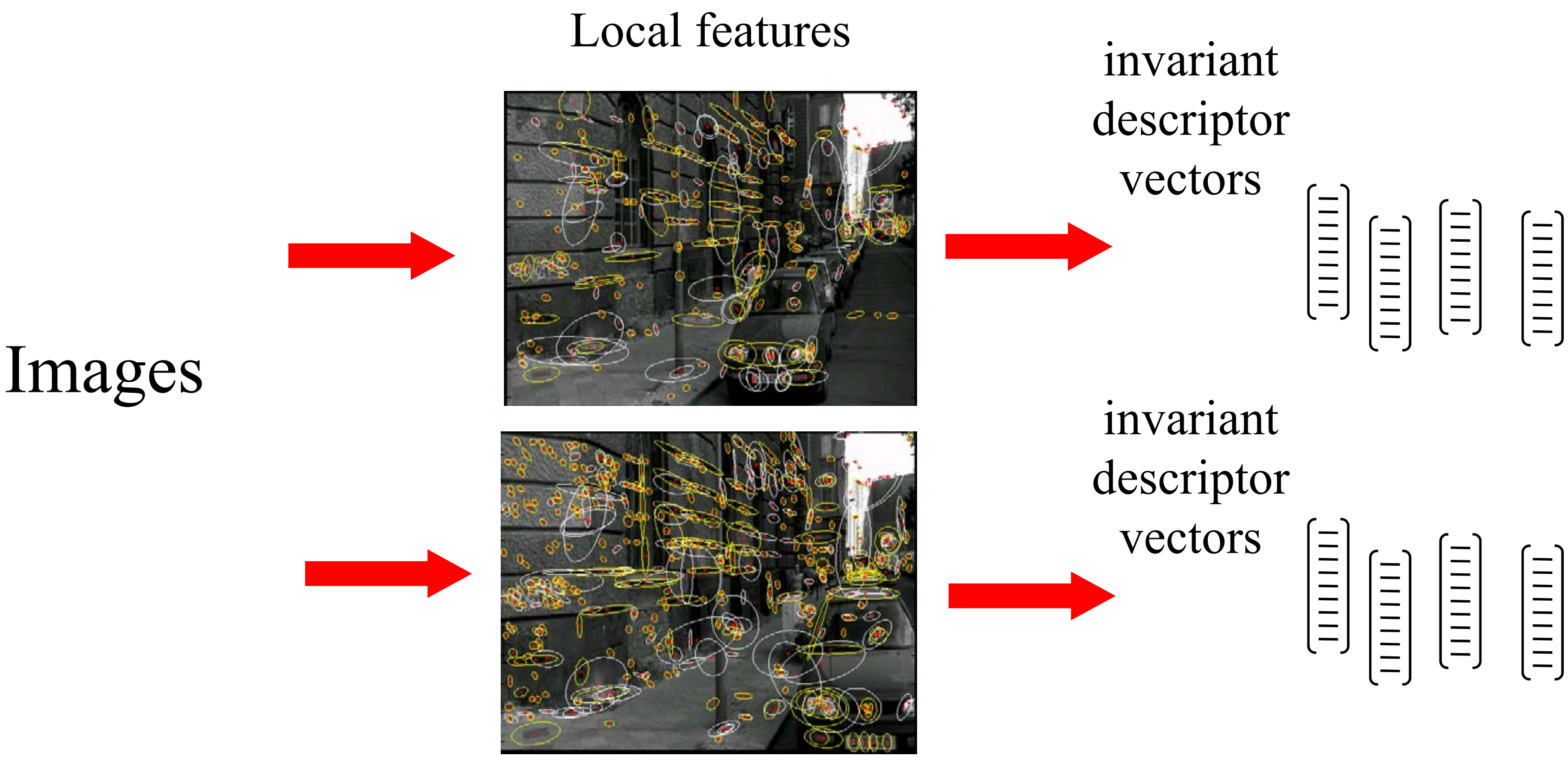
- **Issues** for very large databases
  - to reduce the query time
  - to reduce the storage requirements
  - with minimal loss in retrieval accuracy

Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors.

2. Quantize descriptors into a “visual vocabulary” and use efficient techniques from text retrieval
   (Bag-of-words representation)

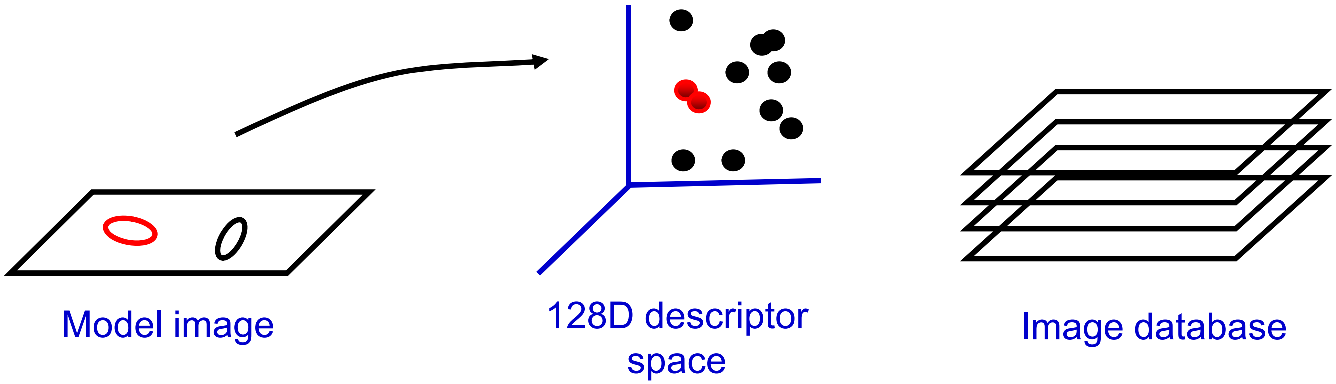
Strategy 1: Efficient approximate NN search



1. Compute local features in each image independently
2. Describe each feature by a descriptor vector
3. Find nearest neighbour vectors between query and database
4. Rank matched images by number of (tentatively) corresponding regions
5. Verify top ranked images based on spatial consistency

Finding nearest neighbour vectors

Establish correspondences between query image and images in the database by **nearest neighbour matching** on SIFT vectors



Solve following problem for all feature vectors, $\mathbf{x}_j \in \mathcal{R}^{128}$, in the query image:

$$\forall j \; NN(j) = \arg\min_i ||\mathbf{x}_i - \mathbf{x}_j||$$

where, $\mathbf{x}_i \in \mathcal{R}^{128}$, are features from all the database images.

Quick look at the complexity of the NN-search

N … images

M … regions per image (~1000)

D … dimension of the descriptor (~128)

Exhaustive linear search: O(M NMD)

Example:
- Matching two images (N=1), each having 1000 SIFT descriptors
  Nearest neighbors search: 0.4 s (2 GHz CPU, implemenation in C)
- Memory footprint: 1000 * 128 = 128kB / image

# of images	CPU time	Memory req.
N = 1,000 …	~7min	(~100MB)
N = 10,000 …	~1h7min	(~ 1GB)
…		
N = 10^7	~115 days	(~ 1TB)
…		
All images on Facebook:		
N = 10^{10} …	~300 years	(~ 1PB)

Nearest-neighbor matching

Solve following problem for all feature vectors, $\mathbf{x_j}$, in the query image:

$$\forall j \; NN(j) = \arg\min_i ||\mathbf{x}_i - \mathbf{x}_j||$$

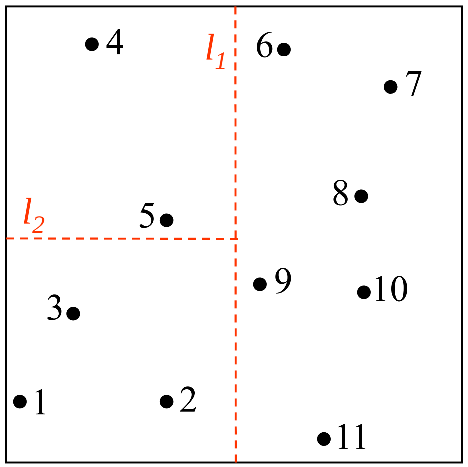
where $\mathbf{x_i}$ are features in database images.

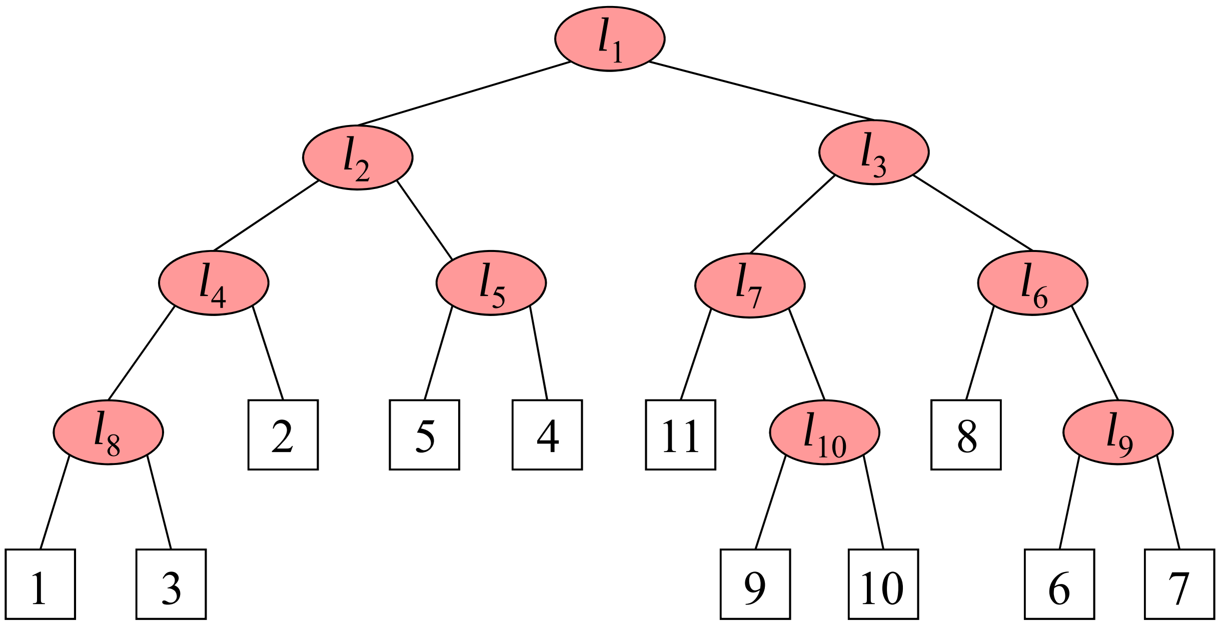
Nearest-neighbour matching is the major computational bottleneck

- Linear search performs *dn* operations for *n* features in the database and *d* dimensions
- No exact methods are faster than linear search for d>10
- Approximate methods can be much faster, but at the cost of missing some correct matches.

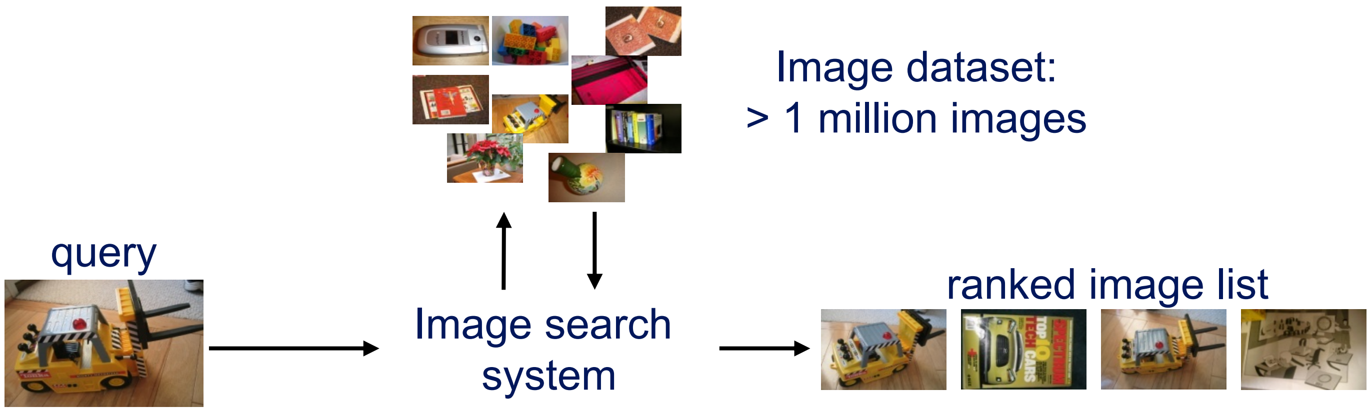
K-d tree

- K-d tree is a binary tree data structure for organizing a set of points
- Each internal node is associated with an axis aligned hyper-plane splitting its associated points into two sub-trees.
- Dimensions with high variance are chosen first.
- Position of the splitting hyper-plane is chosen as the mean/median of the projected points – balanced tree.





Large scale object/scene recognition



- Each image described by approximately 1000 descriptors
  - 10^9 descriptors to index for one million images!

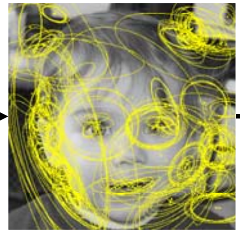
- Database representation in RAM:
  - Size of descriptors : 1 TB, search+memory intractable

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

Query image

Harris-Hessian-Laplace regions + SIFT descriptors

Set of SIFT descriptors



centroids (visual words)

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

sparse frequency vector

Inverted file

querying

ranked image short-list

Geometric verification

[Chum & al. 2007]

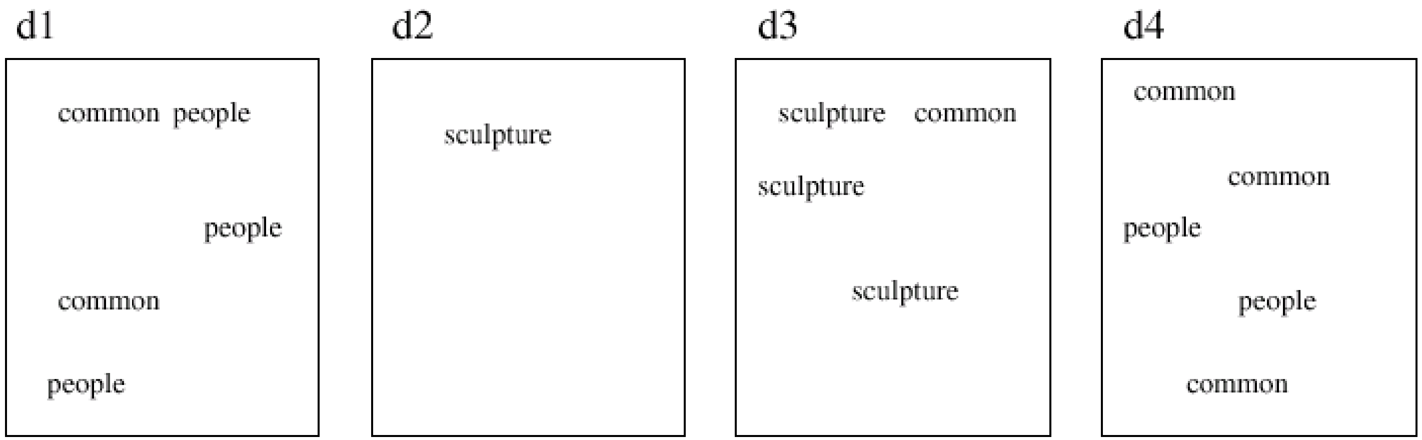
Re-ranked list

- "visual words":
  - 1 "word" (index) per local descriptor
  - only images ids in inverted file

  => 8 GB fits!

