Instance-level recognition

1) Local invariant features

2) Matching and recognition with local features

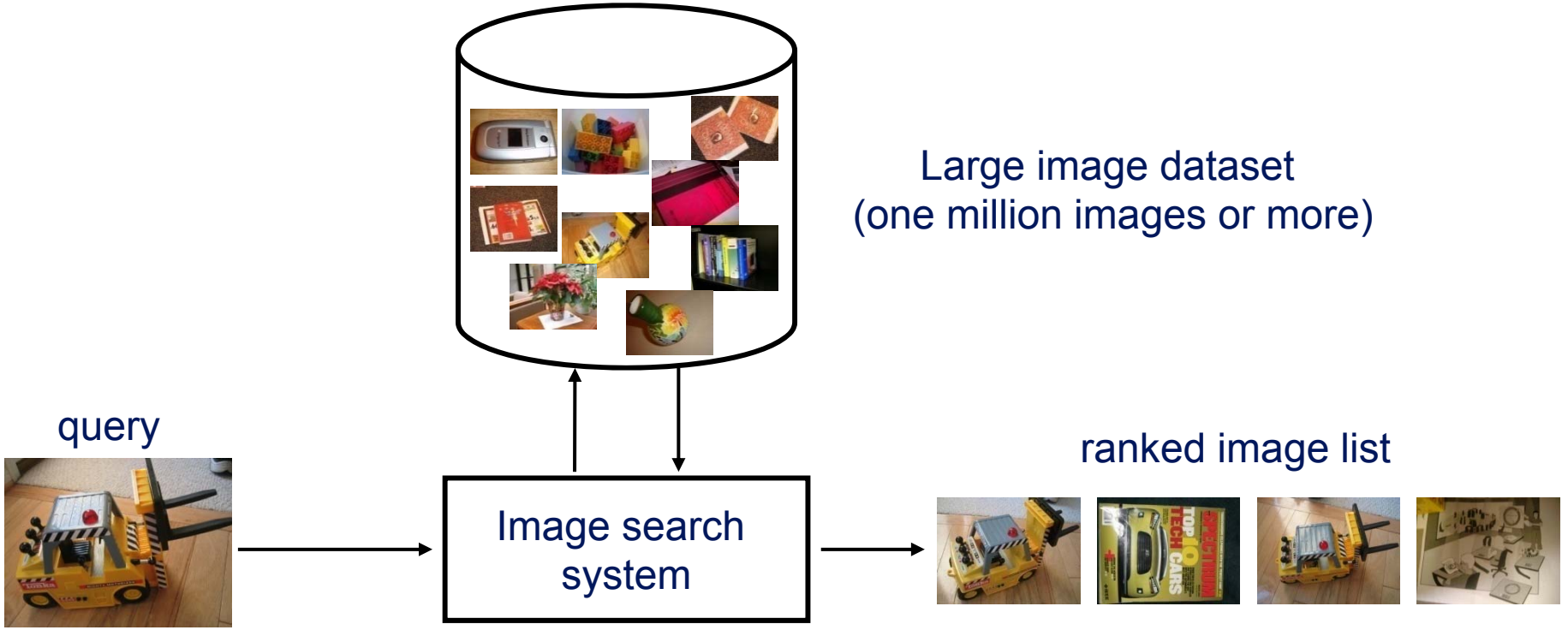
3) Efficient visual search

4) Very large scale indexing

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

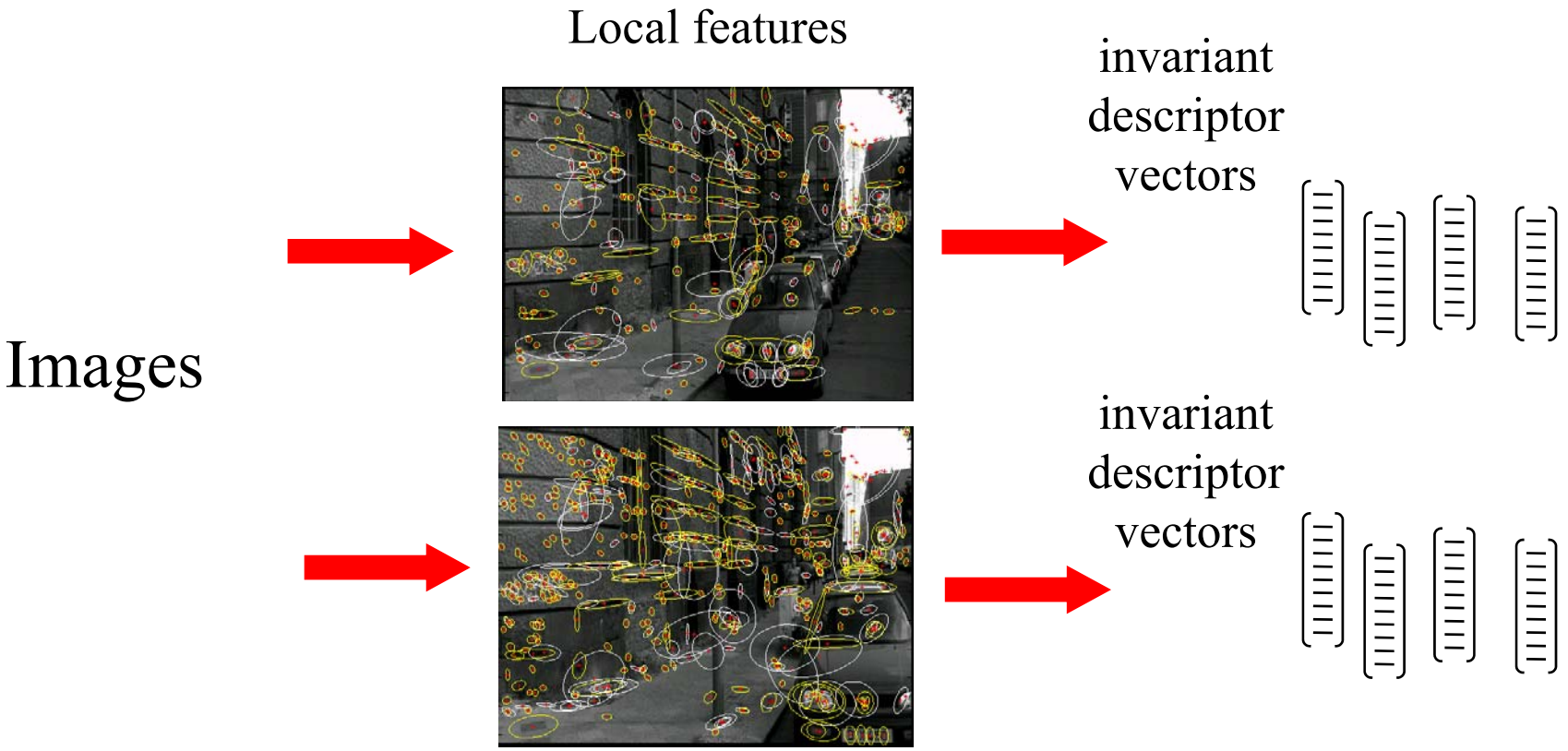
Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors

2. Quantize descriptors into a “visual vocabulary” and use efficient techniques from text retrieval

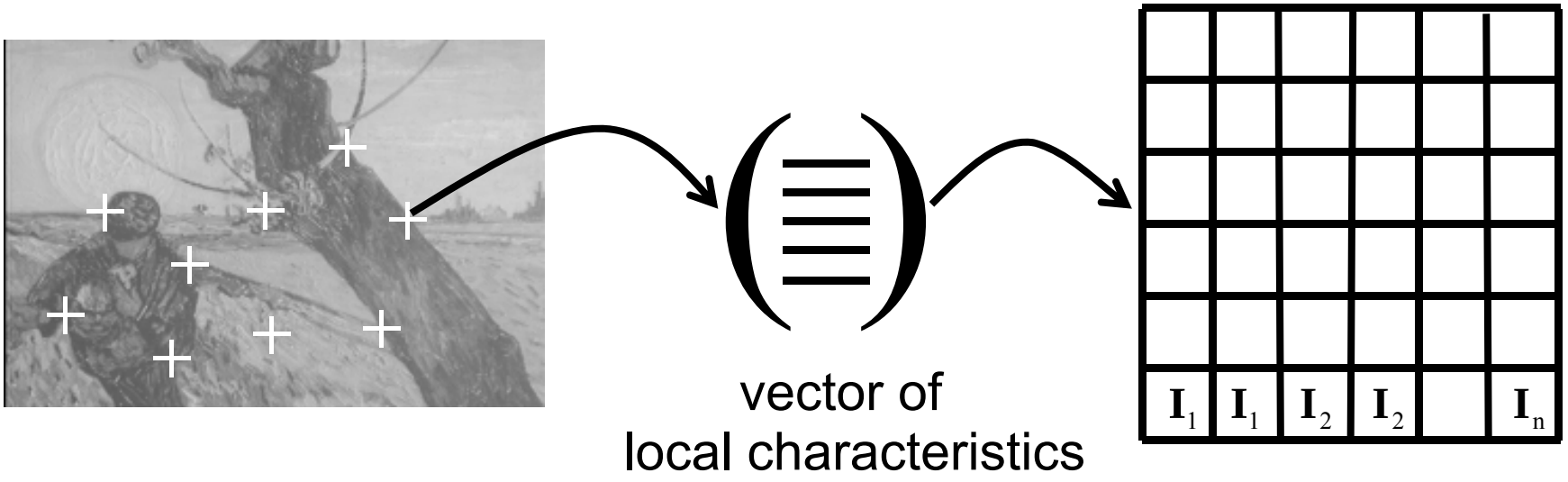
   (Bag-of-words representation)

Strategy 1: Efficient approximate NN search

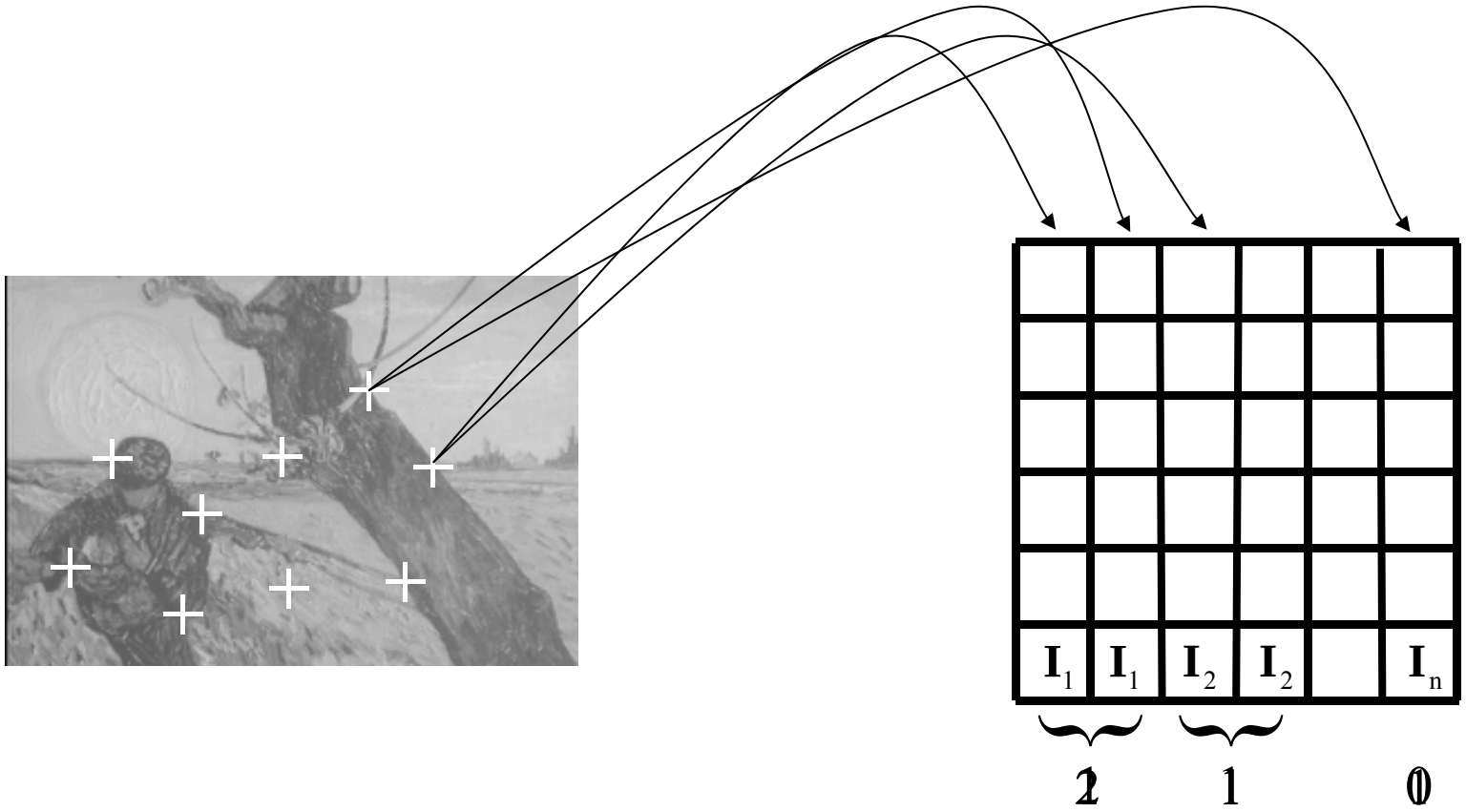


1. Compute local features in each image independently
2. Describe each feature by a descriptor vector
3. Find nearest neighbour vectors between query and database
4. Rank matched images by number of (tentatively) corresponding regions
5. Verify top ranked images based on spatial consistency