Indexing text with inverted files

Document collection:



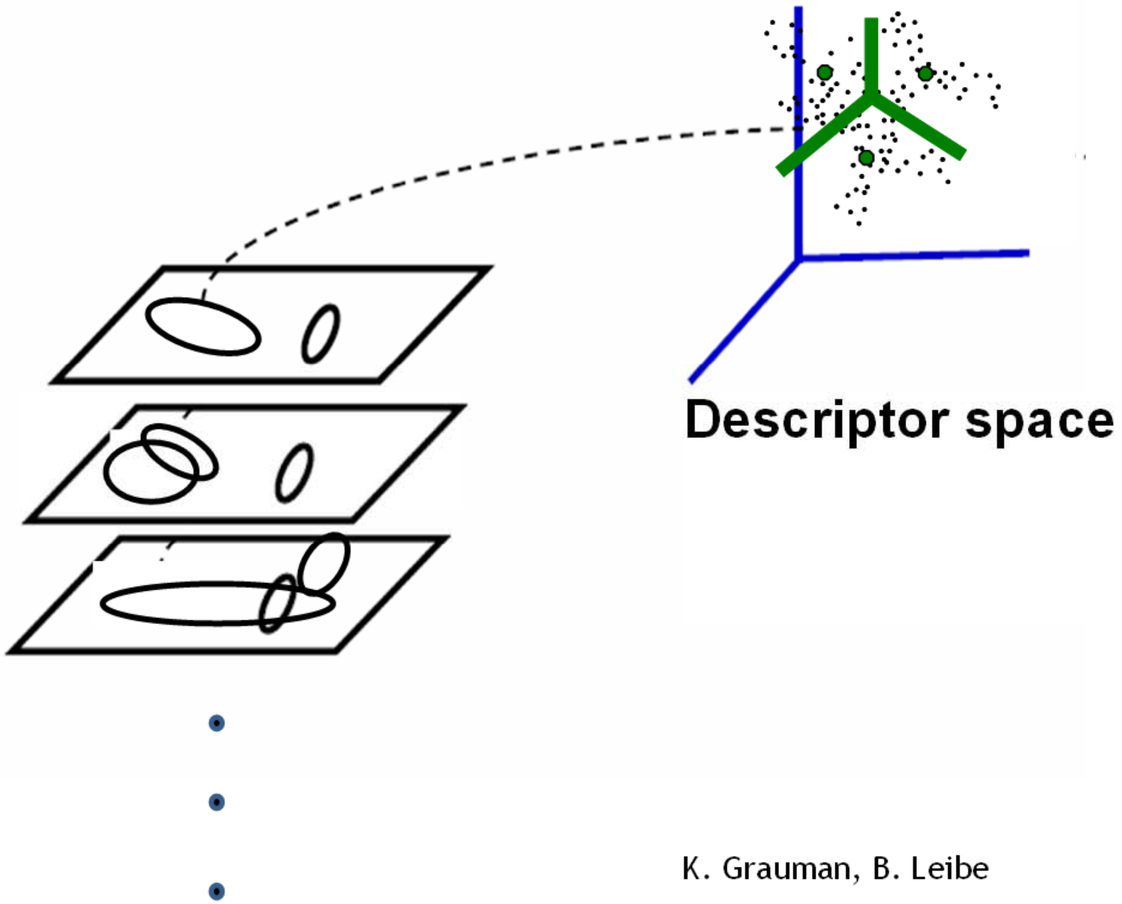
Inverted file:

Term	**List of hits** (occurrences in documents)
People	[d1:hit hit hit], [d4:hit hit] …
Common	[d1:hit hit], [d3: hit], [d4: hit hit hit] …
Sculpture	[d2:hit], [d3: hit hit hit] …

Need to map feature descriptors to “visual words”

Visual words: main idea

Map high-dimensional descriptors to tokens/words by quantizing the feature space

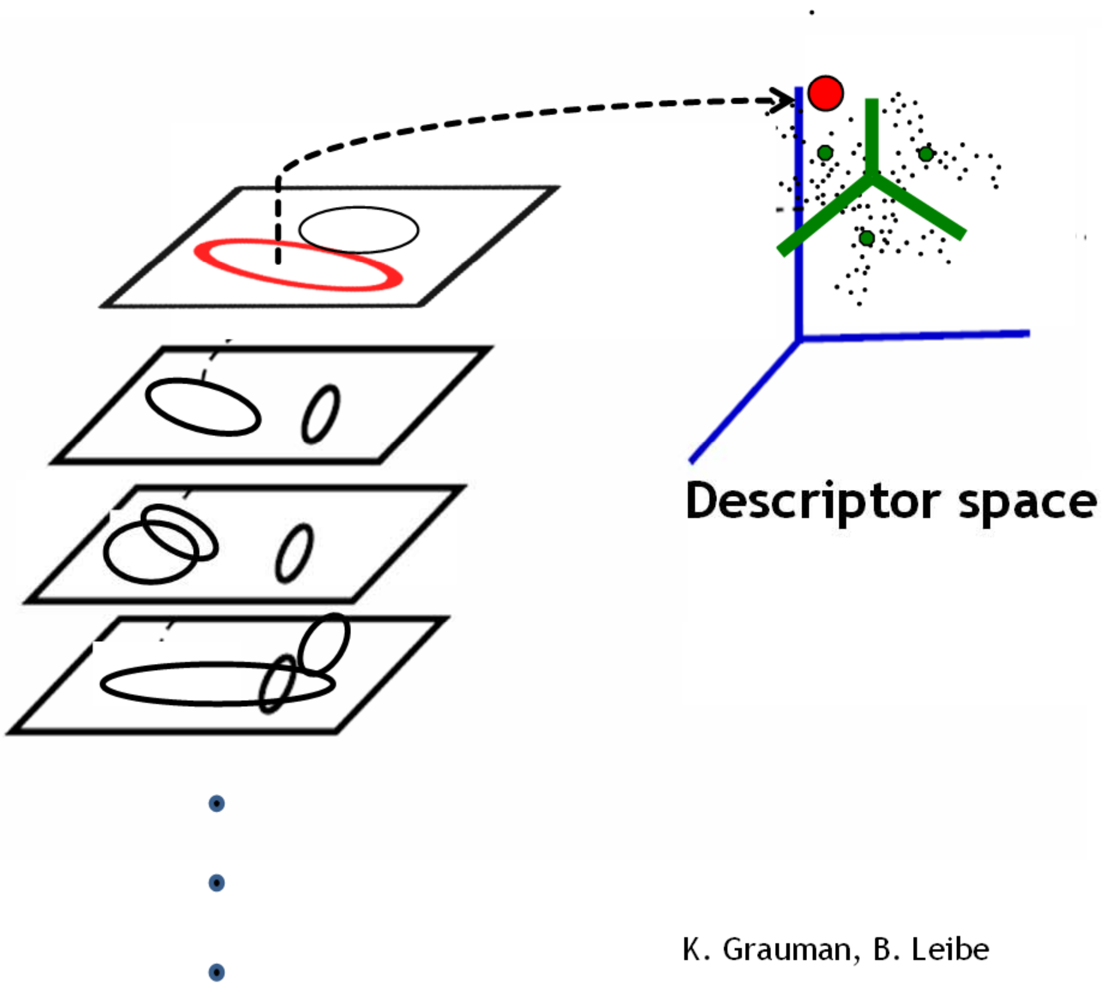


- Quantize via clustering, let cluster centers be the prototype "words"

K. Grauman, B. Leibe

Visual words: main idea

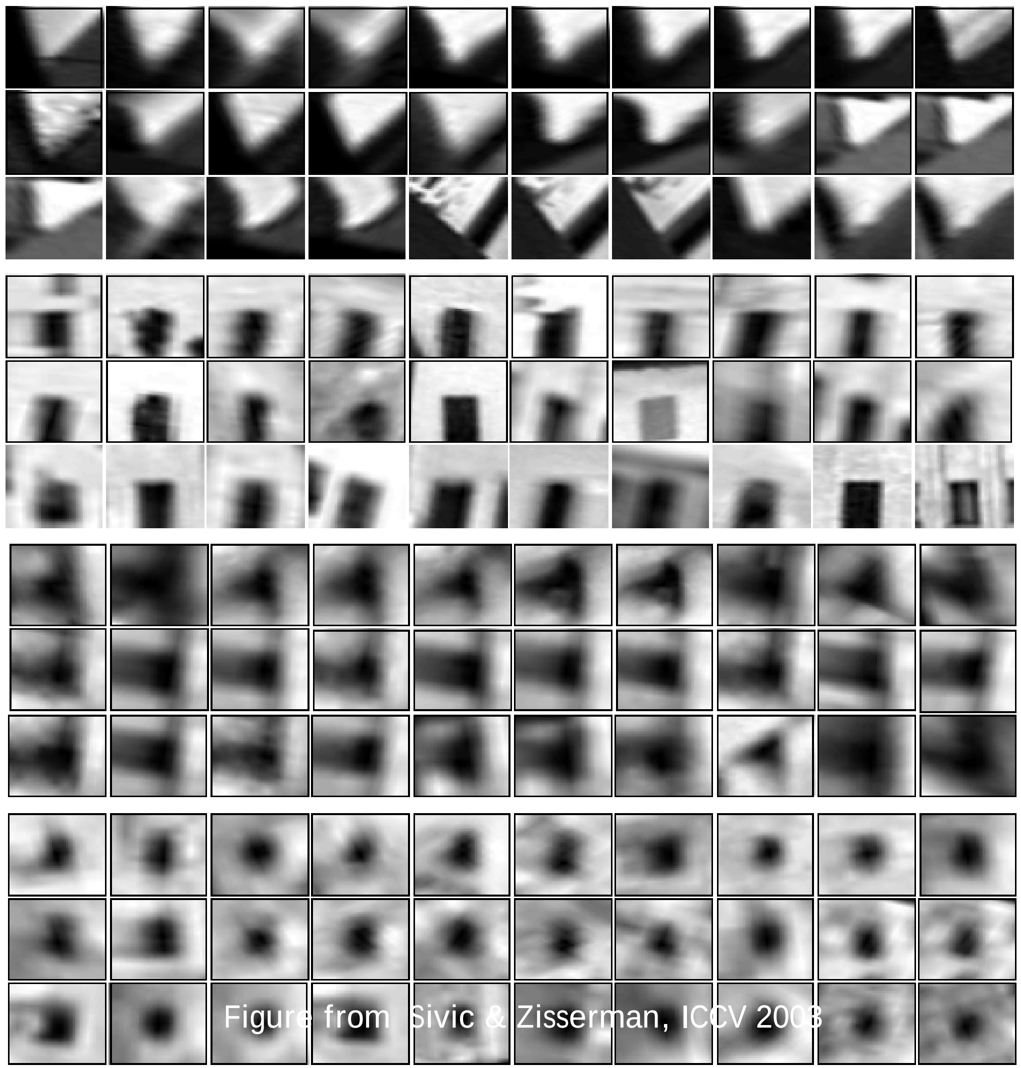
Map high-dimensional descriptors to tokens/words by quantizing the feature space



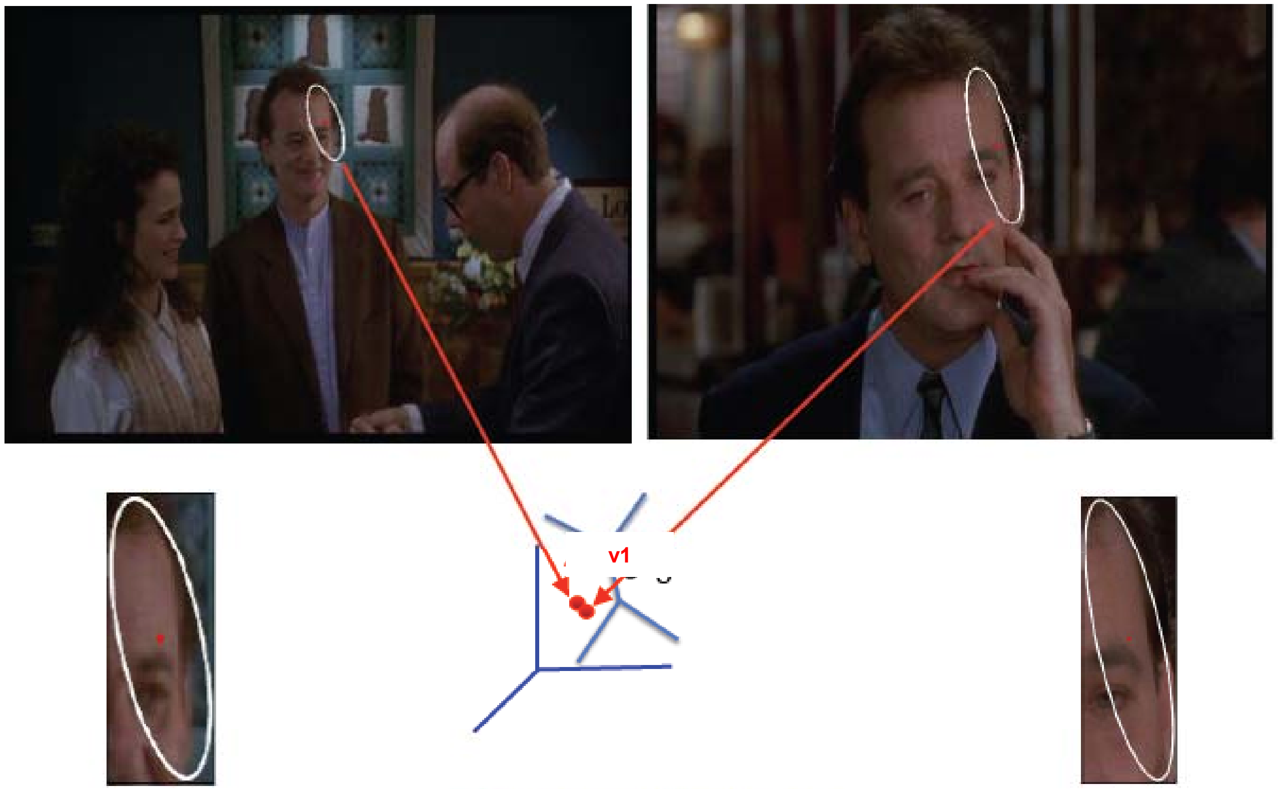
- Determine which word to assign to each new image region by finding the closest cluster center.

Visual words

•Example: each group of patches belongs to the same visual word

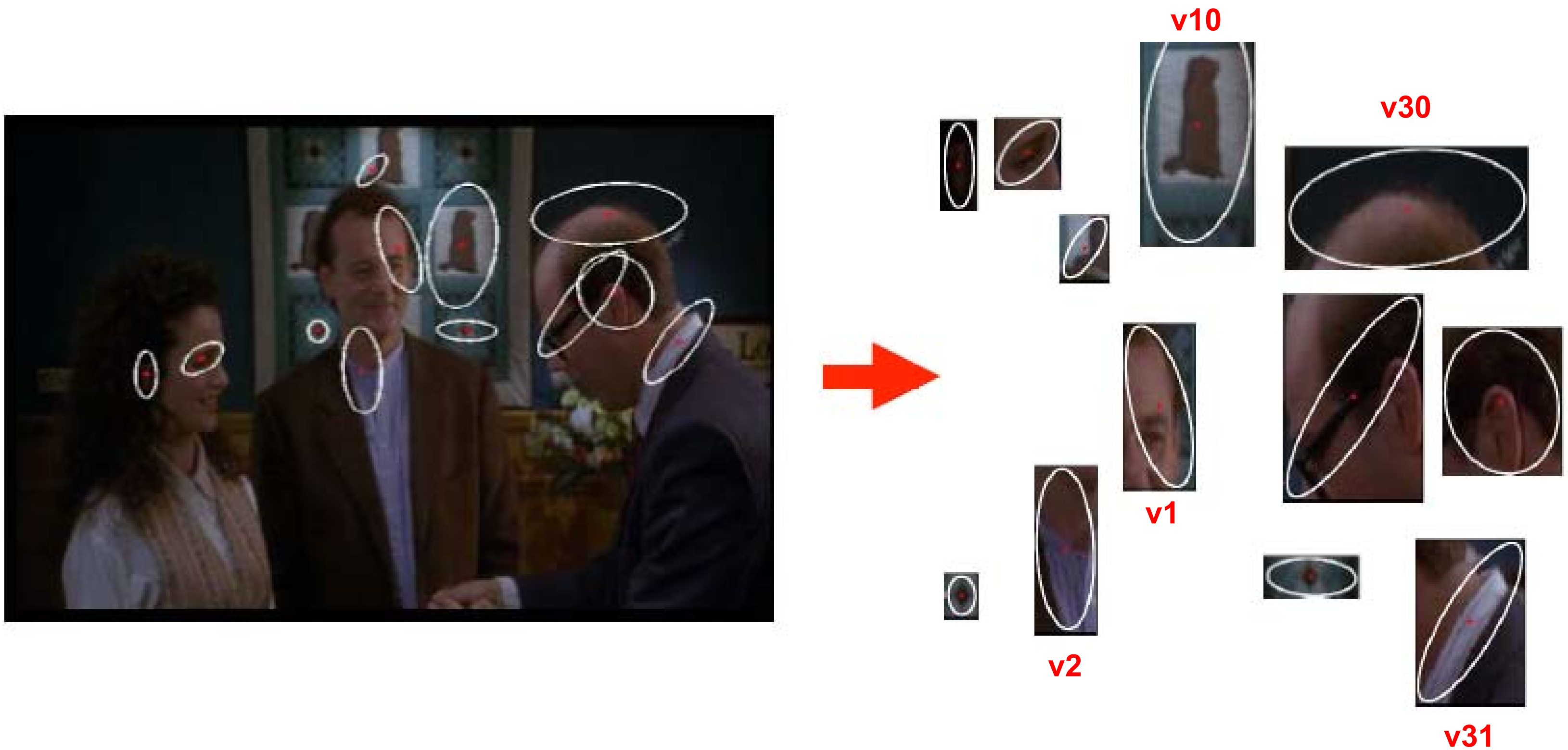


Vector quantize the descriptor space



The same visual word

Vector quantize the descriptor space



$$\begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \end{pmatrix}$$

- Histogram of visual word occurrence represents the image
- Sparse if large visual vocabulary

Inverted file index for visual words

frame #5

frame #10

Word number	List of image numbers
1	5,10, ...
2	10,...
...	...

- Score each image by the number of common visual words (tentative correspondences)
- Dot product between bag-of-features
- Fast for sparse vectors !

Inverted file index for visual words

frame #5

frame #10

Word number	List of image numbers
1	5,10, ...
2	10,...
...	...

- For fast search, store a posting list for the dataset
- This maps visual word occurrences to the images they occur in (like a book index)
- Increment a counter for each query descriptor

Inverted file index for visual words

frame #5

frame #10

Word number	List of image numbers
1	5,10, ...
2	10,...
...	...

- Worst case complexity is linear in the number of images
- In practice it is linear in the length of the list (<< N)
- Storage: one index per descriptor

Visual words – approximate NN search

- Map descriptors to words by quantizing the feature space
  - Quantize via k-means clustering to obtain visual words
  - Assign descriptors to closest visual words

- Bag-of-features as approximate nearest neighbor search

  Descriptor matching with *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 a 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

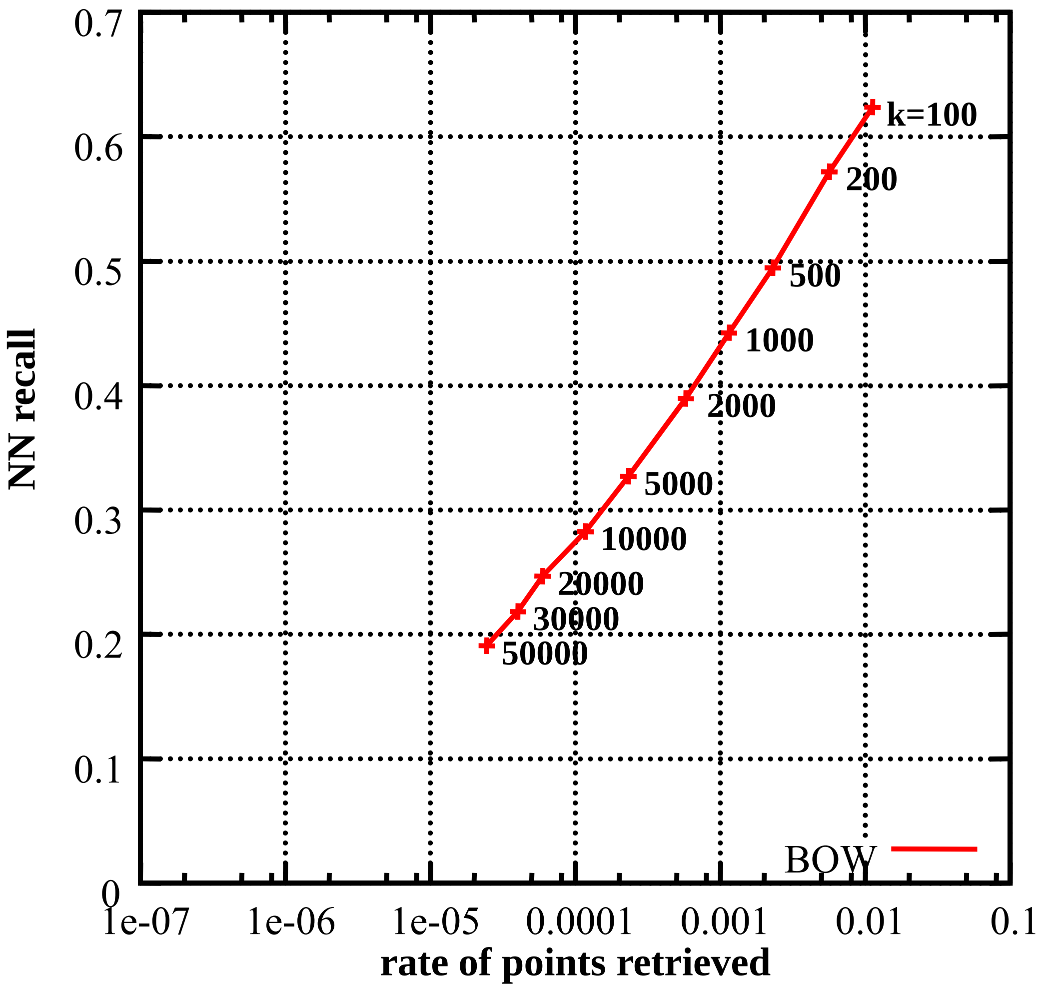
- Proposed quality evaluation of ANN search: trade-off between
  - **Accuracy**: **NN recall** = probability that *the* NN is in this list

    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



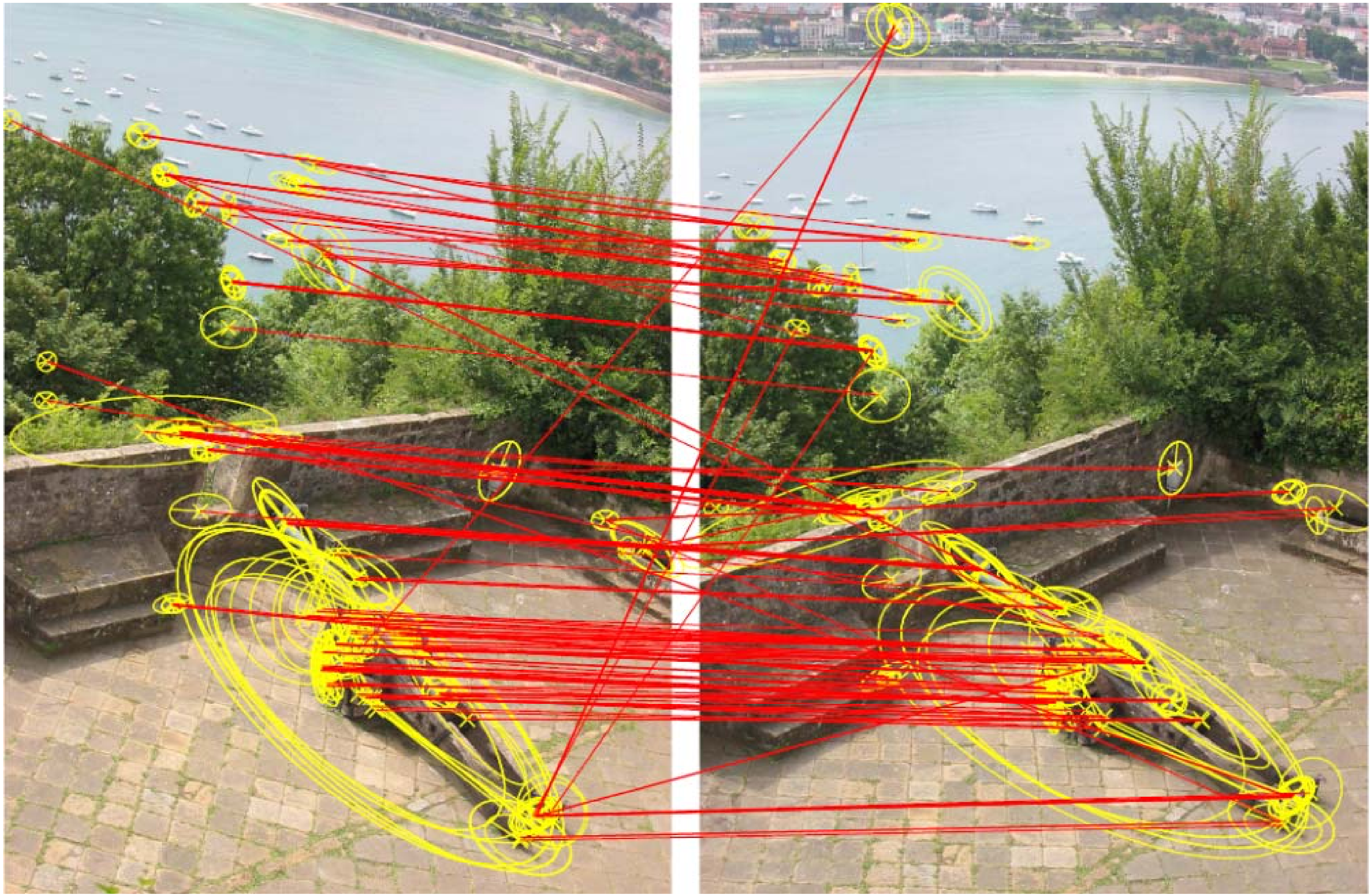
•ANN algorithms returns a list of potential neighbors

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

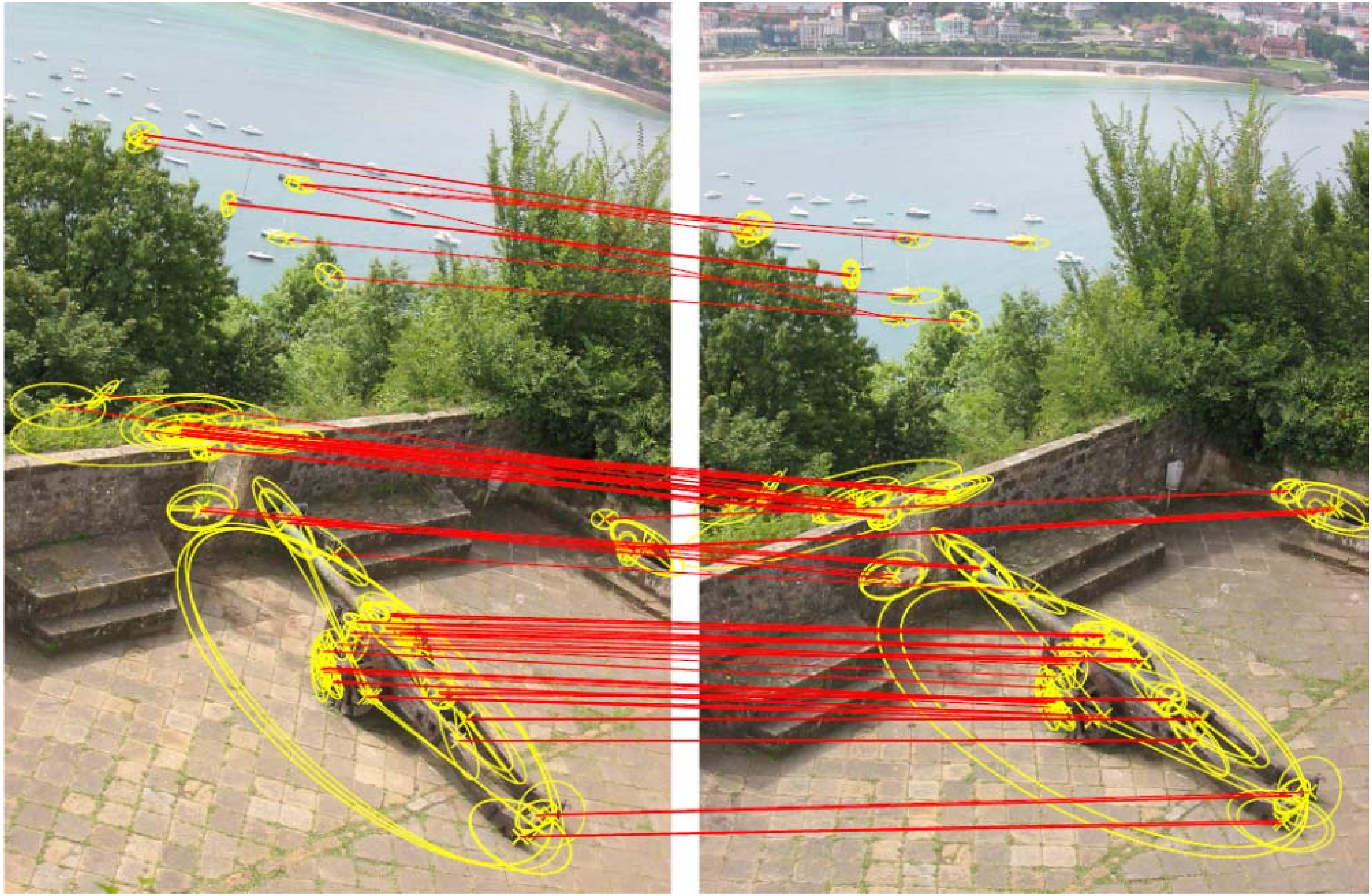
•**Ambiguity removal**: = proportion of vectors in the short-list