Voting algorithm



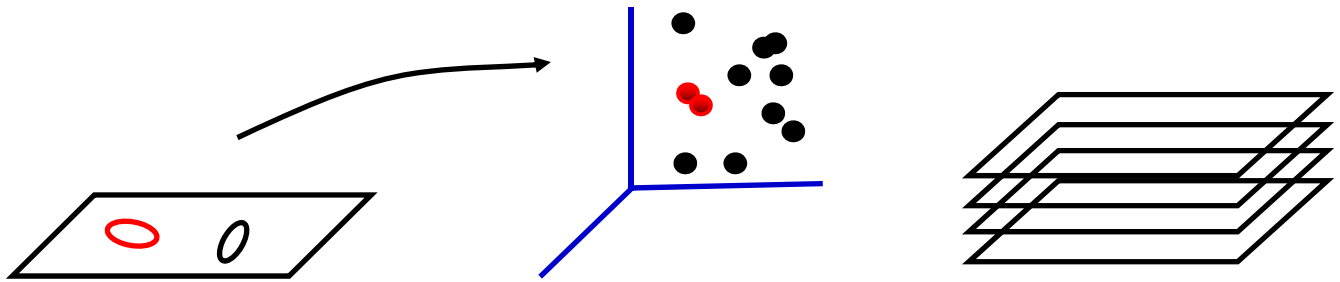
Voting algorithm



I_1 is the corresponding model image

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

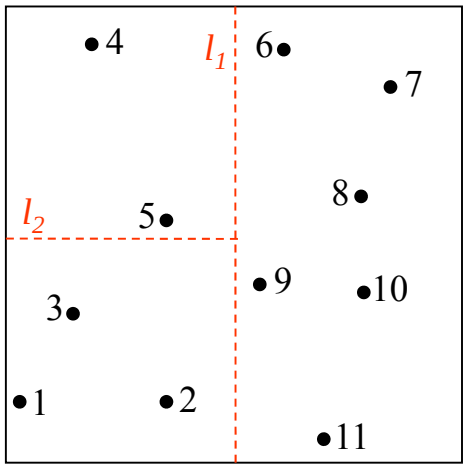
K-d tree

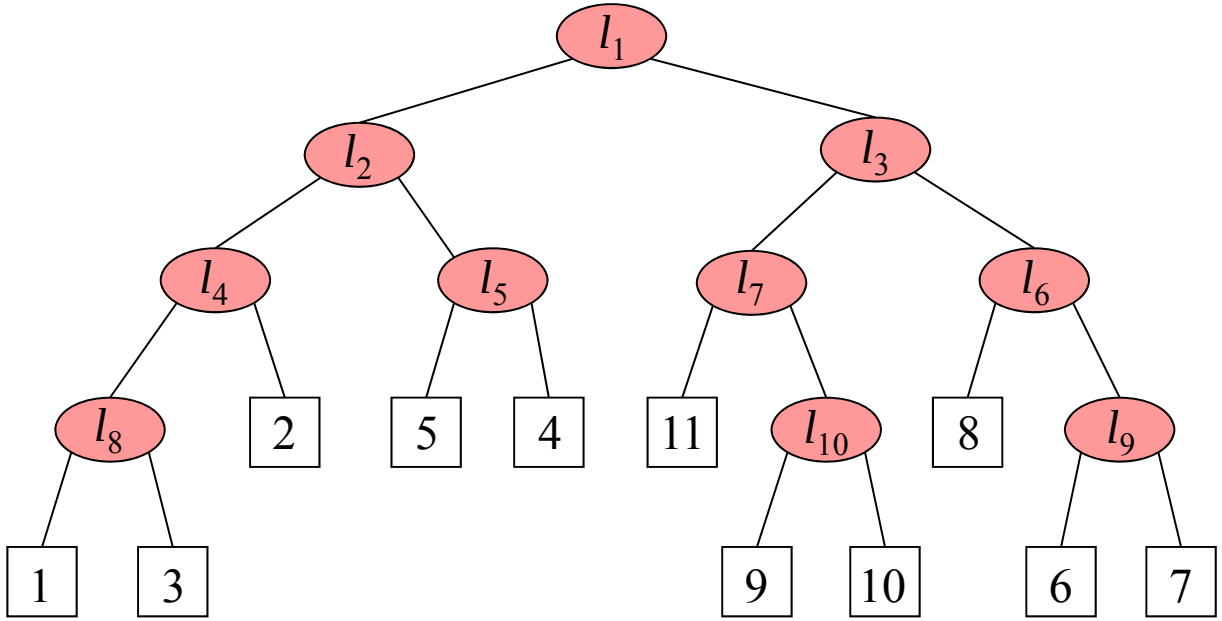
• K-d tree is a binary tree data structure for organizing a set of points

• Each internal node is associated with an axis aligned hyper-plane splitting its associated points into two sub-trees

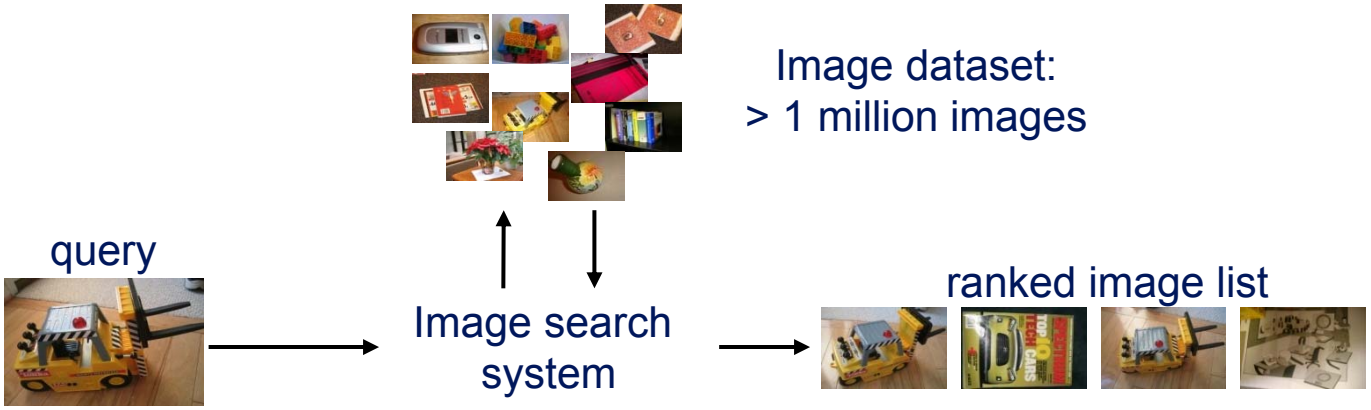
• Dimensions with high variance are chosen first

• Position of the splitting hyper-plane is chosen as the mean/median of the projected points – balanced tree



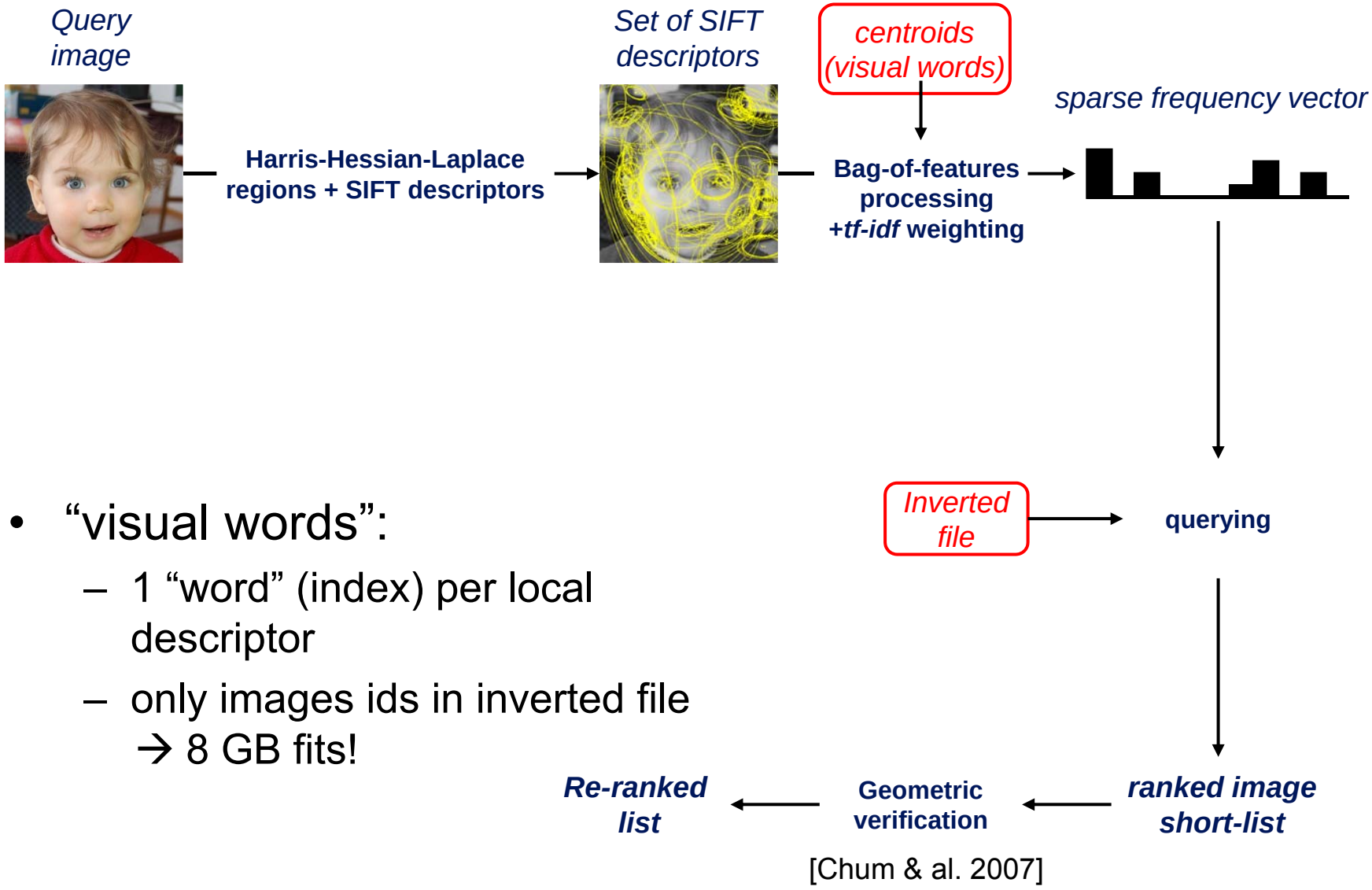


Large scale object/scene recognition



- 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]

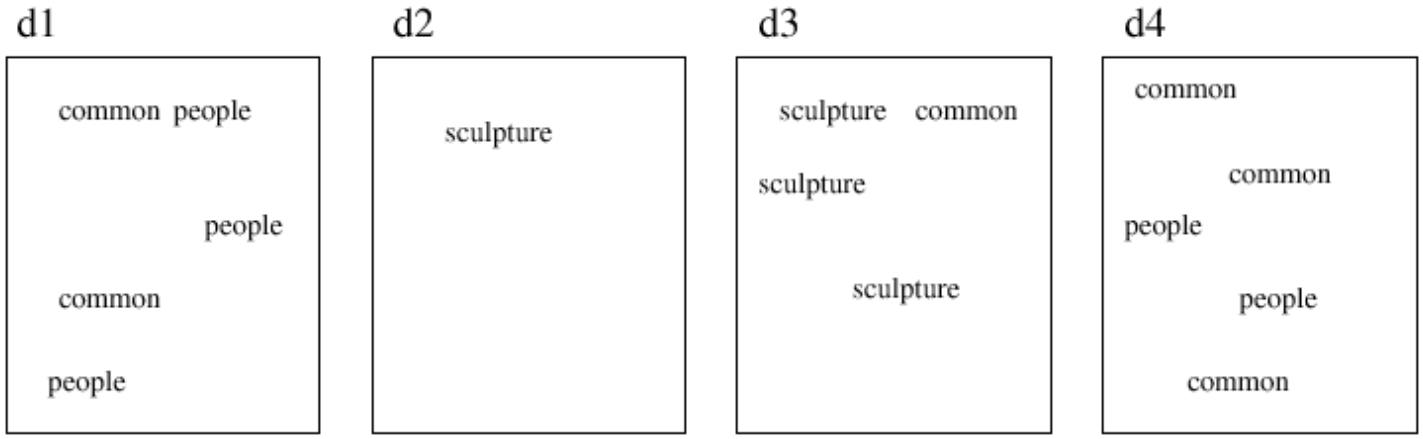


[Chum & al. 2007]

- "visual words":
  - 1 "word" (index) per local descriptor
  - only images ids in inverted file → 8 GB fits!