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

20K visual word: false matches



200K visual word: good matches missed



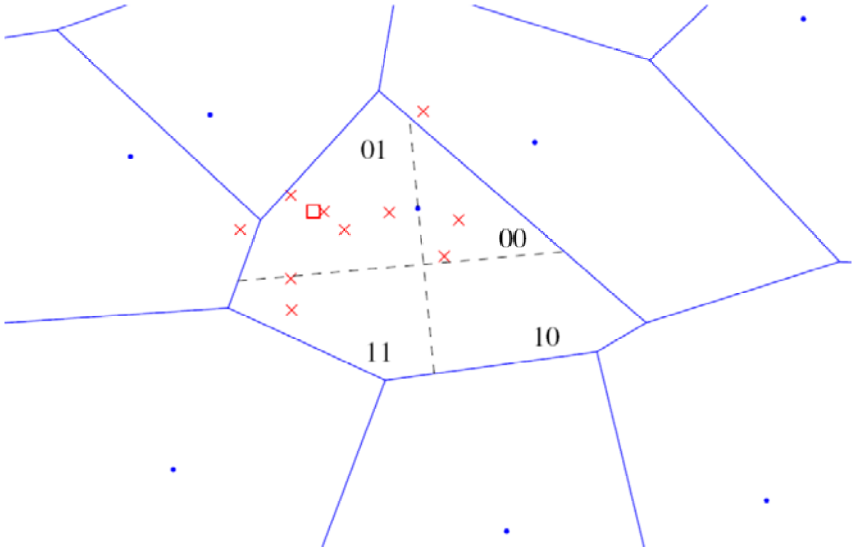
Problem with bag-of-features

- 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 [Jegou et al. ECCV'08]



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

$$f_{\mathrm{HE}}(x,y) = \begin{cases} (\text{tf-idf}(q(x)))^2 & \text{if } q(x) = q(y) \\ & \quad \text{and } h\left(b(x), b(y)\right) \leq h_t \\ 0 & \text{otherwise} \end{cases}$$

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

Hamming Embedding

- Nearest neighbors for Hamming distance $\approx$ those for Euclidean distance
$\rightarrow$ a metric in the embedded space reduces dimensionality curse effects

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

Hamming Embedding

•**Off-line** (given a quantizer)

- draw an orthogonal projection matrix *P* of size $d_b \times d$

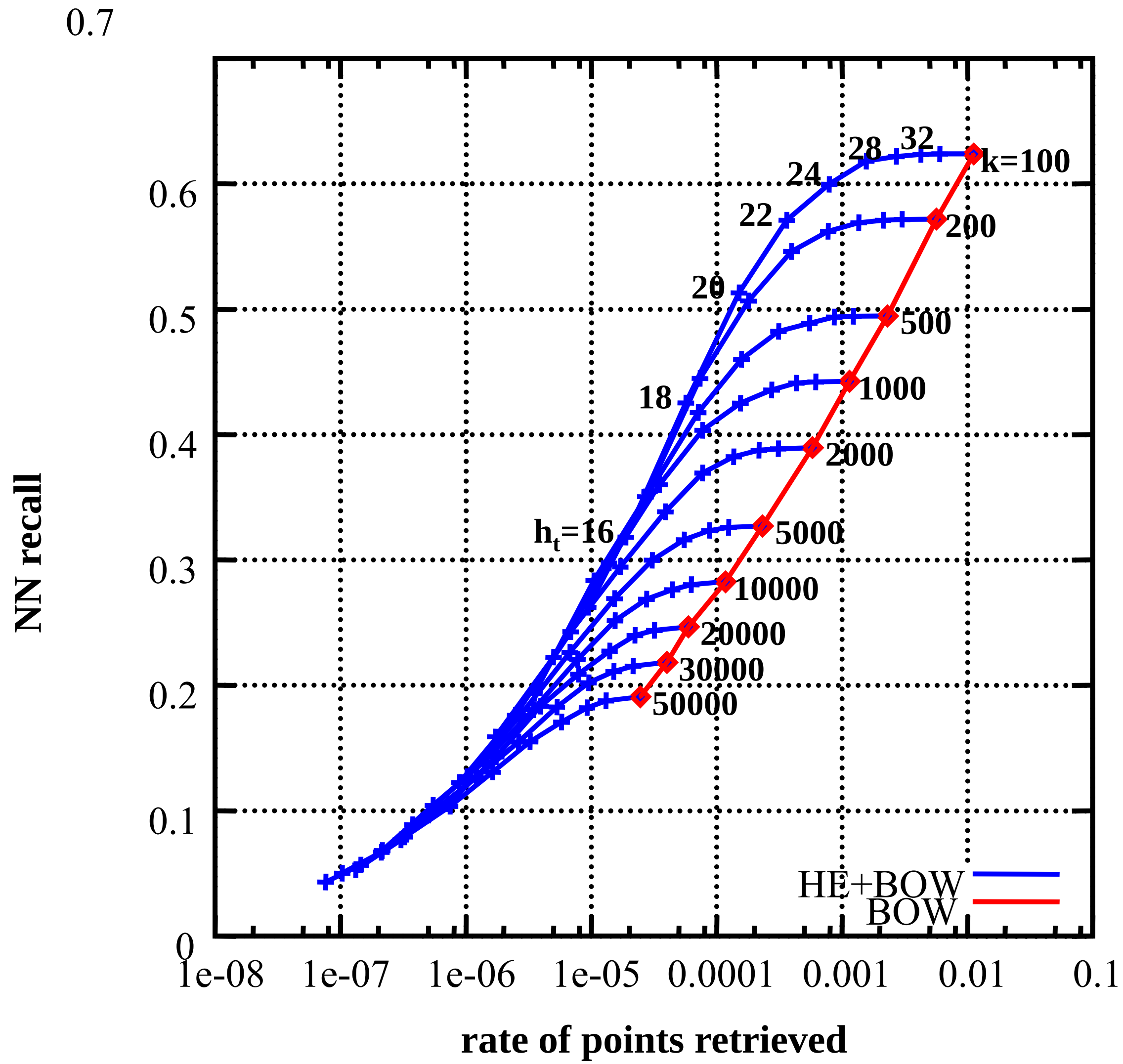
$\rightarrow$ this defines d_b random projection directions

- for each Voronoi cell and projection direction, compute the median value for 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

ANN evaluation of Hamming Embedding



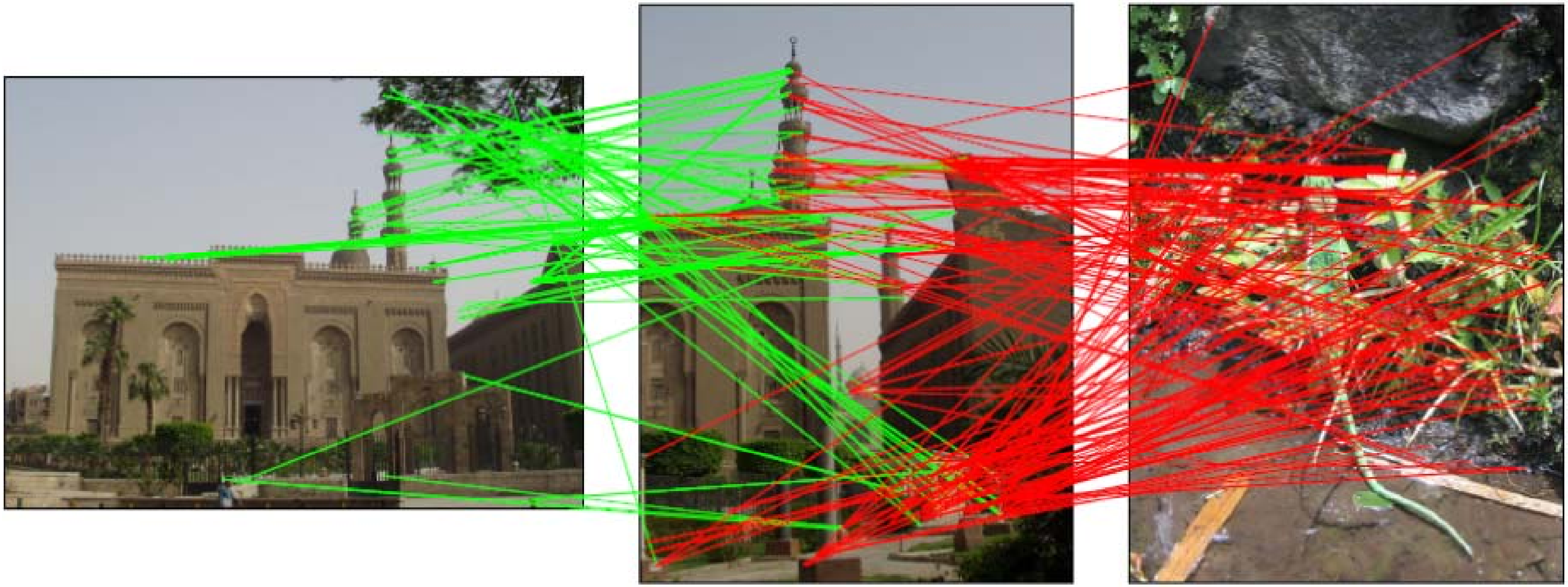
compared to BOW: at least 10 times less points in the short-list for the same level of accuracy

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

Matching points - 20k word vocabulary

201 matches

240 matches

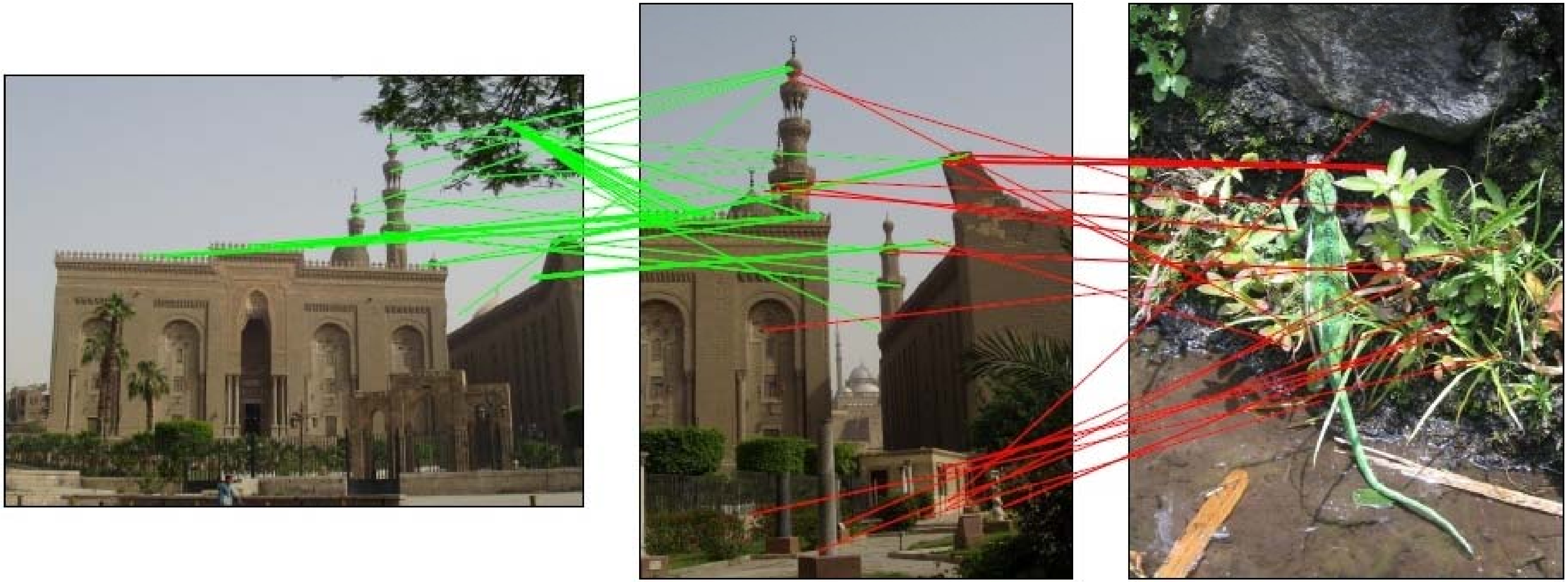


Many matches with the non-corresponding image!

Matching points - 200k word vocabulary

69 matches

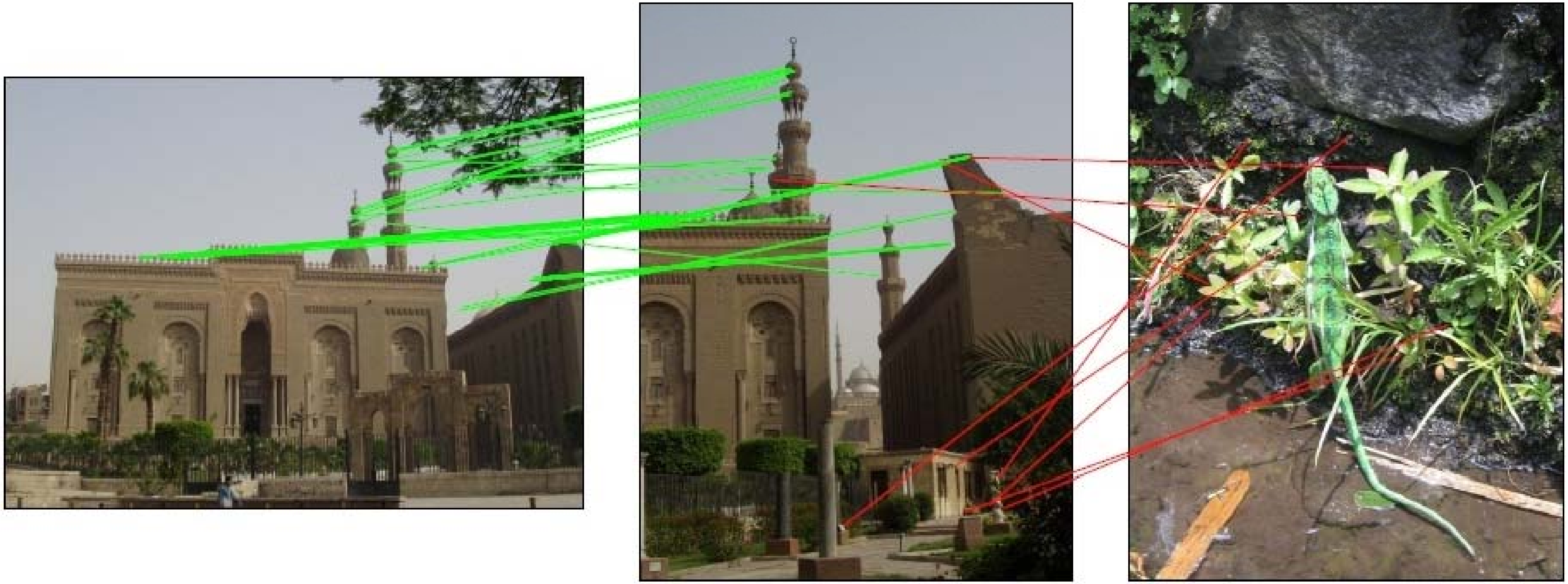
35 matches



Still many matches with the non-corresponding one

Matching points - 20k word vocabulary + HE

83 matches   8 matches



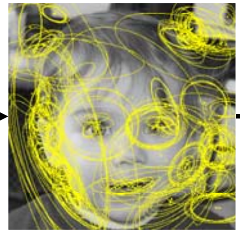
10x more matches with the corresponding image!

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

Query image

Harris-Hessian-Laplace regions + SIFT descriptors

Set of SIFT descriptors



centroids (visual words)

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

sparse frequency vector

Inverted file

querying

ranked image short-list

Geometric verification

Re-ranked list

[Chum & al. 2007]

- "visual words":
  - 1 "word" (index) per local descriptor
  - only images ids in inverted file

  => 8 GB fits!

Geometric verification

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

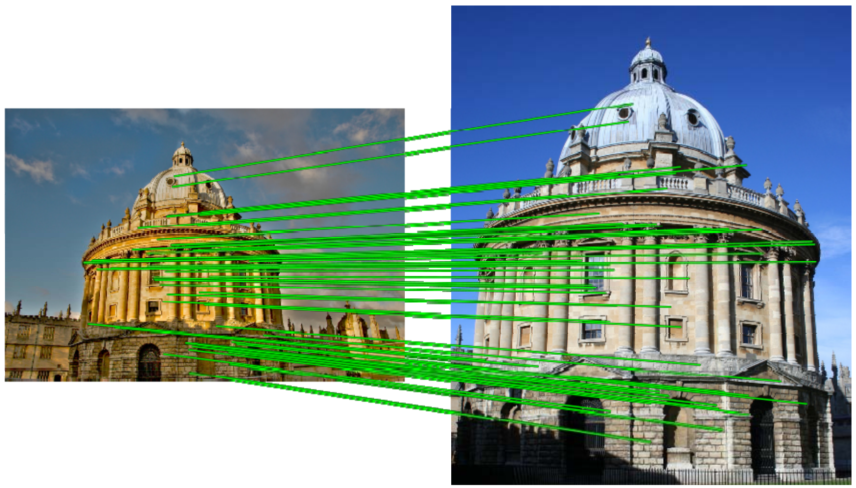
Both images have many matches – which is correct?

Geometric verification

- Remove outliers, matches contain a high number of incorrect ones
- Estimate geometric transformation
- Robust strategies
  - RANSAC
  - Hough transform

Geometric verification

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

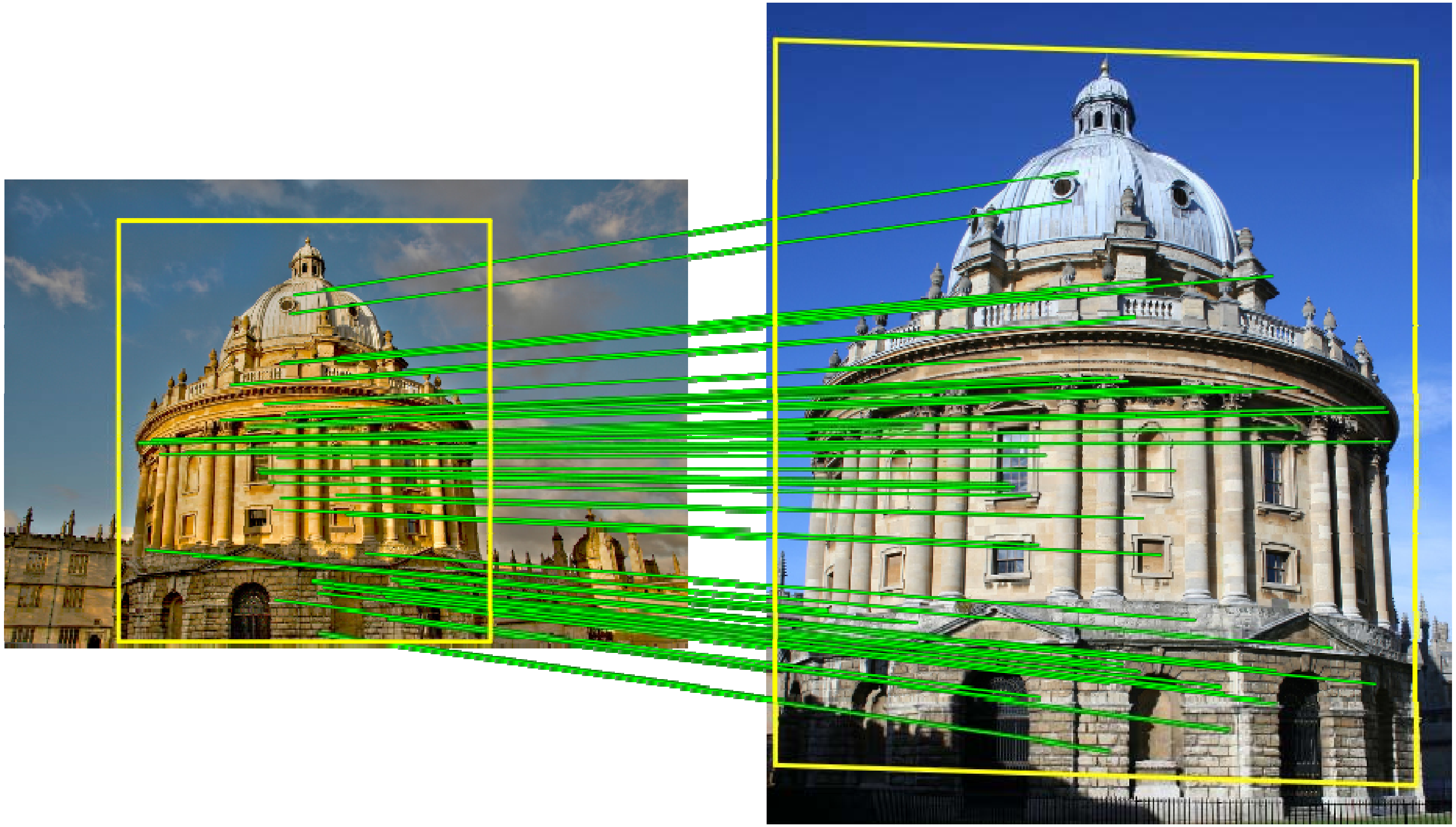


Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**

Geometric verification

Gives **localization** of the object

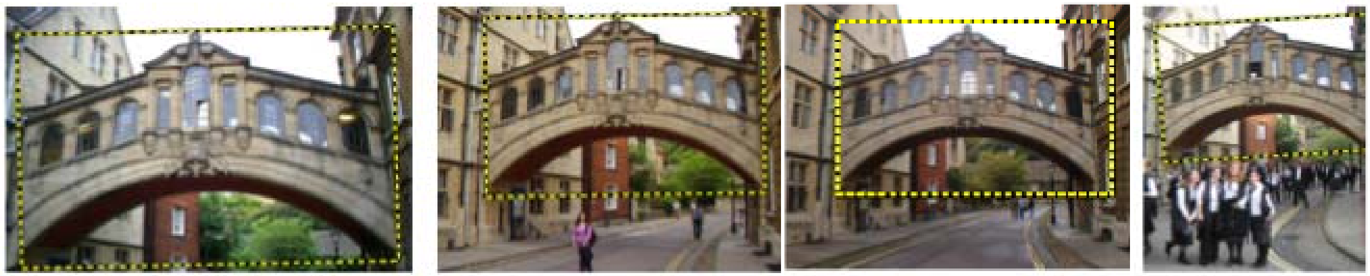


Geometric verification – example

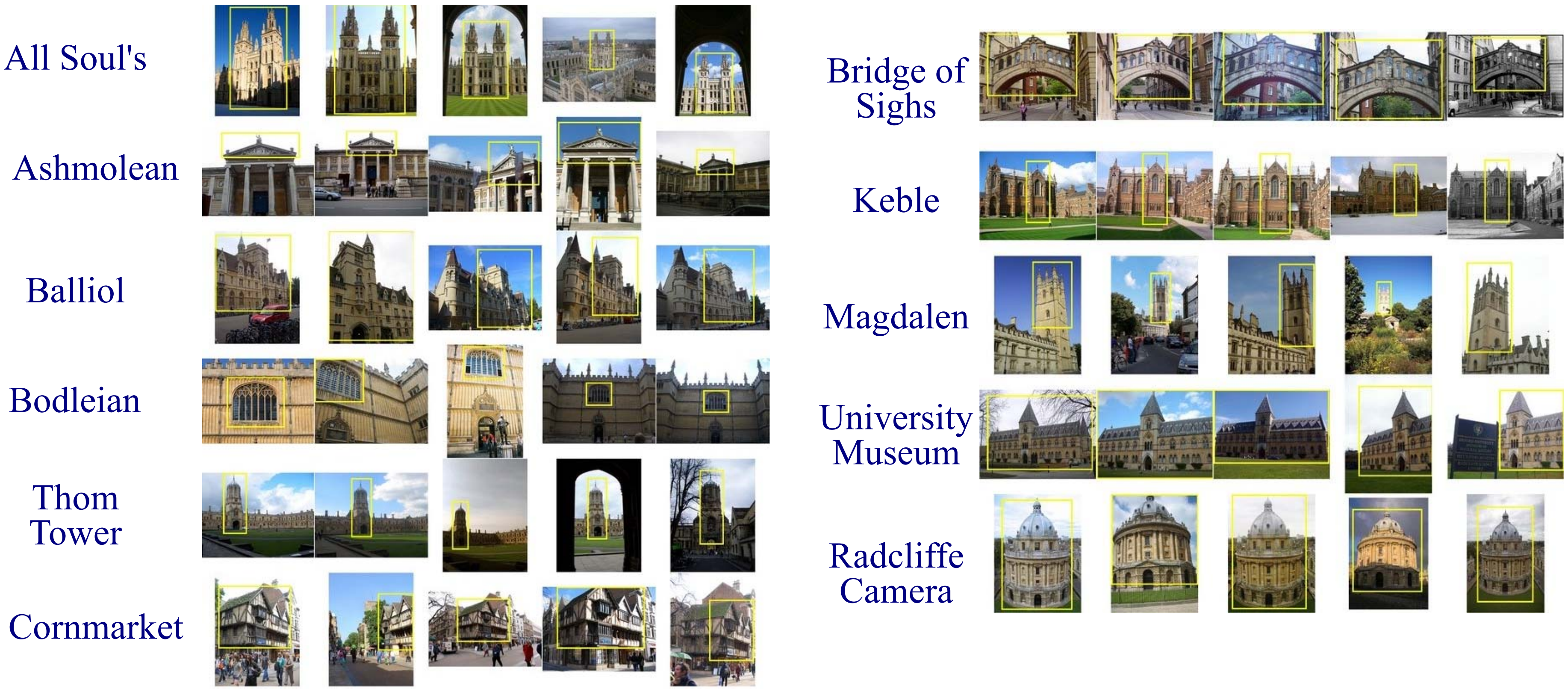
1. Query

2. Initial retrieval set (bag of words model)

3. Spatial verification (re-rank on # of inliers)



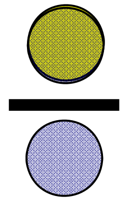
Evaluation dataset: Oxford buildings



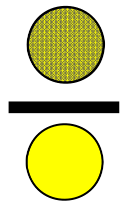
- Ground truth obtained for 11 landmarks
- Evaluate performance by mean Average Precision

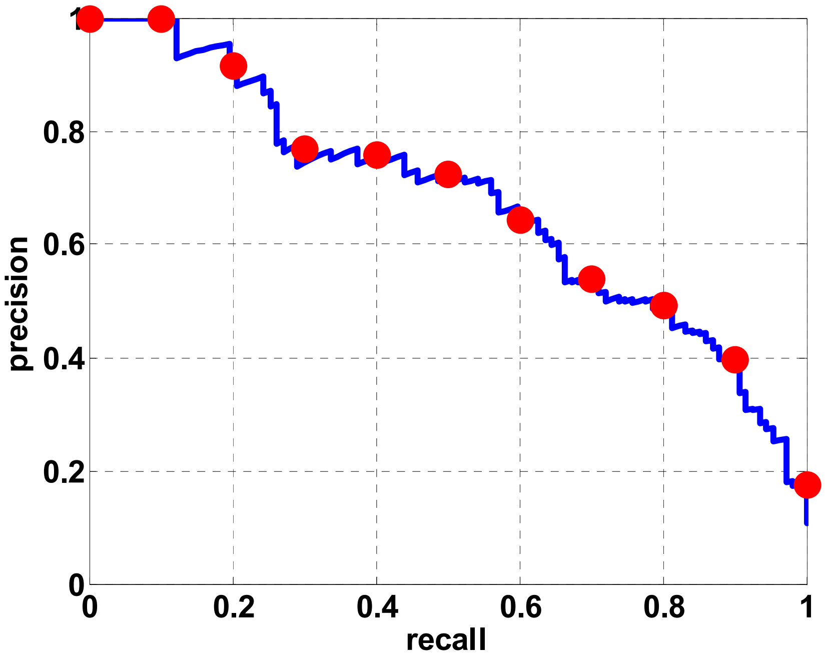
Measuring retrieval performance: Precision - Recall

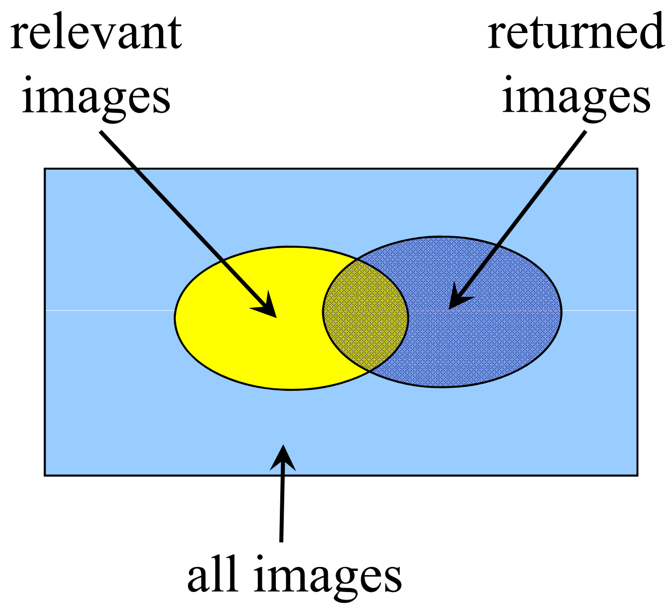
- Precision: % of returned images that are relevant