Indexing text with inverted files

Document collection:

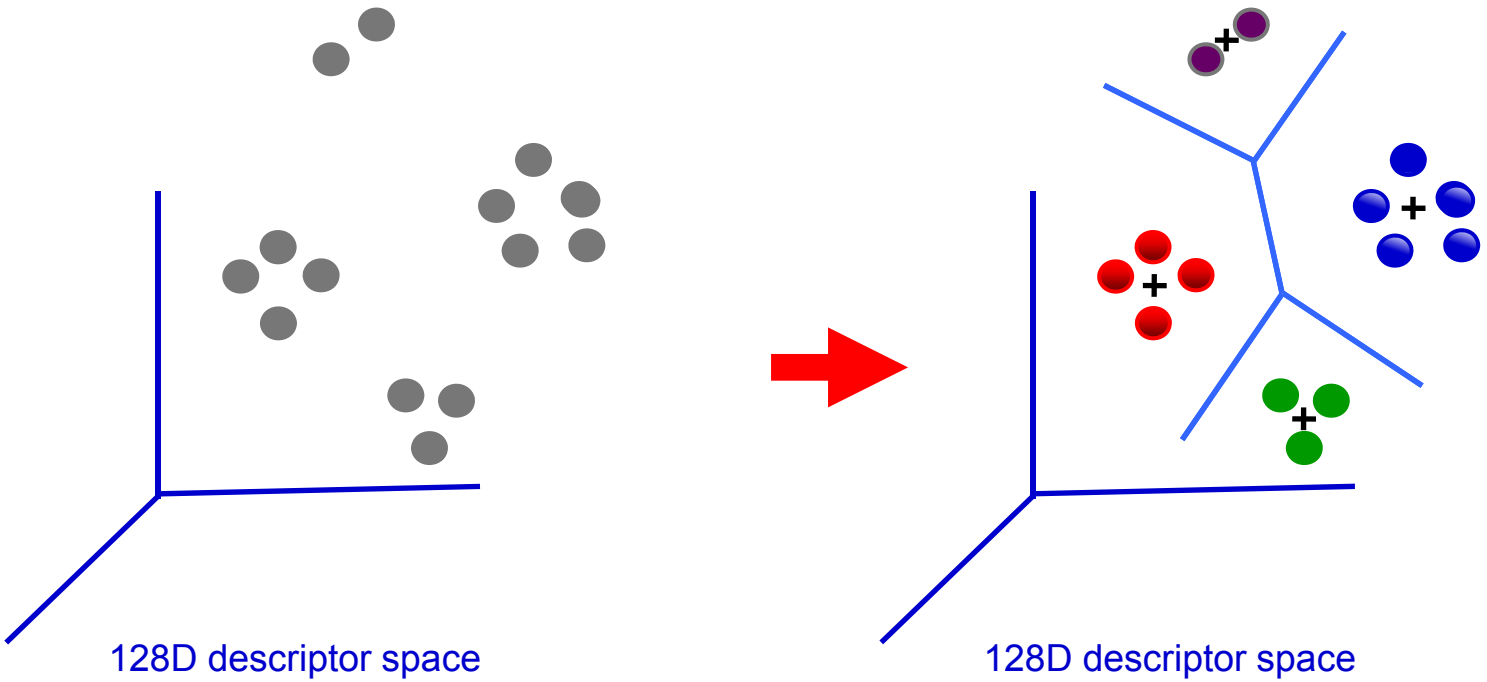


Inverted file:

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”

Build a visual vocabulary



Vector quantize descriptors

- Compute SIFT features from a subset of images
- K-means clustering (need to choose K)

[Sivic and Zisserman, ICCV 2003]

K-means clustering

Minimizing sum of squared Euclidean distances between points x_i and their nearest cluster centers

Algorithm:

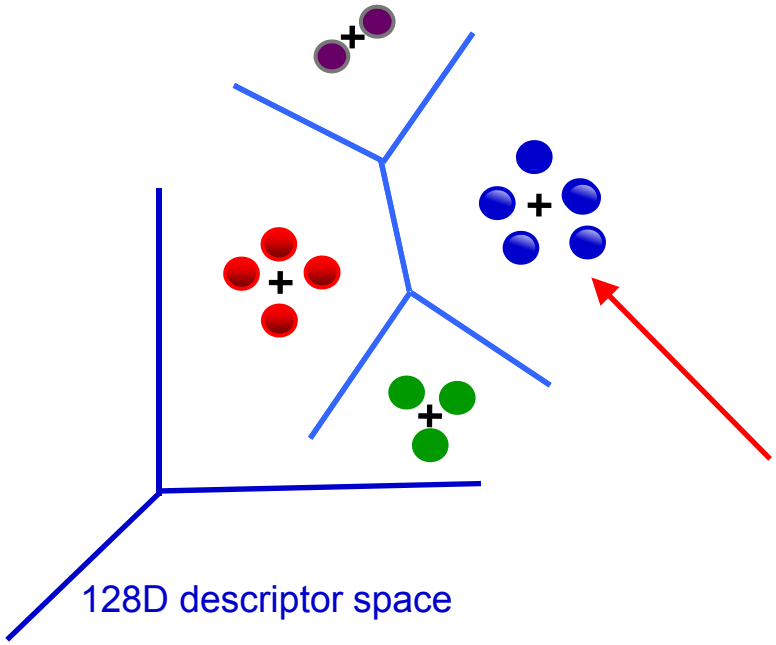
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

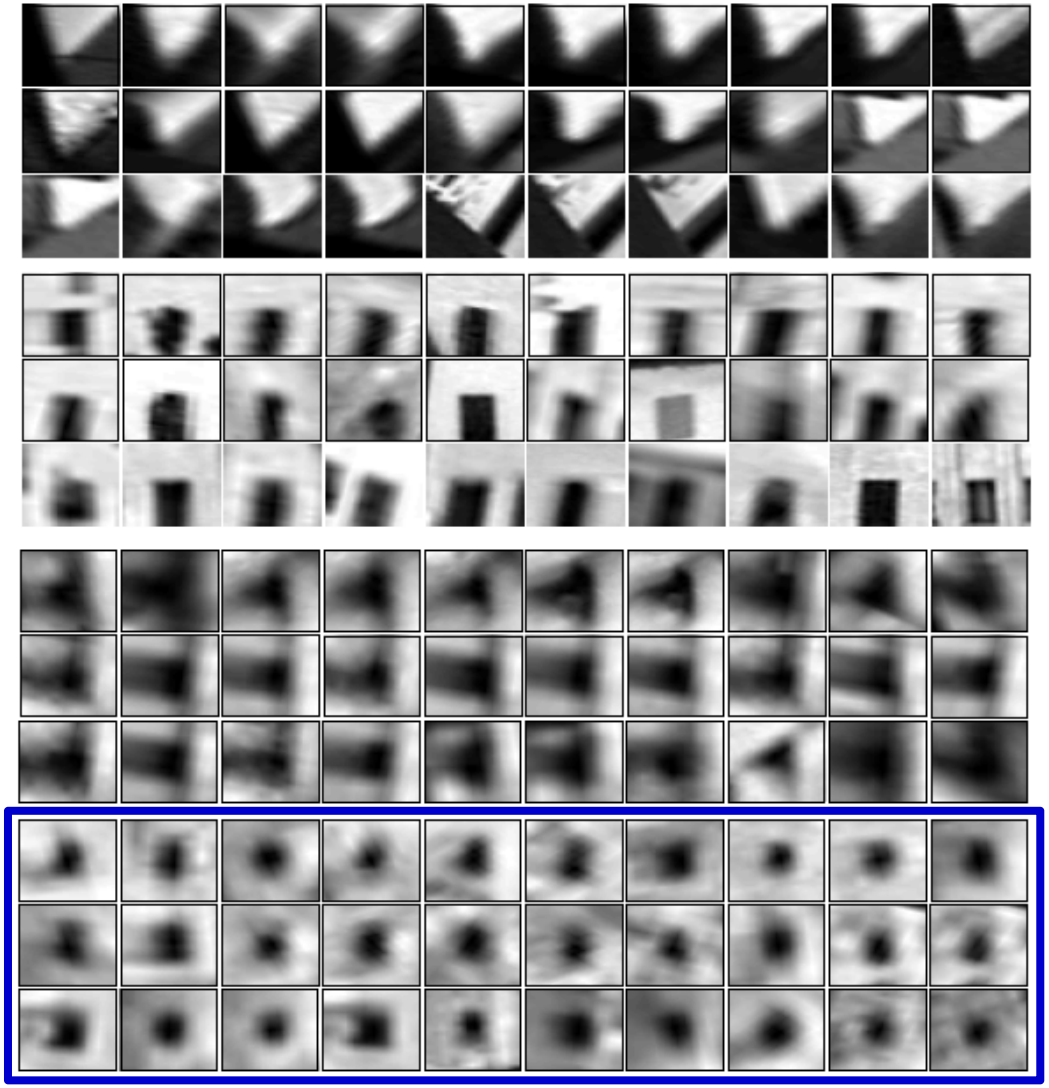
Local minimum, solution dependent on initialization

Initialization important, run several times, select best

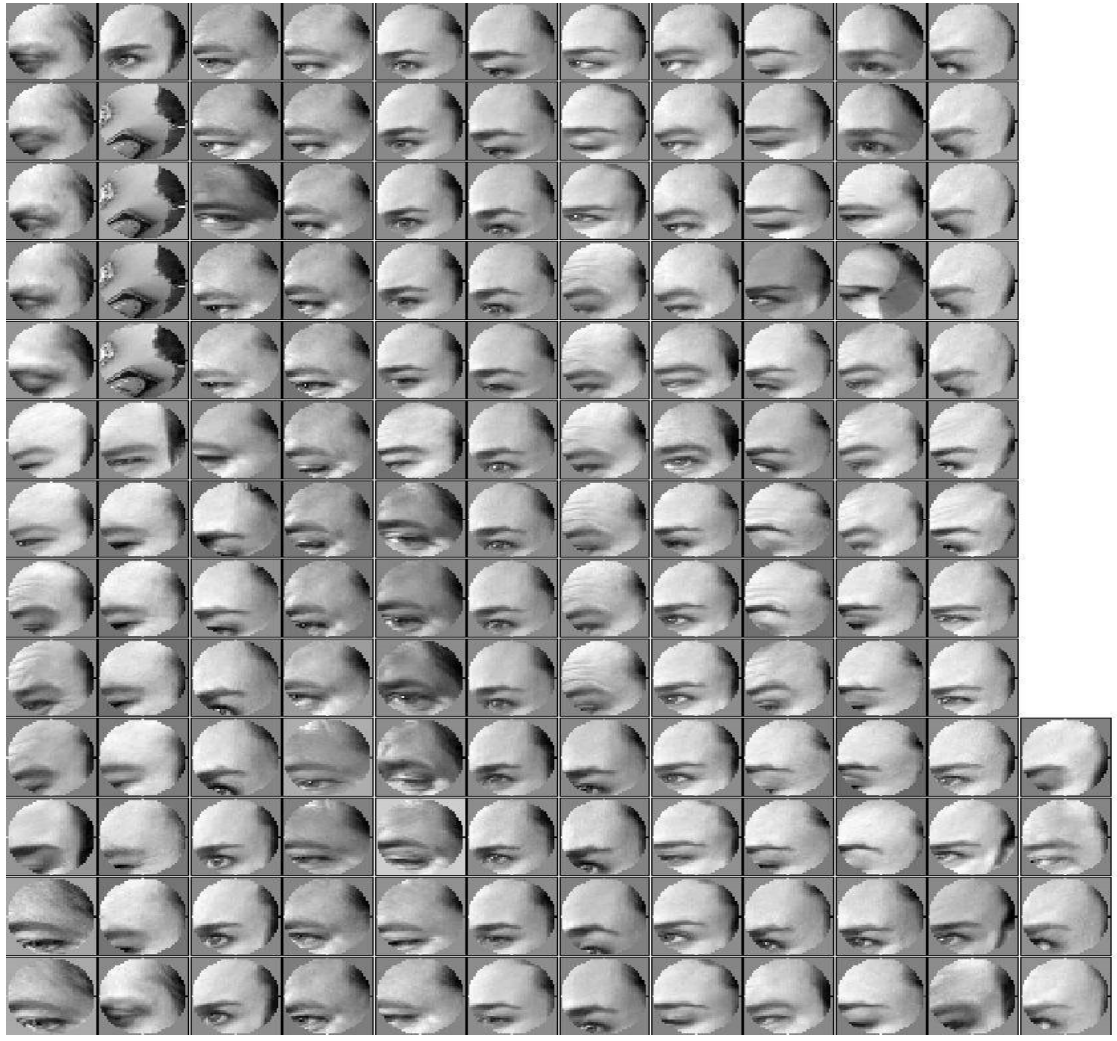
Visual words

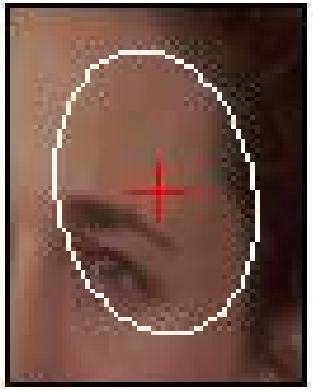
Example: each group of patches belongs to the same visual word



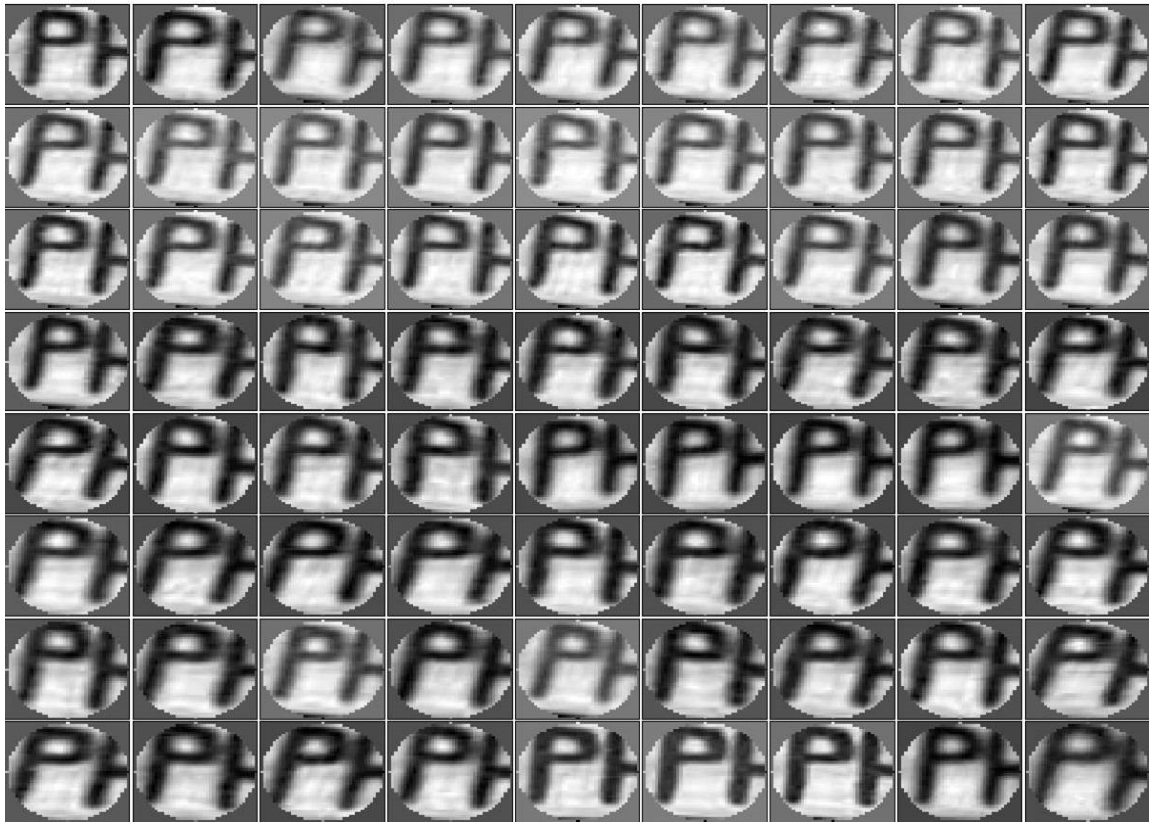


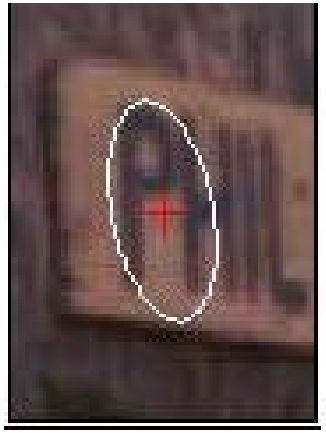
Samples of visual words (clusters on SIFT descriptors):