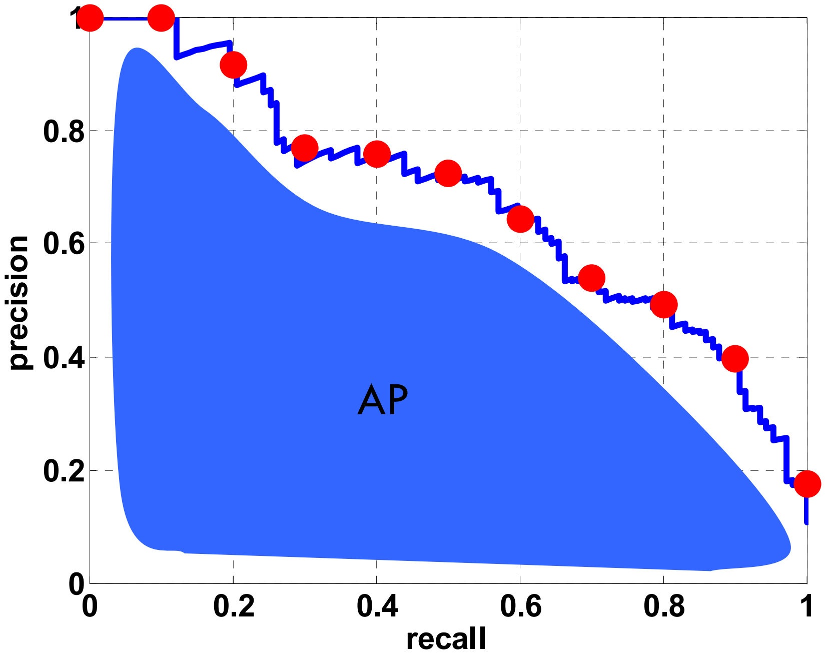
- Recall: % of relevant images that are returned





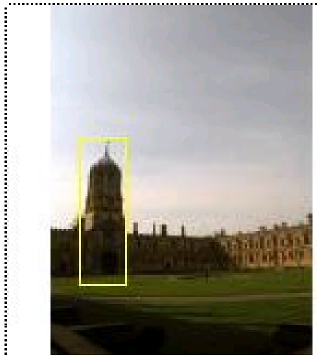


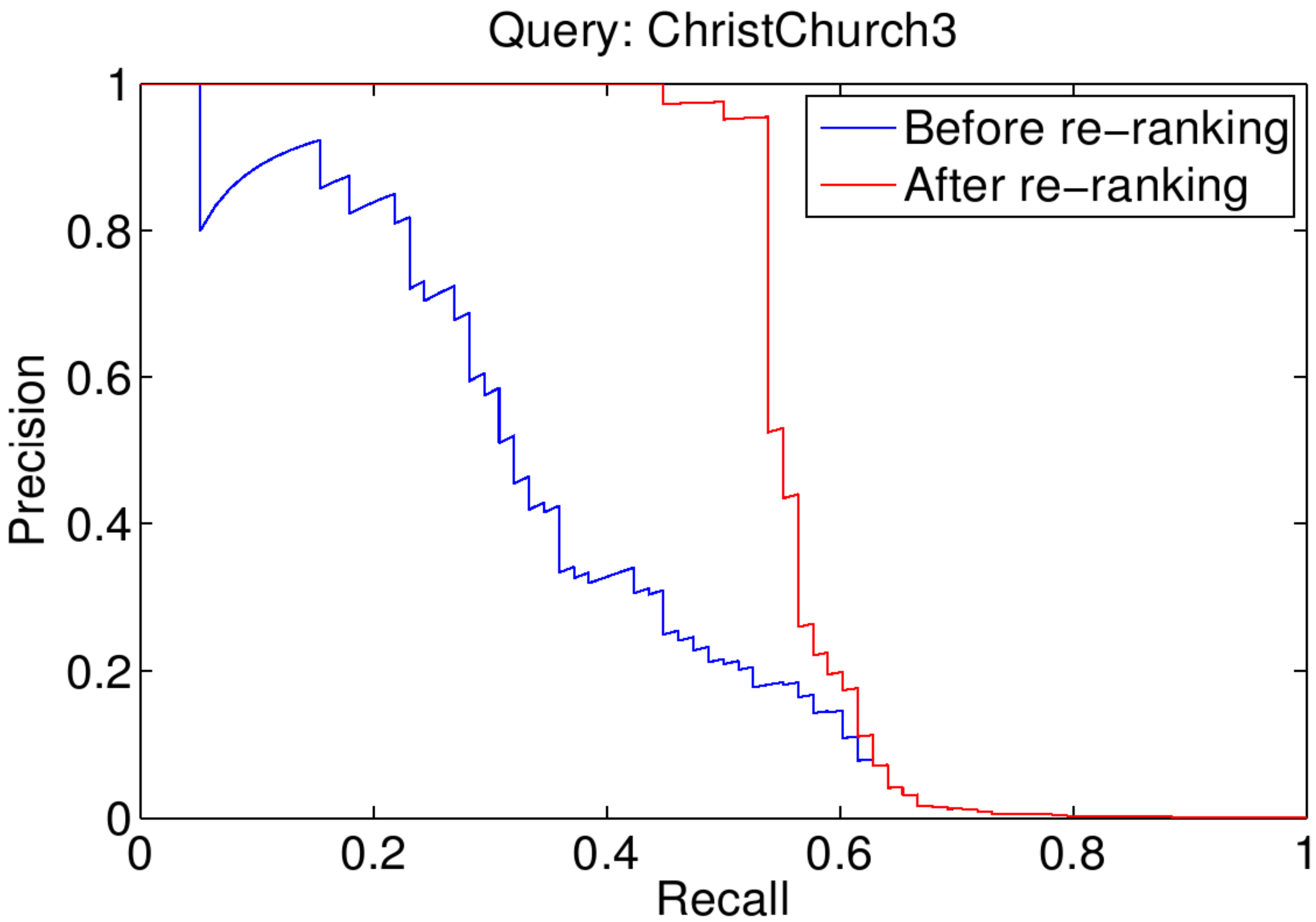
Average Precision



- A good AP score requires both high recall and high precision
- Application-independent

Performance measured by mean Average Precision (mAP) over 55 queries on 100K or 1.1M image datasets



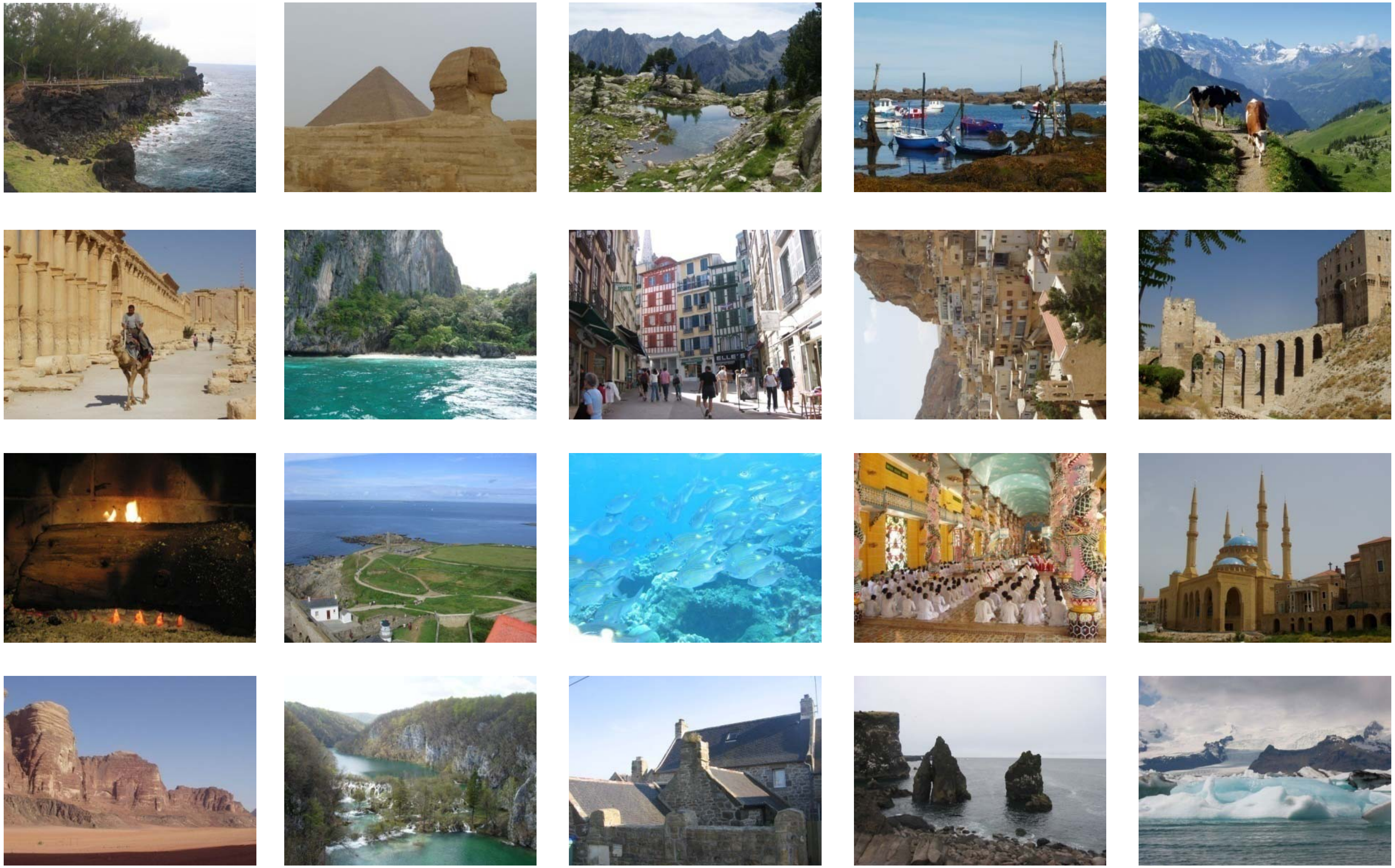

Query: ChristChurch3
Before re−ranking
After re−ranking
Precision
Recall
0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1

INRIA holidays dataset

- 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

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

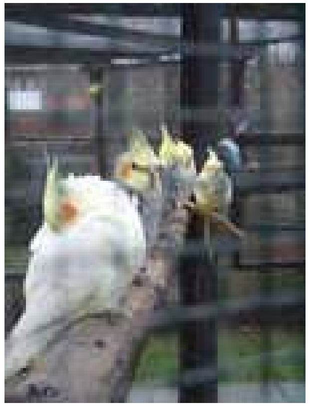
Holiday dataset – example queries



Dataset : Venice Channel

Dataset : San Marco square

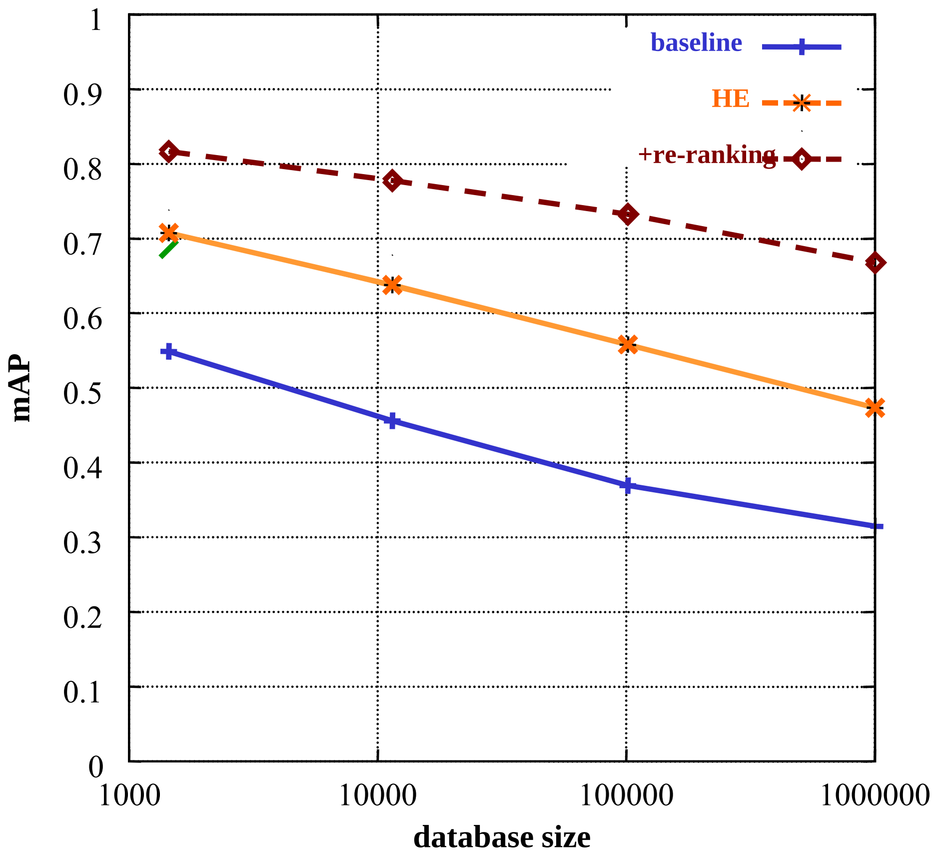
Example distractors - Flickr



Experimental evaluation

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



Results – Venice Channel

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

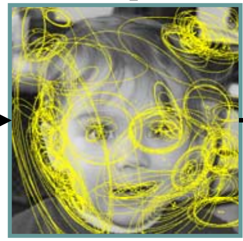
Very large scale image search

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

description vector

Vector compression

Vector search

ranked image short-list

Geometric verification

[Lowe 04, Chum & al 2007]

Re-ranked list

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

•Vector compression to reduce storage requirements and search time

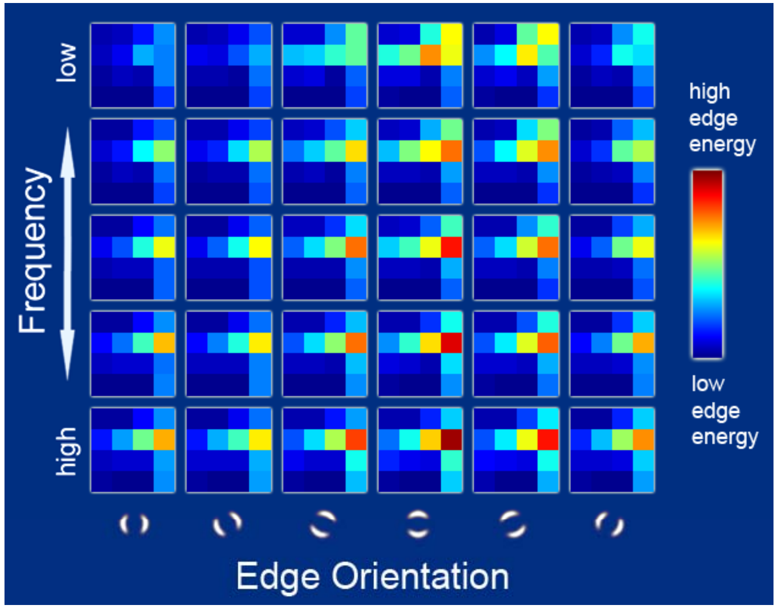
Very large scale image search

- GIST descriptors with spectral hashing [Weiss et al.'08]
  - very limited invariance to crop, scale, rotation

- Aggregating local descriptors into compact image representations [Jegou et al.'10]

Global scene context – GIST descriptor

- The "gist" of a scene: Oliva & Torralba (2001)



- 5 frequency bands and 6 orientations for each image location
- Tiling of the image to describe the image

- The position of the descriptor in the image is encoded in the representation

GIST descriptor + spectral hashing

- Spectral hashing produces binary codes similar to spectral clusters [Weiss et al.'08]
    - Each image is represented by a binary code, comparison with Hamming distance
    - Hamming distance should correlate with semantic similarity
    - Spectral clustering to generate cluster and codes
    - Very compact codes

Aggregating local descriptors

- Set of n local descriptors → 1 vector
- Popular approach: bag of features, often with SIFT features
- Recently improved aggregation schemes
  - Fisher vector [Perronnin & Dance ‘07]
  - VLAD descriptor [Jegou, Douze, Schmid, Perez ‘10]
  - Supervector [Zhou et al. ‘10]
  - Sparse coding [Wang et al. ’10, Boureau et al.’10]
- Used in very large-scale retrieval and classification

Aggregating local descriptors

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

→ significant dimensionality reduction introduces loss

- Vector of locally aggregated descriptors (VLAD) [Jegou et al. 10]
  - non sparse vector
  - fast to compute
  - excellent results with a small vector dimensionality

- Fisher vector [Perronnin & Dance 07]
  - probabilistic version of VLAD
  - initially used for image classification
  - comparable or improved performance over VLAD for image retrieval

VLAD : vector of locally aggregated descriptors

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

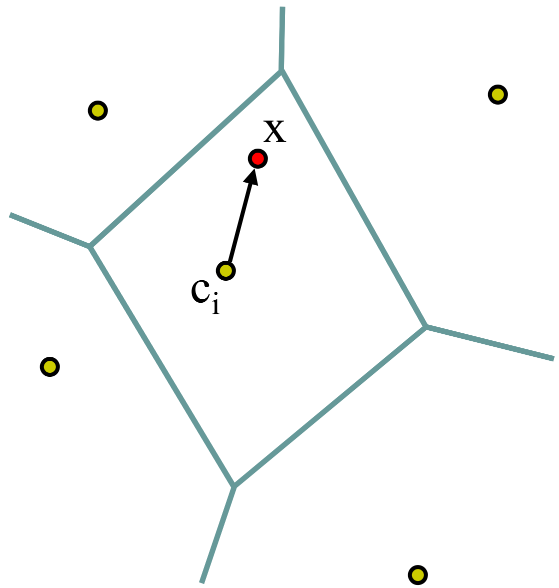
- For a given image
  - assign each descriptor to closest center c_i
  - accumulate (sum) descriptors per cell

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

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

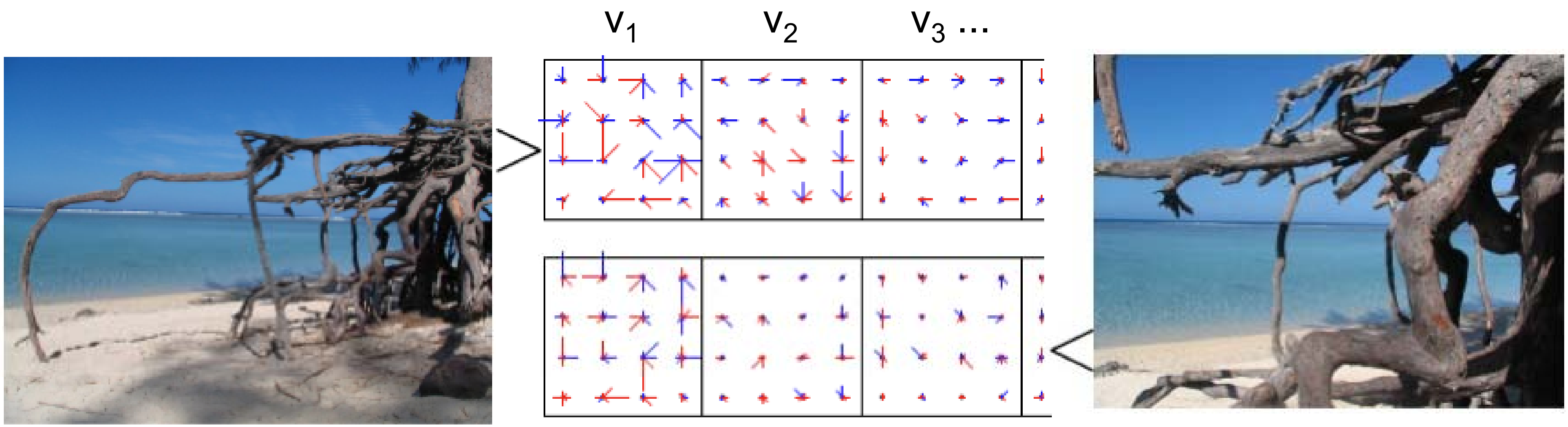
- The vector is square-root + L2-normalized

- Alternative: Fisher vector



[Jegou, Douze, Schmid, Perez, CVPR'10]

VLADs for corresponding images

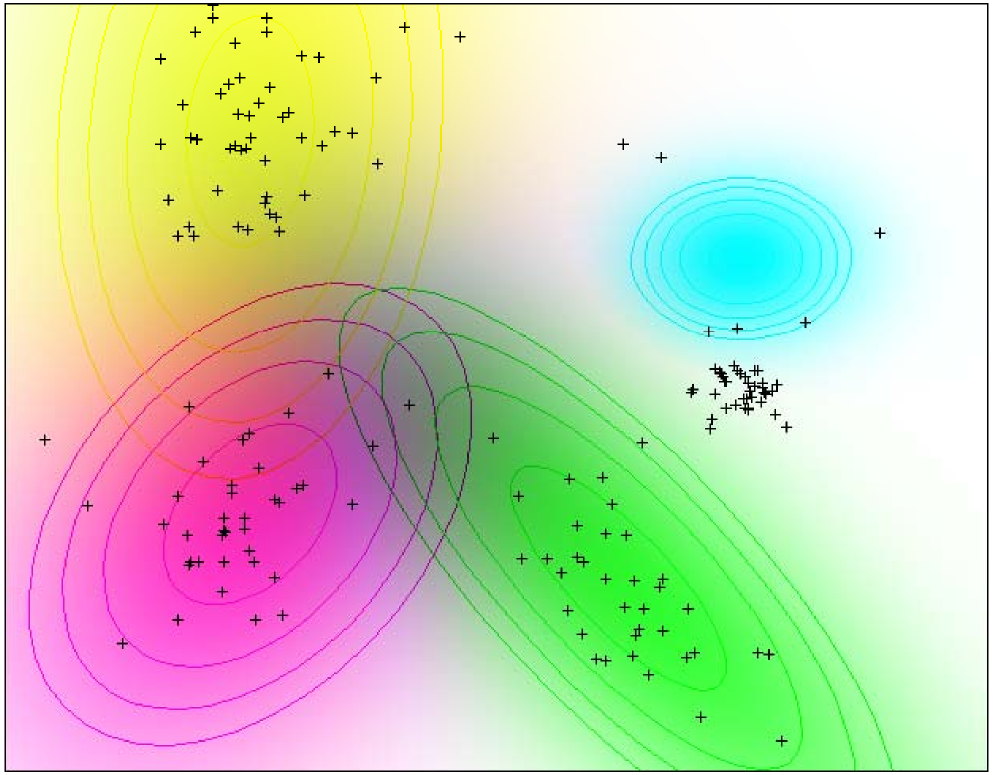


SIFT-like representation per centroid (+ components: blue, - components: red)

- good coincidence of energy & orientations

Fisher vector

- Use a Gaussian Mixture Model as vocabulary
- Statistical measure of the descriptors of the image w.r.t the GMM
- Derivative of likelihood w.r.t. GMM parameters



GMM parameters:

w_i weight

μ_i mean

σ_i co-variance (diagonal)

Translated cluster →
large derivative on μ_i for this component

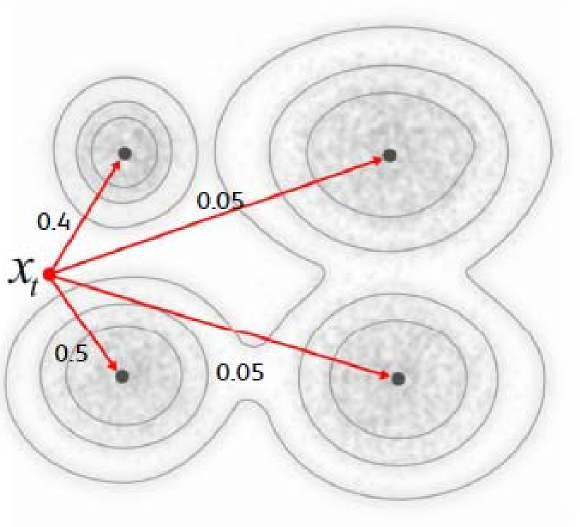
[Perronnin & Dance 07]

Fisher vector

FV formulas:

$$\mathcal{G}_{\mu,i}^{X} = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right)$$

$$\mathcal{G}_{\sigma,i}^{X} = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right]$$



$\gamma_t(i)$ = soft-assignment of patch x_t to Gaussian i

Fisher Vector = concatenation of per-Gaussian gradient vectors

For image retrieval in our experiments:

- only deviation wrt mean, dim: K*D [K number of Gaussians, D dim of descriptor]
- variance does not improve for comparable vector length

VLAD/Fisher/BOF performance and dimensionality reduction

- We compare Fisher, VLAD and BoF on INRIA Holidays Dataset (mAP %)
- Dimension is reduced to D' dimensions with PCA

Descriptor	K	D	Holidays (mAP)					
			$D' = D$	$\rightarrow D'$=2048	$\rightarrow D'$=512	$\rightarrow D'$=128	$\rightarrow D'$=64	$\rightarrow D'$=32
BOW	1 000	1 000	40.1		43.5	44.4	43.4	40.8
	20 000	20 000	43.7	41.8	44.9	45.2	44.4	41.8
Fisher (μ)	16	1 024	54.0		54.6	52.3	49.9	46.6
	64	4 096	59.5	60.7	61.0	56.5	52.0	48.0
	256	16 384	62.5	62.6	57.0	53.8	50.6	48.6
VLAD	16	1 024	52.0		52.7	52.6	50.5	47.7
	64	4 096	55.6	57.6	59.8	55.7	52.3	48.4
	256	16 384	58.7	62.1	56.7	54.2	51.3	48.1
GIST		960	36.5					

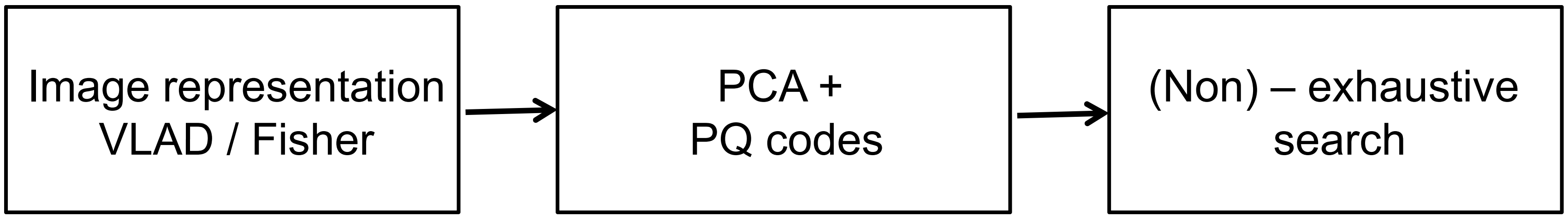
- Observations:
  - Fisher, VLAD better than BoF for a given descriptor size
  - Choose a small D if output dimension D' is small
  - Performance of GIST not competitive

[Jegou, Perronnin, Douze, Sanchez, Perez, Schmid, PAMI'12]

Compact image representation

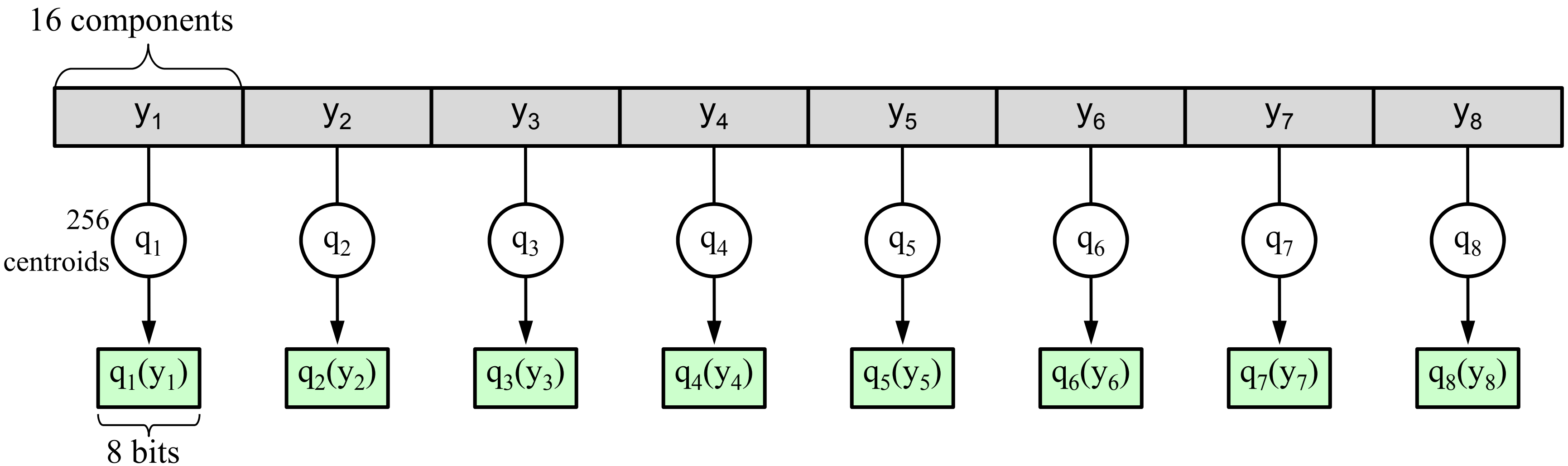
- Aim: improving the tradeoff between
  - search speed
  - memory usage
  - search quality

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



Product quantization for nearest neighbor search

- Vector split into *m* subvectors: $y \rightarrow [y_1| \dots |y_m]$
- Subvectors are quantized separately by quantizers $q(y) = [q_1(y_1)| \dots |q_m(y_m)]$
  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