Samples of visual words (clusters on SIFT descriptors):



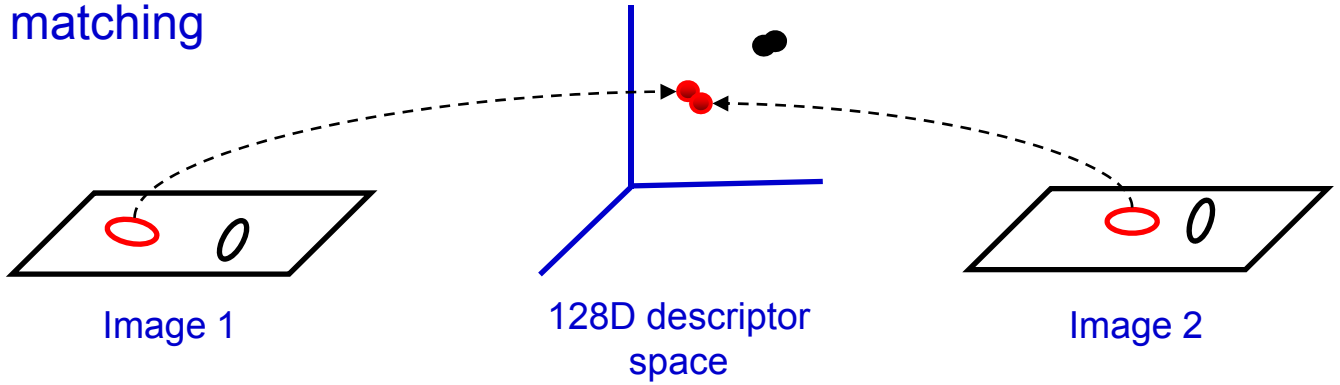


Visual words: quantize descriptor space

Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames

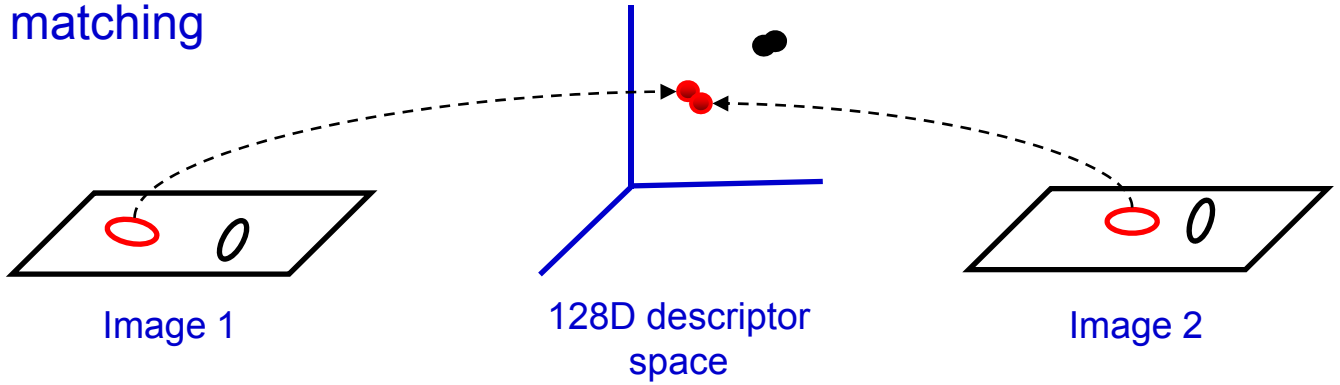


Visual words: quantize descriptor space

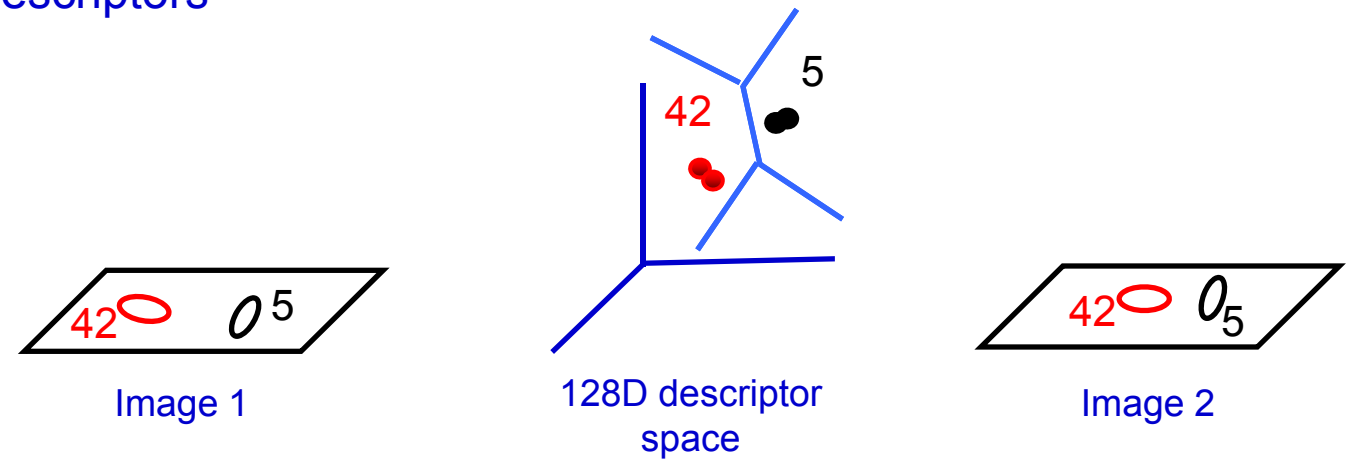
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

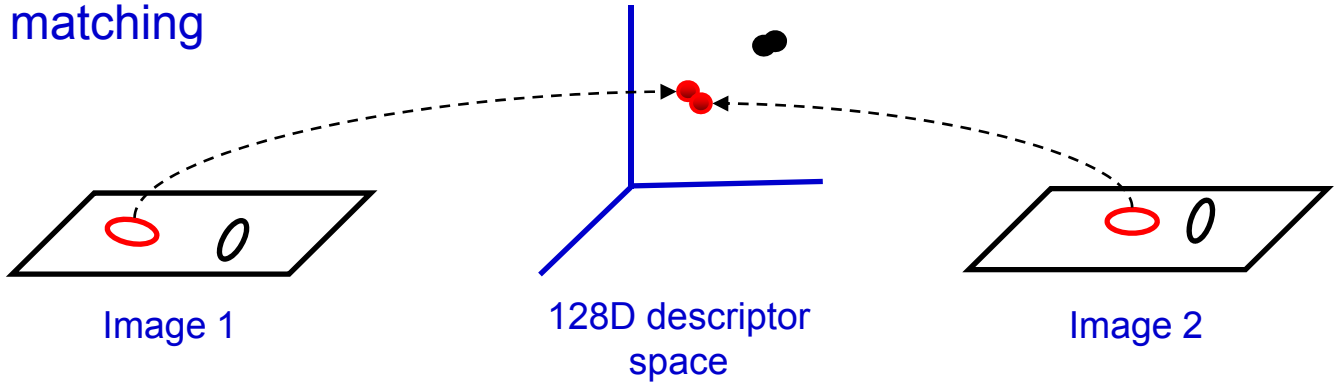


Visual words: quantize descriptor space

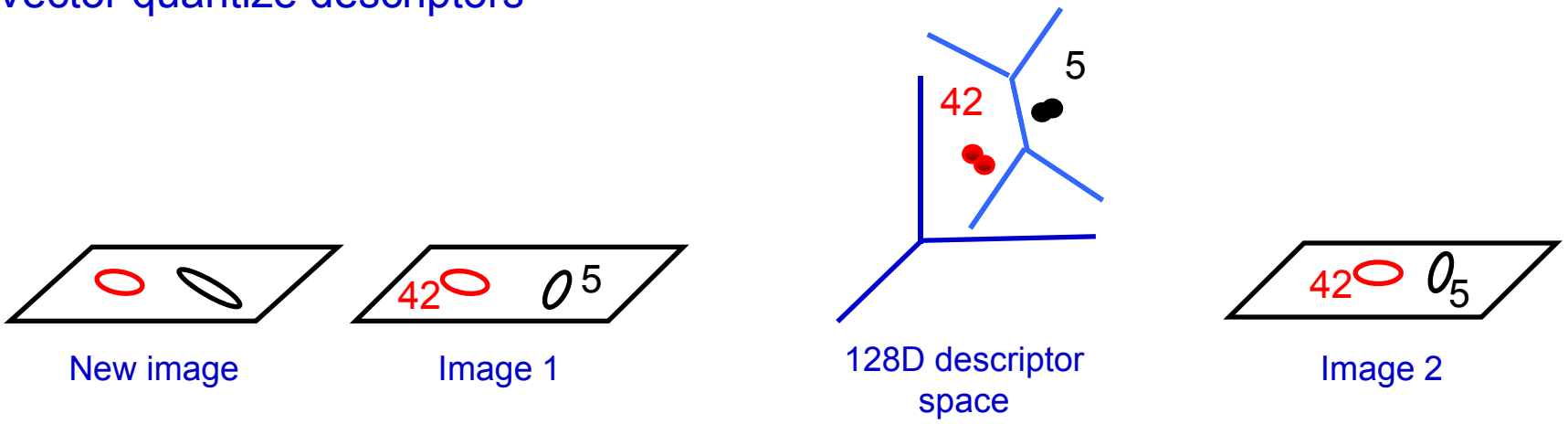
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

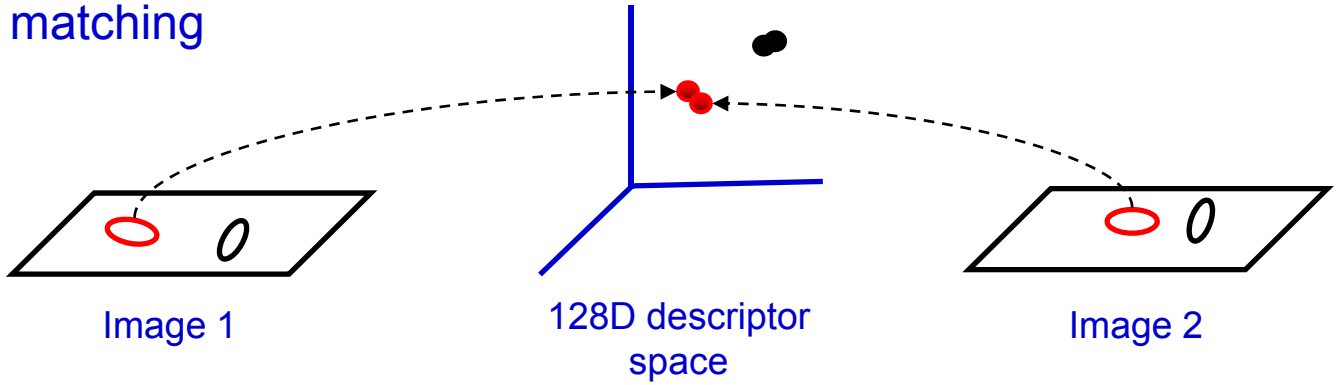


Visual words: quantize descriptor space

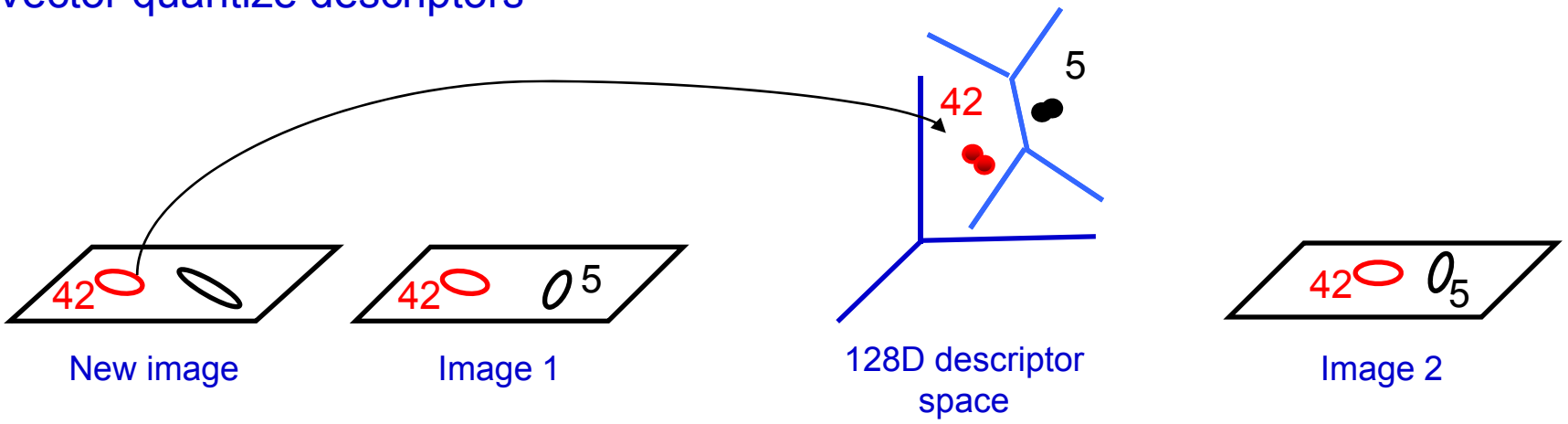
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

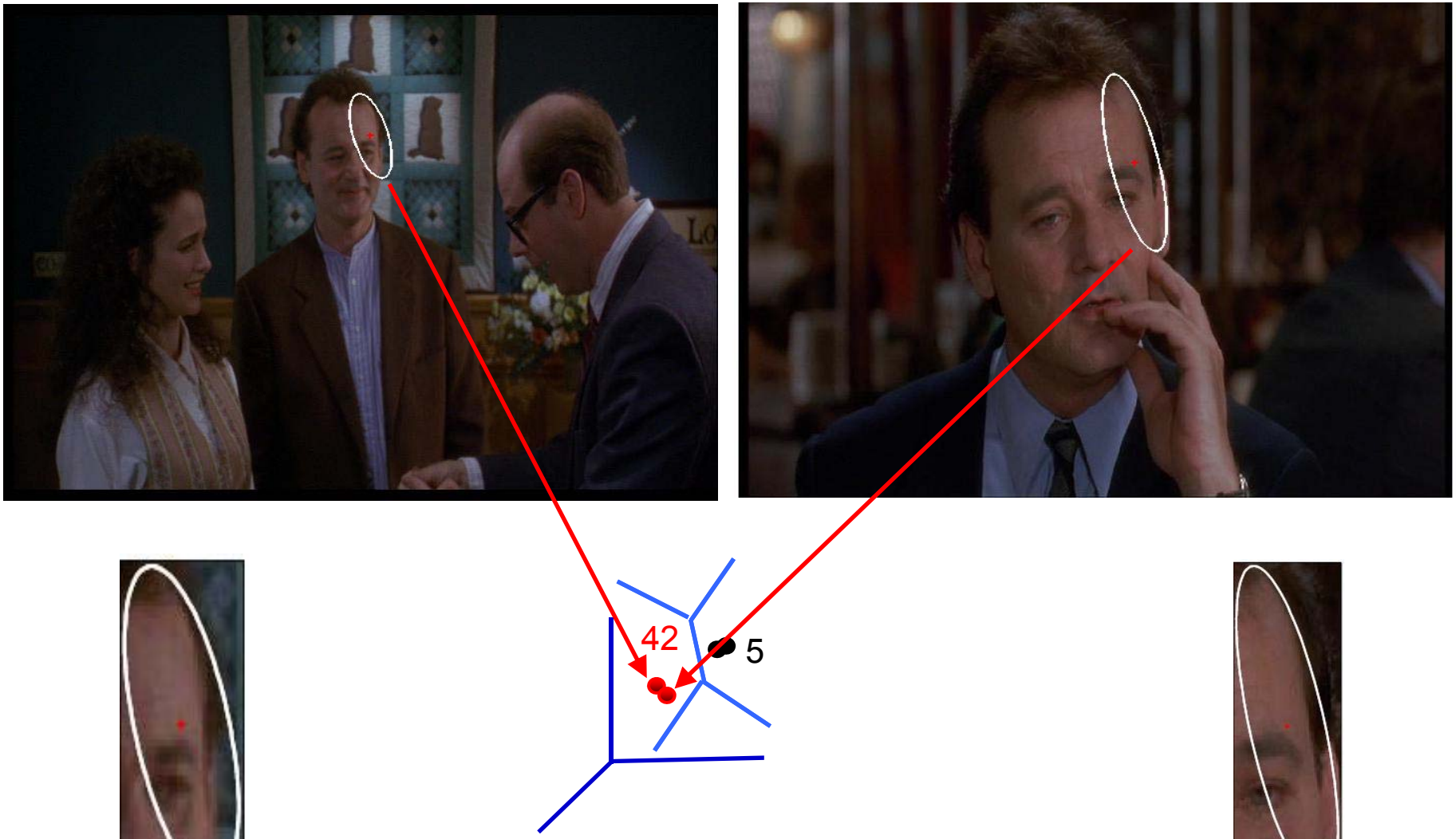
- expensive to do for all frames



Vector quantize descriptors



Vector quantize the descriptor space (SIFT)

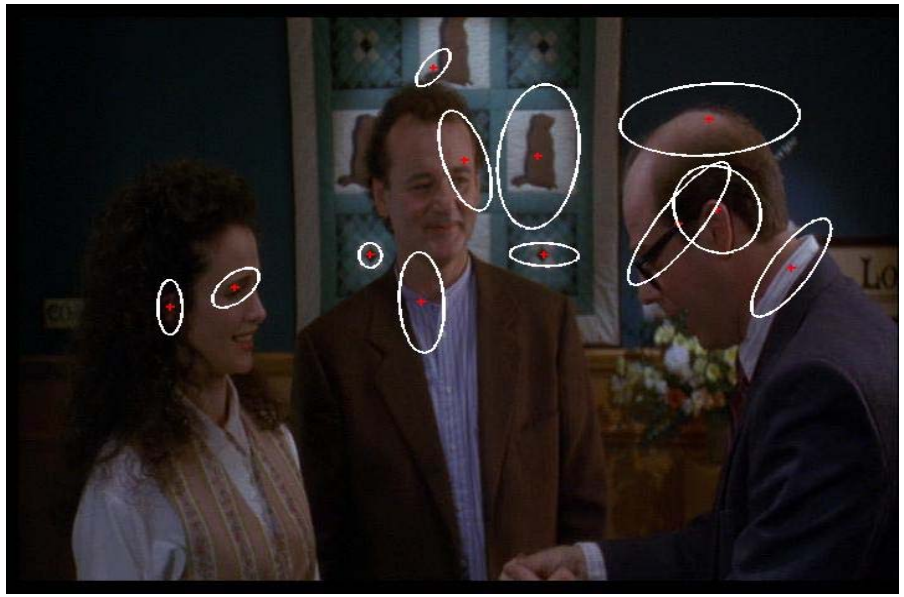


The same visual word

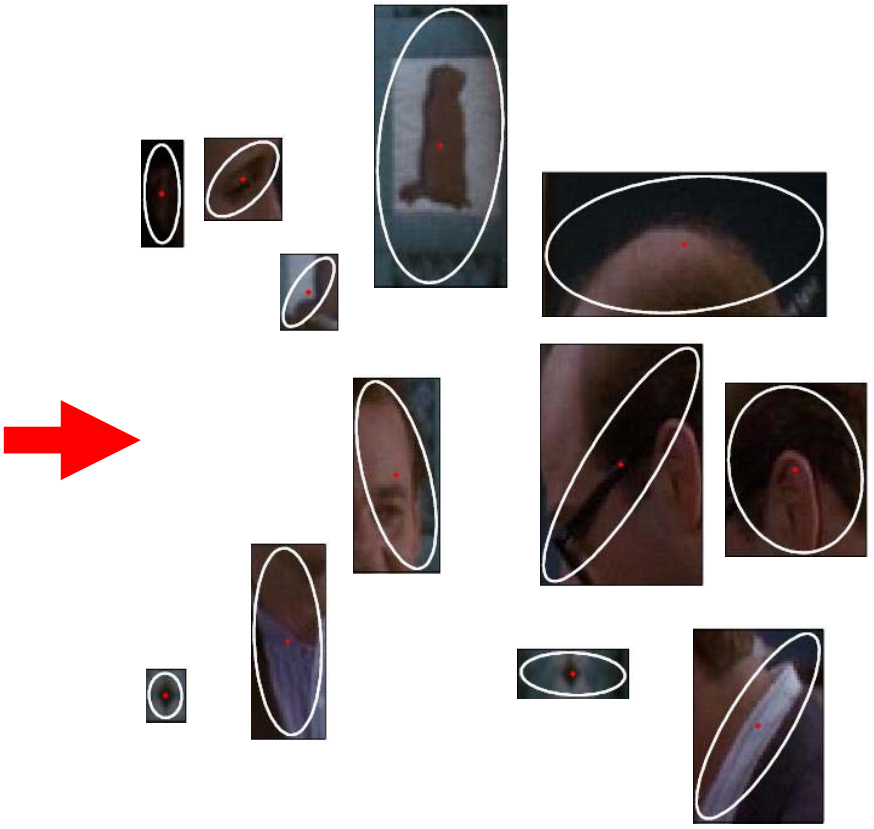
Representation: bag of (visual) words

Visual words are ‘iconic’ image patches or fragments

- represent their frequency of occurrence
- but not their position

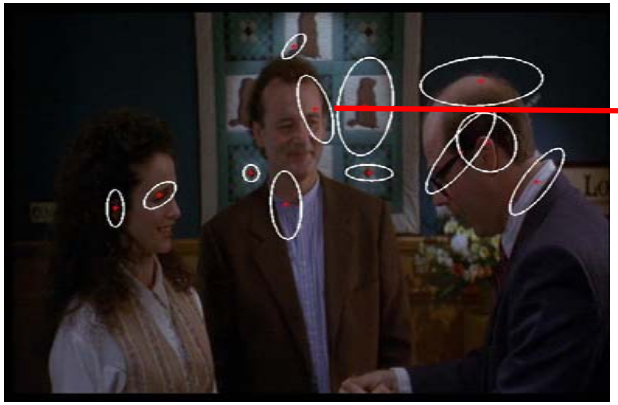


Image

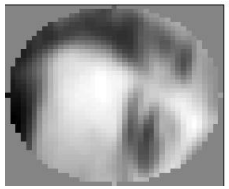


Set of visual words

Offline: Assign visual words and compute histograms for each image

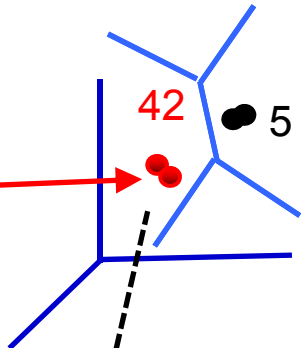


Detect patches

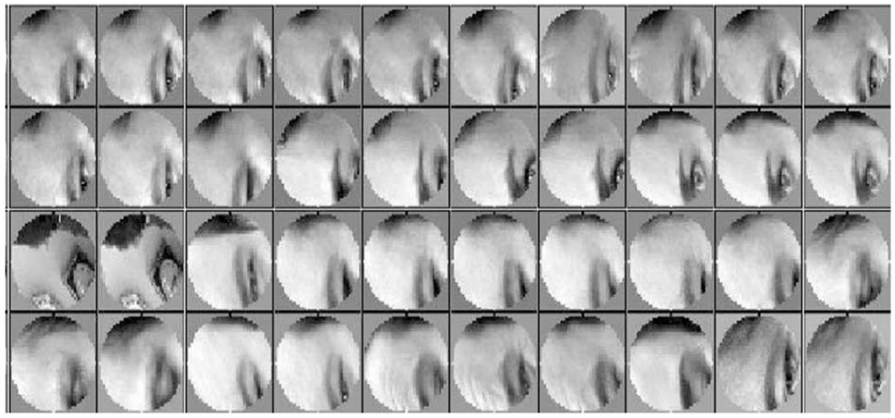


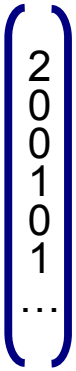
Normalize patch

Compute SIFT descriptor



Find nearest cluster center



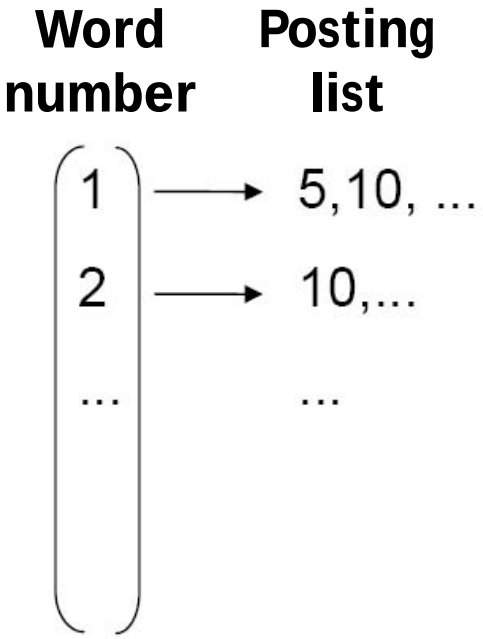


Represent image as a sparse histogram of visual word occurrences

Offline: create an index

frame #5

frame #10



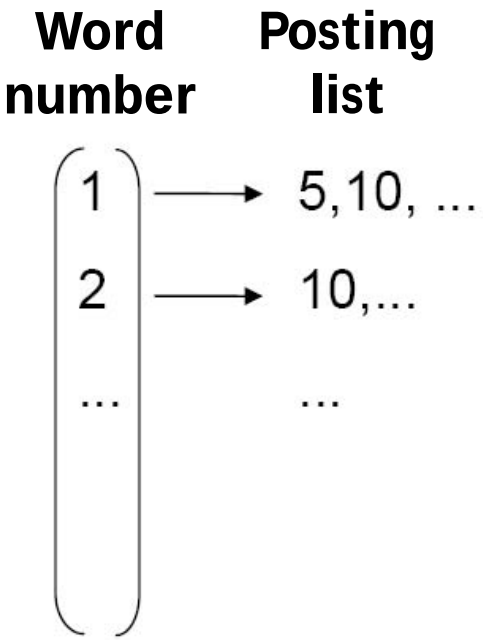
- For fast search, store a “posting list” for the dataset
- This maps visual word occurrences to the images they occur in