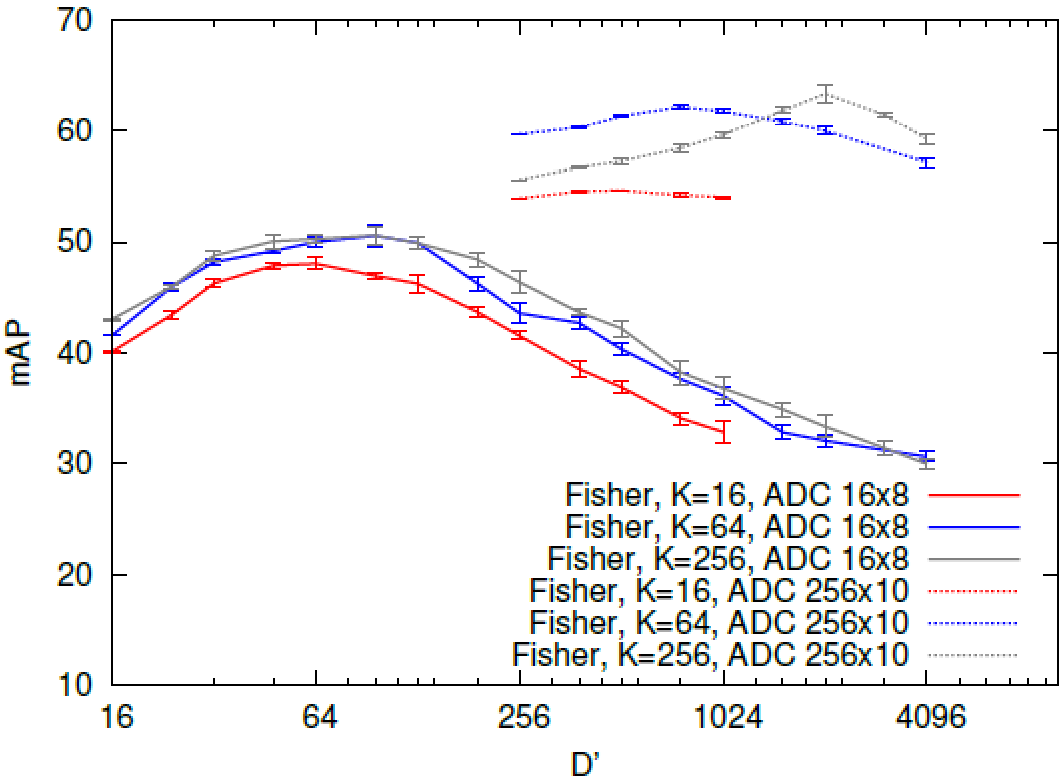
$\Rightarrow$ 8 subvectors x 8 bits = 64-bit quantization index

[Jegou, Douze, Schmid, PAMI'11]

Optimizing the dimension reduction and quantization together

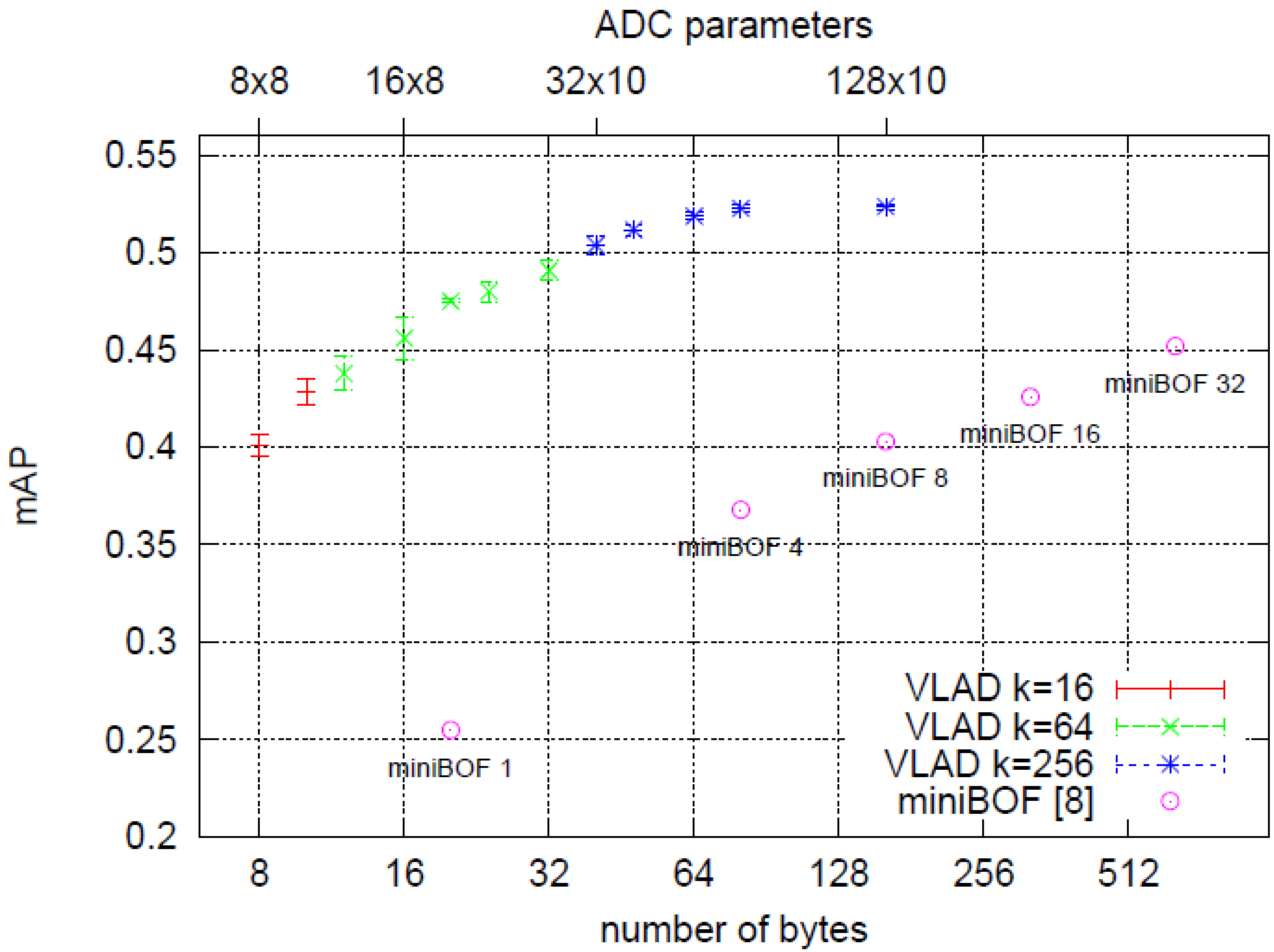
- Fisher vectors undergoes two approximations
  - mean square error from PCA projection
  - mean square error from quantization

- 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

Results on the Holidays dataset with various quantization parameters



Joint optimization of Fisher/VLAD and dimension reduction-indexing

- For Fisher/ \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

Comparison to the state of the art

Method	bytes	UKB	Holidays
BOW, K=20,000	10 364	2.87	43.7
BOW, K=200,000	12 886	2.81	54.0
miniBOF [12]	20	2.07	25.5
	80	2.72	40.3
	160	2.83	42.6
FV K=64, spectral hashing 128 bits	16	2.57	39.4
VLAD, K=16, ADC 16×8 [23]	16	2.88	46.0
VLAD, K=64, ADC 32×10 [23]	40	3.10	49.5
FV K=8, binarized [22]	65	2.79	46.0
FV K=64, binarized [22]	520	3.21	57.4
FV K=64, ADC 16×8 (D'=96)	**16**	**3.10**	**50.6**
FV K=256, ADC 256×10 (D'=2048)	**320**	**3.47**	**63.4**

[12] H. Jégou, M. Douze, and C. Schmid, "Packing bag-of-features," in *ICCV*, September 2009.

[22] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *CVPR*, June 2010.

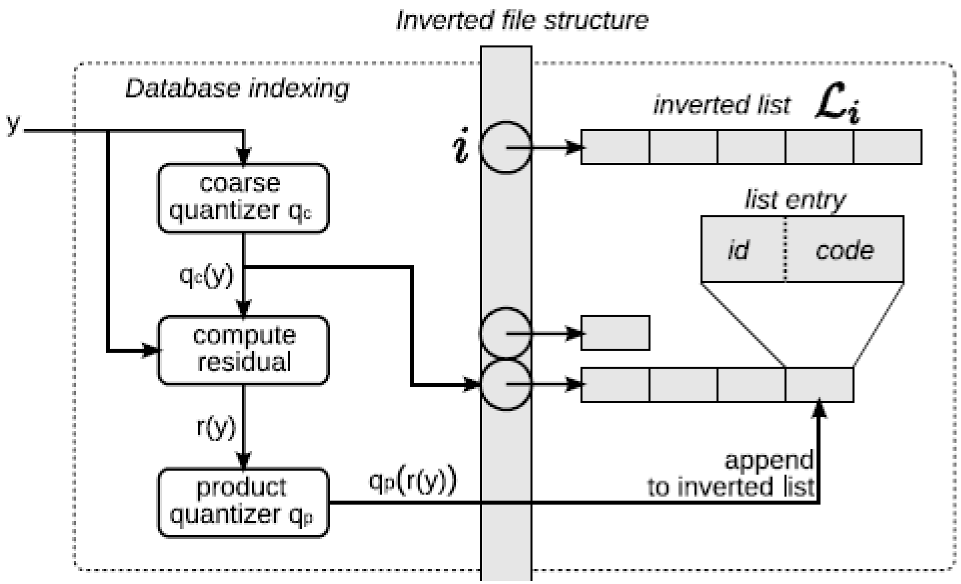
[23] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, June 2010.

Large scale experiments (10 million images)

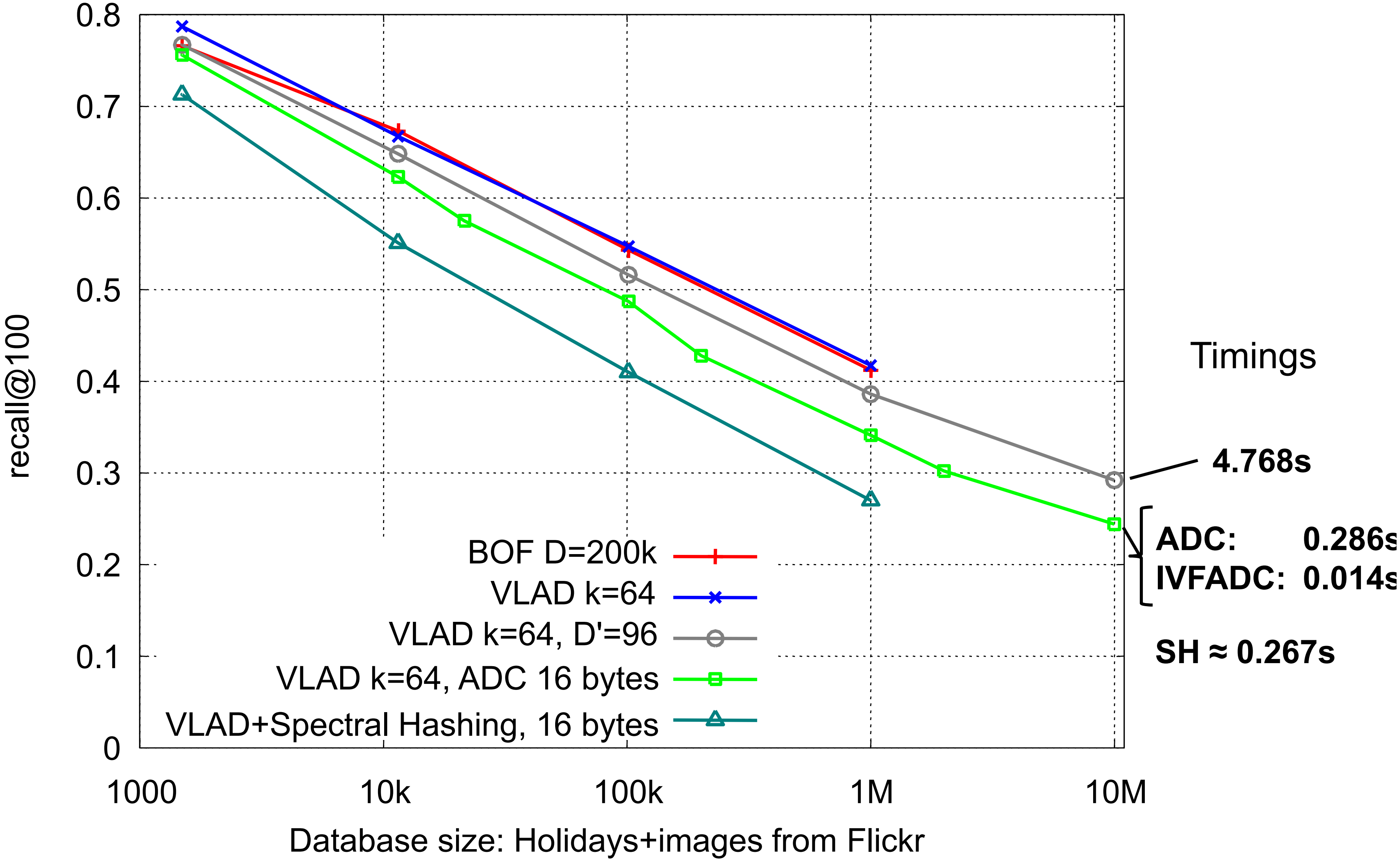
- Exhaustive search of VLADs, D'=64
  - 4.77s

- 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)



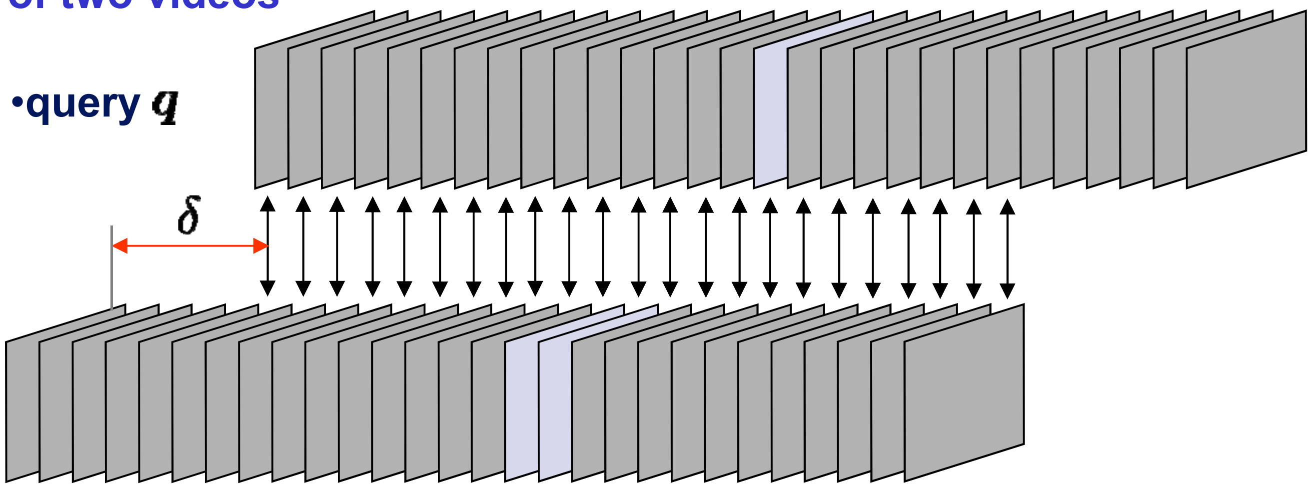
Event retrieval in large video collections [Revaud et al. 2013]

Video description

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

Comparison of two videos

- query q



- database
  - video v

Fast calculation in the frequency domain + product quantization