(i.e. like the “book index”)

At run time

frame #5

frame #10

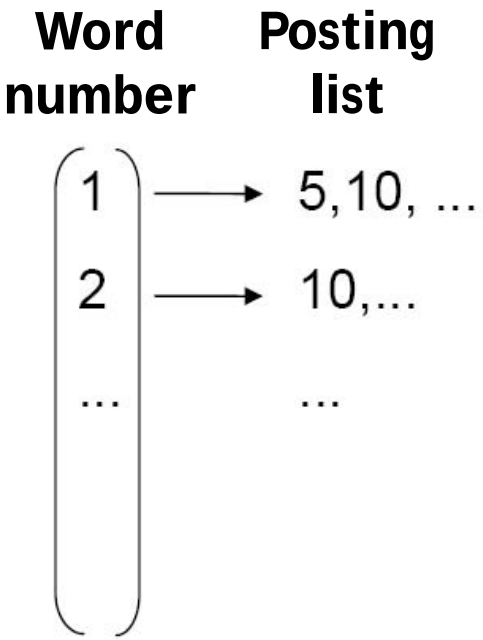


- User specifies a query region
- Generate a short-list of images with visual words in the region
  1. Accumulate all visual words within the query region
  2. Use “book index” to find other images with these words

At run time

frame #5

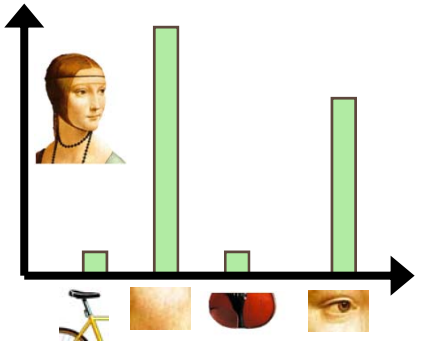
frame #10

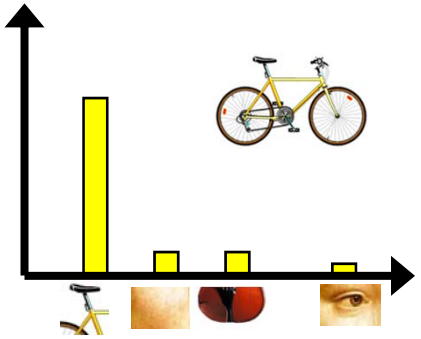


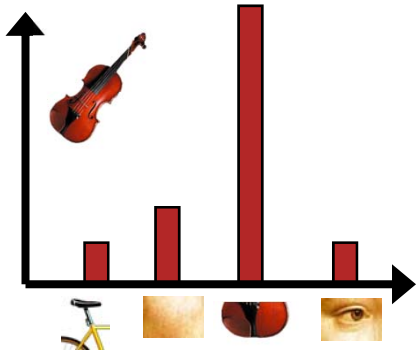
- Score each image by the (weighted) number of common visual words (tentative correspondences)
- Worst case complexity is linear in the number of images N
- In practice, it is linear in the length of the lists (<< N)

Another interpretation: Bags of visual words

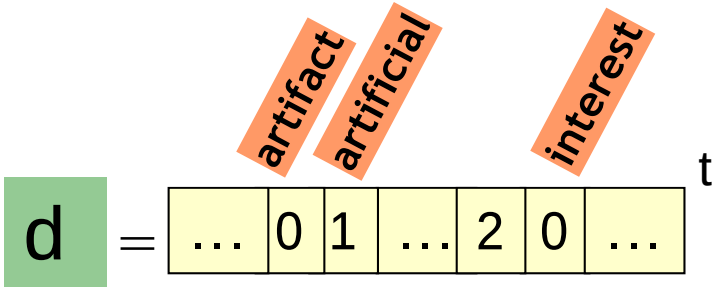
Summarize entire image based on its distribution (histogram) of visual word occurrences







Analogous to bag of words representation commonly used for text documents



Hofmann 2001

Slide: Grauman&Leibe, Image: L. Fei-Fei

Another interpretation: the bag-of-visual-words model

For a vocabulary of size K, each image is represented by a K-vector

$$\mathbf{v}_d = (t_1, \ldots, t_i, \ldots, t_K)^\top$$

where t_i is the number of occurrences of visual word i

Images are ranked by the normalized scalar product between the query vector v_q and all vectors in the database v_d:

$$f_d = \frac{\mathbf{v}_q{}^\top \mathbf{v}_d}{\|\mathbf{v}_q\|_2 \ \|\mathbf{v}_d\|_2}$$

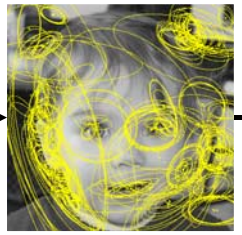
Scalar product can be computed efficiently using inverted file

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

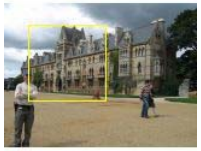
sparse frequency vector

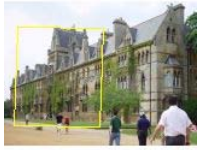
Inverted file

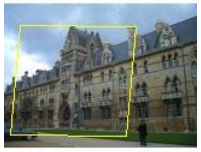
querying

Results

1 

3 

2 

4 

3 

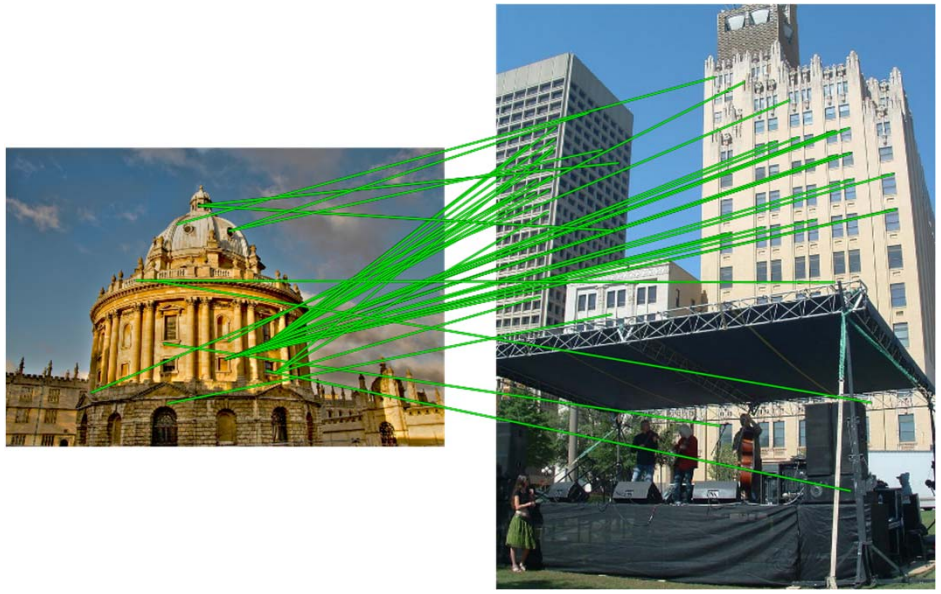
5 

Re-ranked list ← **Geometric verification** ← ***ranked image short-list***

[Chum & al. 2007]

Geometric verification

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



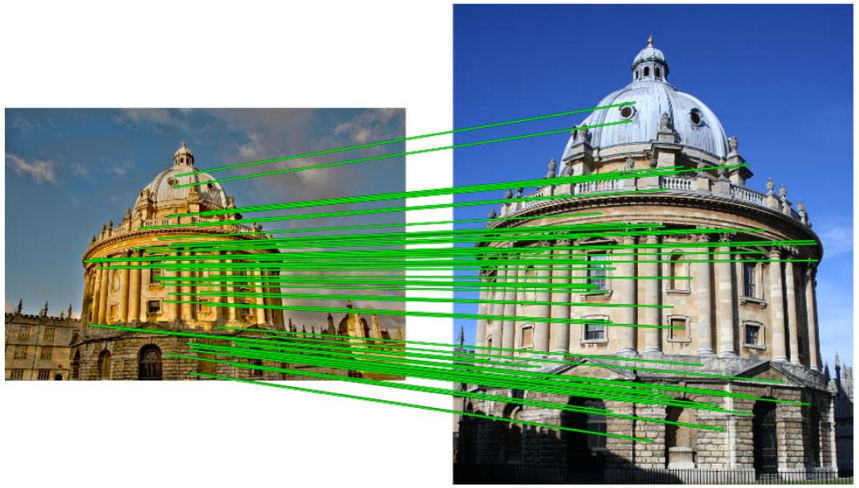
Both images have many matches – which is correct?

Geometric verification

- Remove outliers, many matches are incorrect
- 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, re-rank

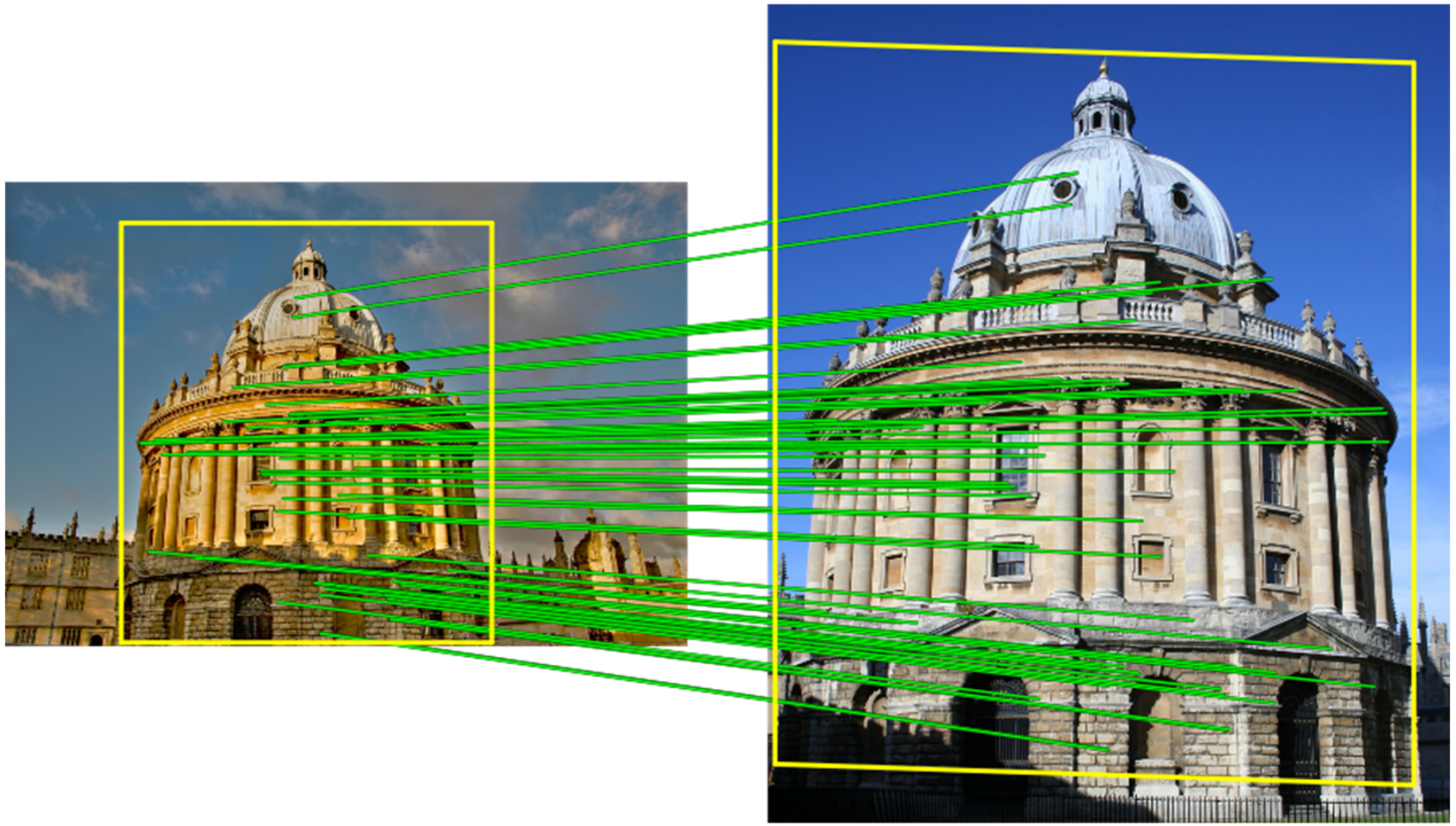


Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**

Geometric verification

Gives **localization** of the object

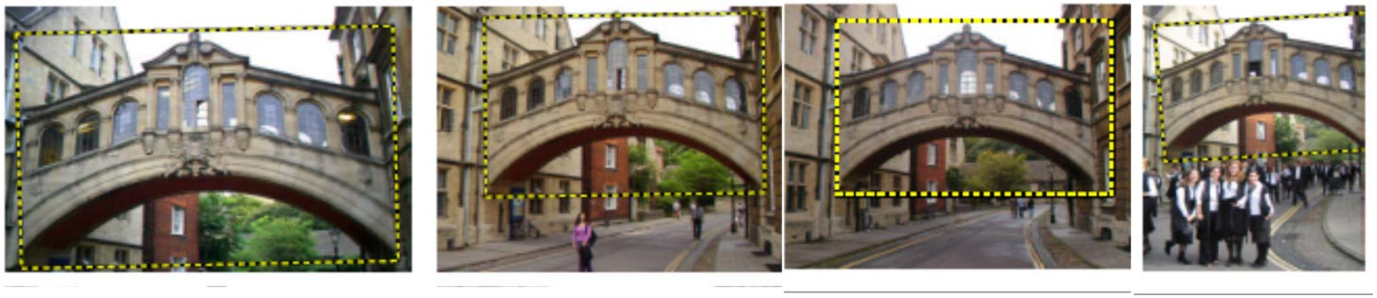


Geometric verification – example

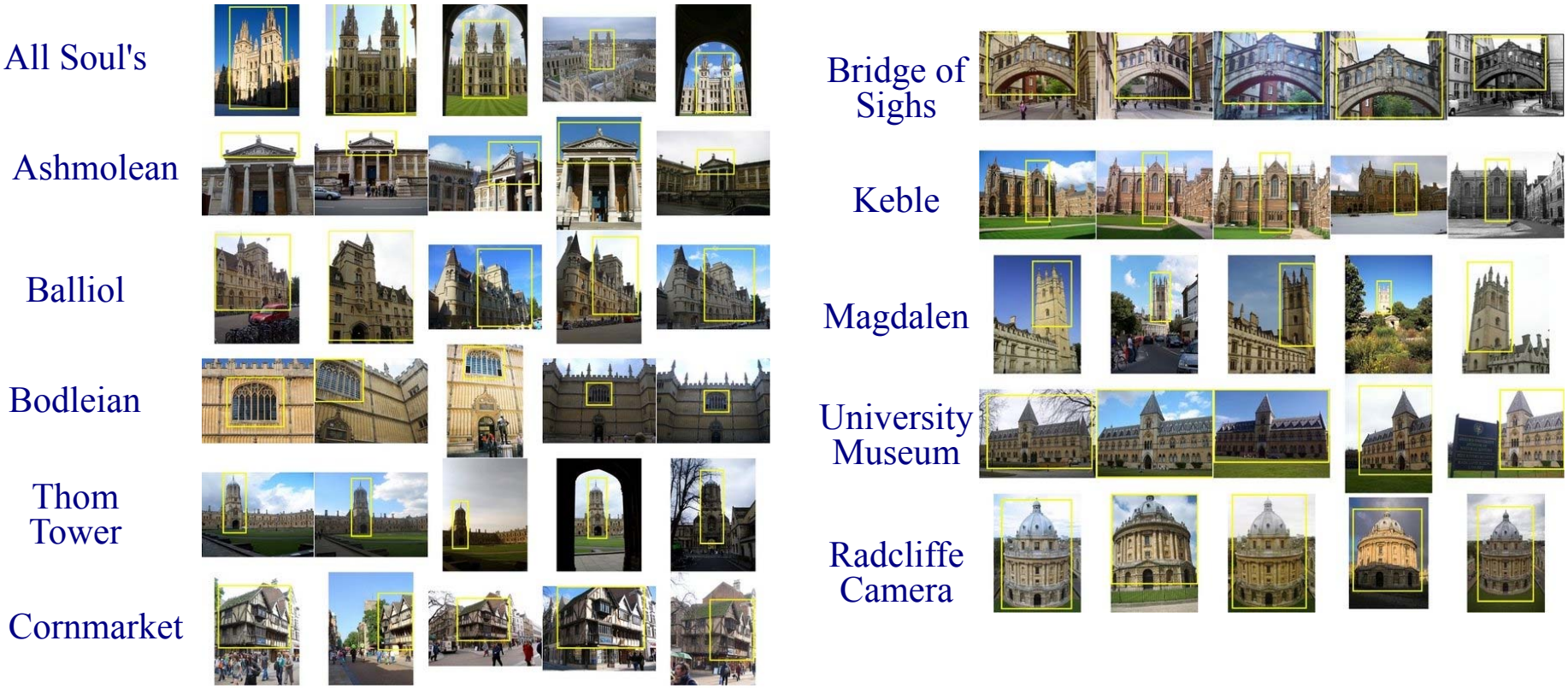
1. Query

2. Initial retrieval set (bag of words model)

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



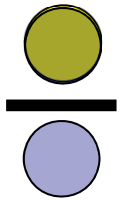
Evaluation dataset: Oxford buildings



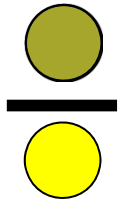
- Ground truth obtained for 11 landmarks
- Evaluate performance by mean Average Precision

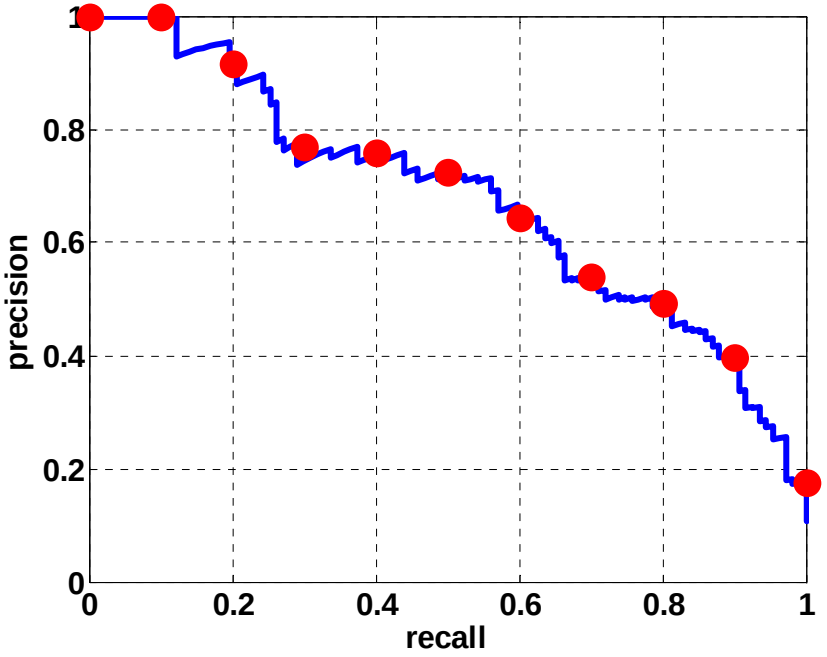
Measuring retrieval performance: Precision - Recall

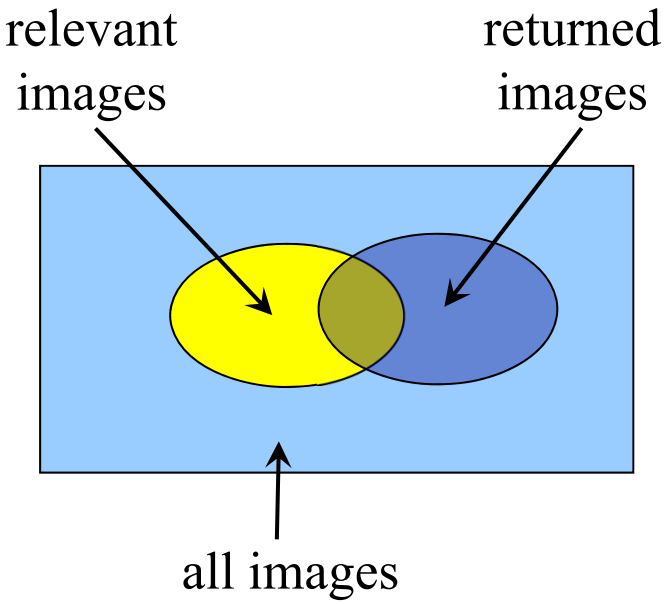
- Precision: % of returned images that are relevant



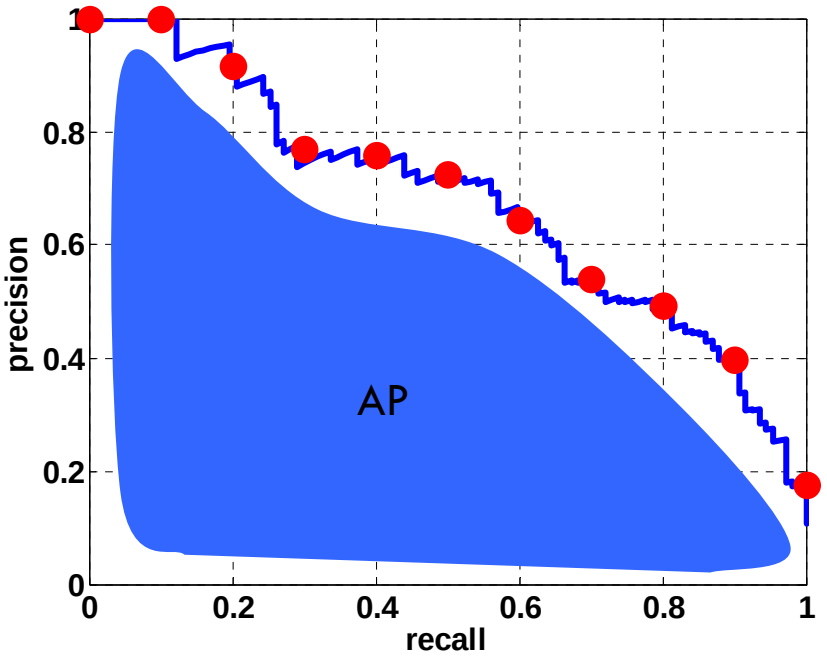
- Recall: % of relevant images that are returned





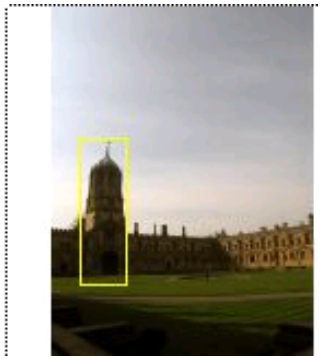


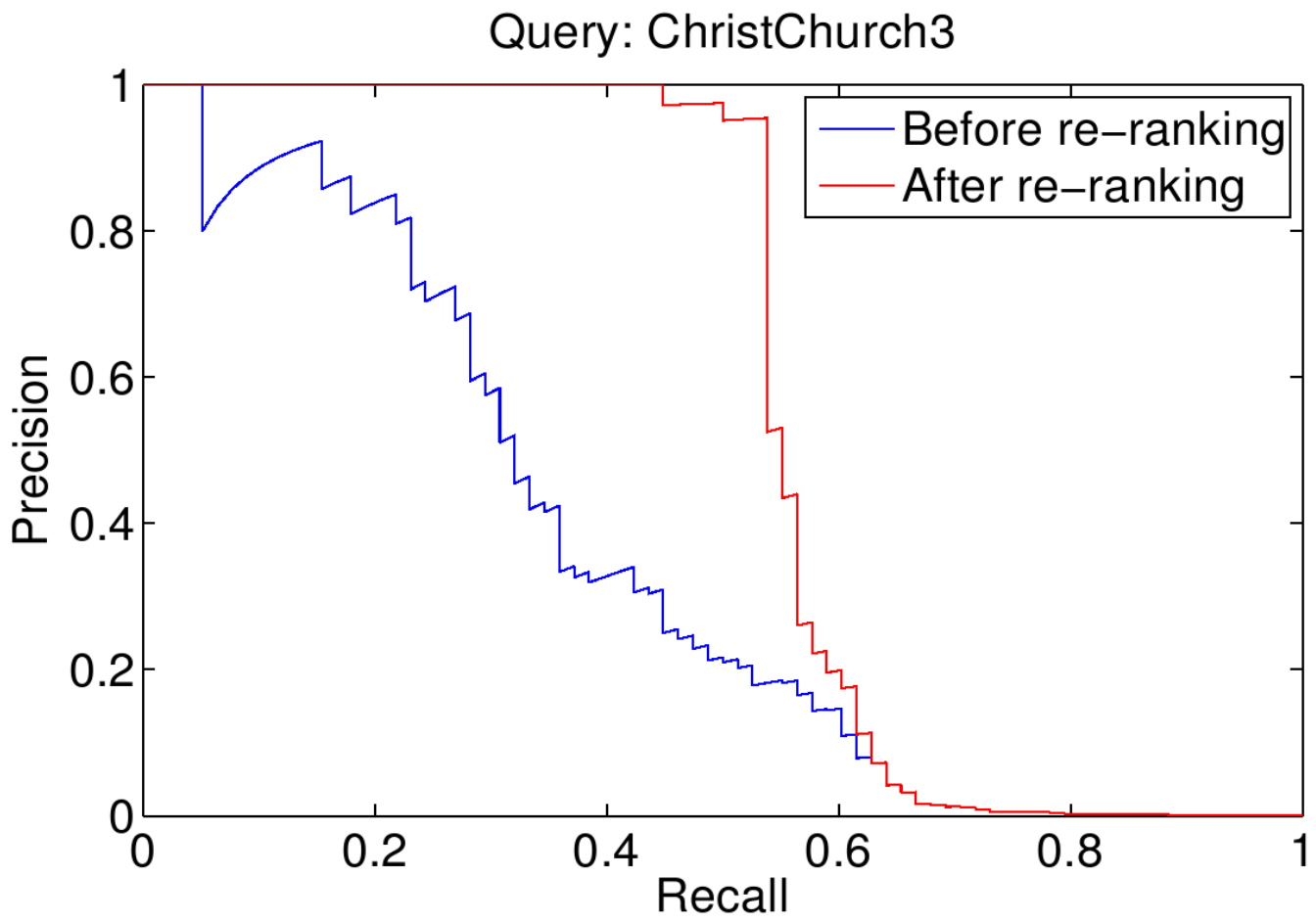
Average Precision



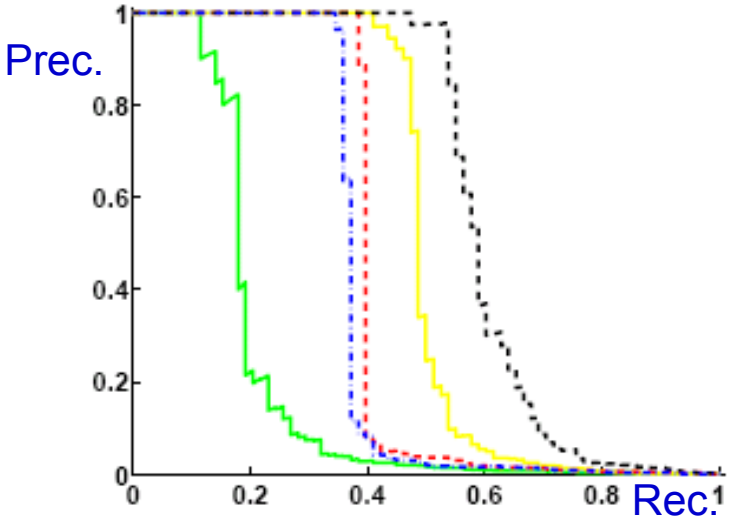
- A good AP score requires both high recall and high precision
- Application-independent

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



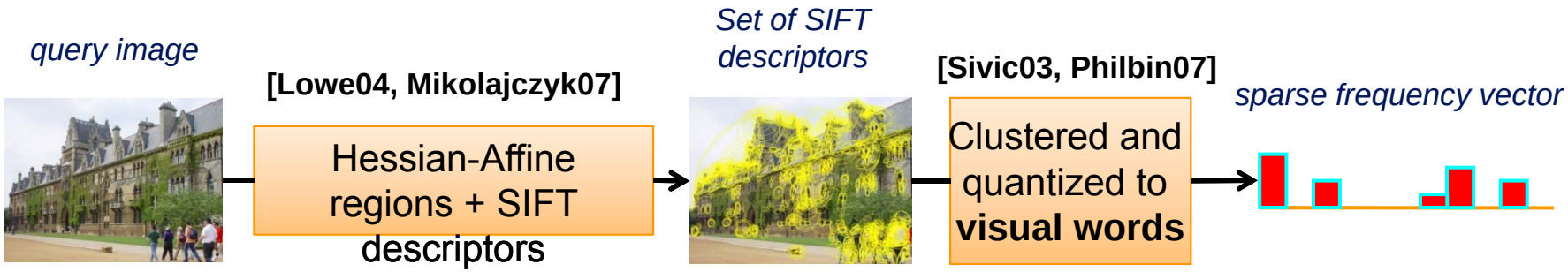

Query: ChristChurch3
Before re−ranking
After re−ranking
Precision
Recall
0
0.2
0.4
0.6
0.8
1

Query images



- high precision at low recall
- variation in performance over queries
- does not retrieve all instances

Why aren't all objects retrieved?



Obtaining visual words is like a sensor measuring the image

"noise" in the measurement process means that some visual words are missing or incorrect, e.g. due to

- Missed detections
- Changes beyond built in invariance
- Quantization effects

} 1. Query expansion
} 2. Better quantization

Consequence: Visual word in query is missing

Query Expansion in text

In text :

- Reissue top n responses as queries
- Blind relevance feedback
- Danger of topic drift

In vision:

- Reissue spatially verified image regions as queries

Automatic query expansion

Visual word representations of two images of the same object may differ (due to e.g. detection/quantization noise) resulting in missed returns

Initial returns may be used to add new relevant visual words to the query

Strong spatial model prevents ‘drift’ by discarding false positives

[Chum, Philbin, Sivic, Isard, Zisserman, ICCV’07;

Chum, Mikulik, Perdoch, Matas, CVPR’11]

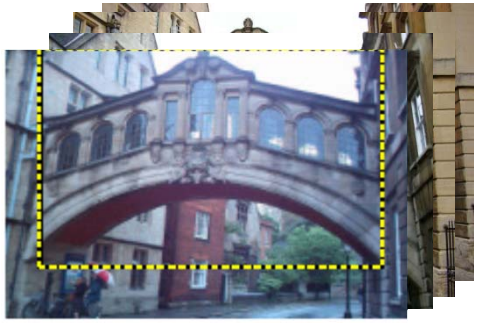
Visual query expansion - overview

1. Original query

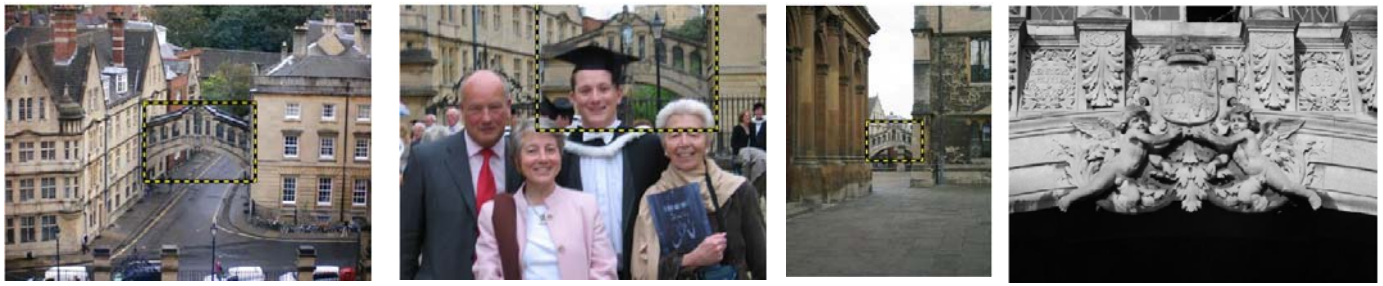
2. Initial retrieval set

3. Spatial verification

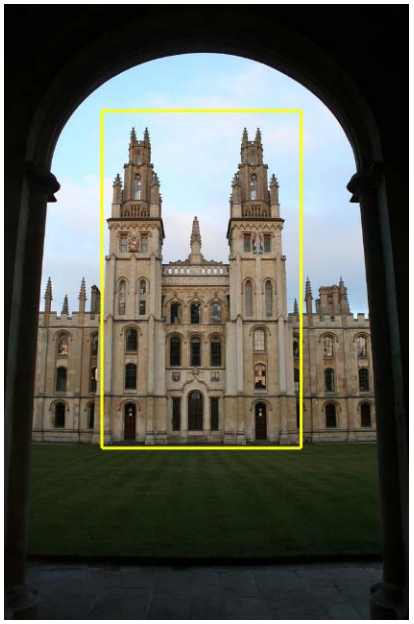
4. New enhanced query



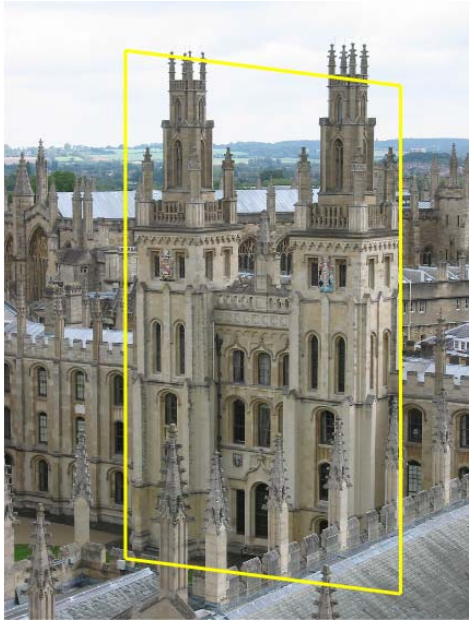
5. Additional retrieved images



Query Expansion



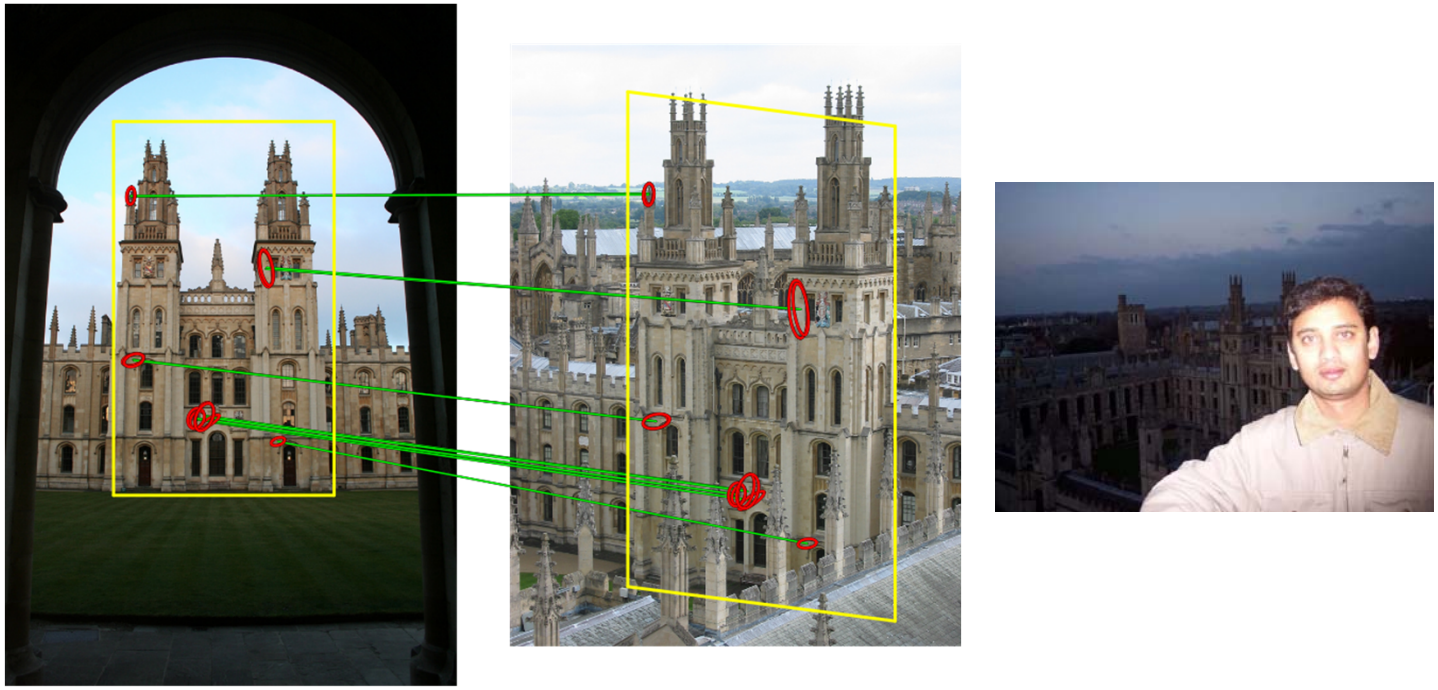
Query Image



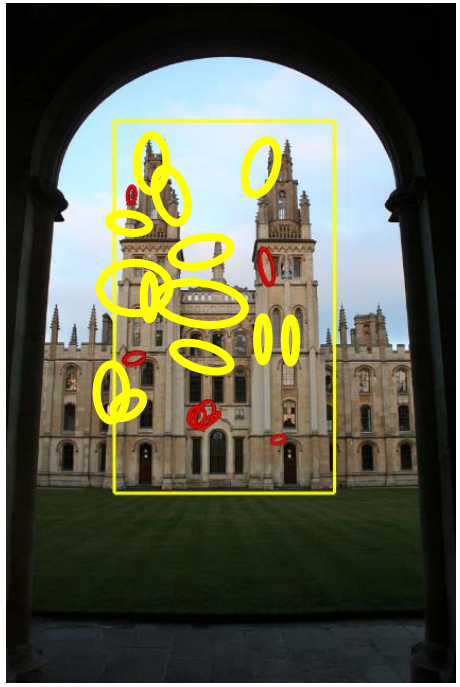
Originally retrieved image

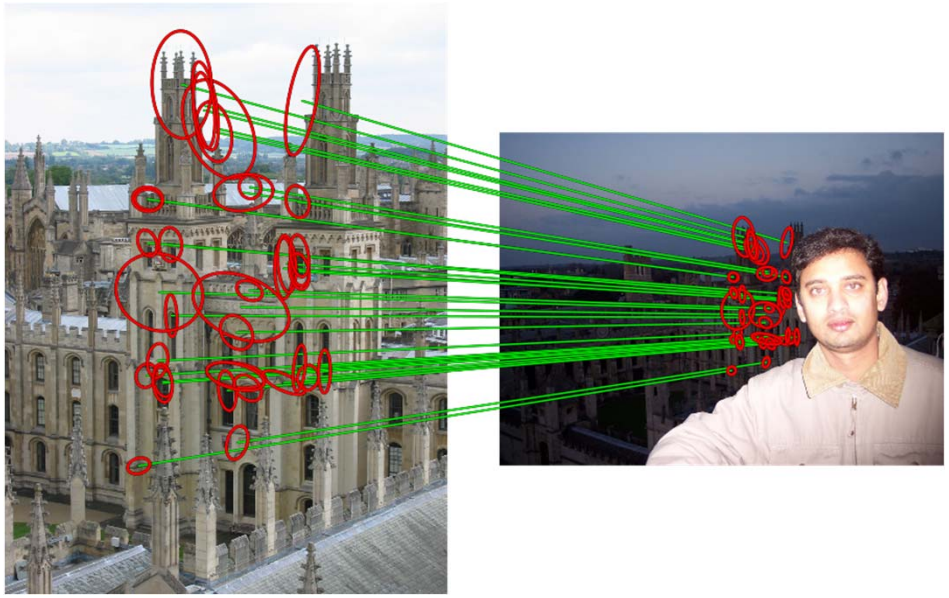
Originally not retrieved

Query Expansion

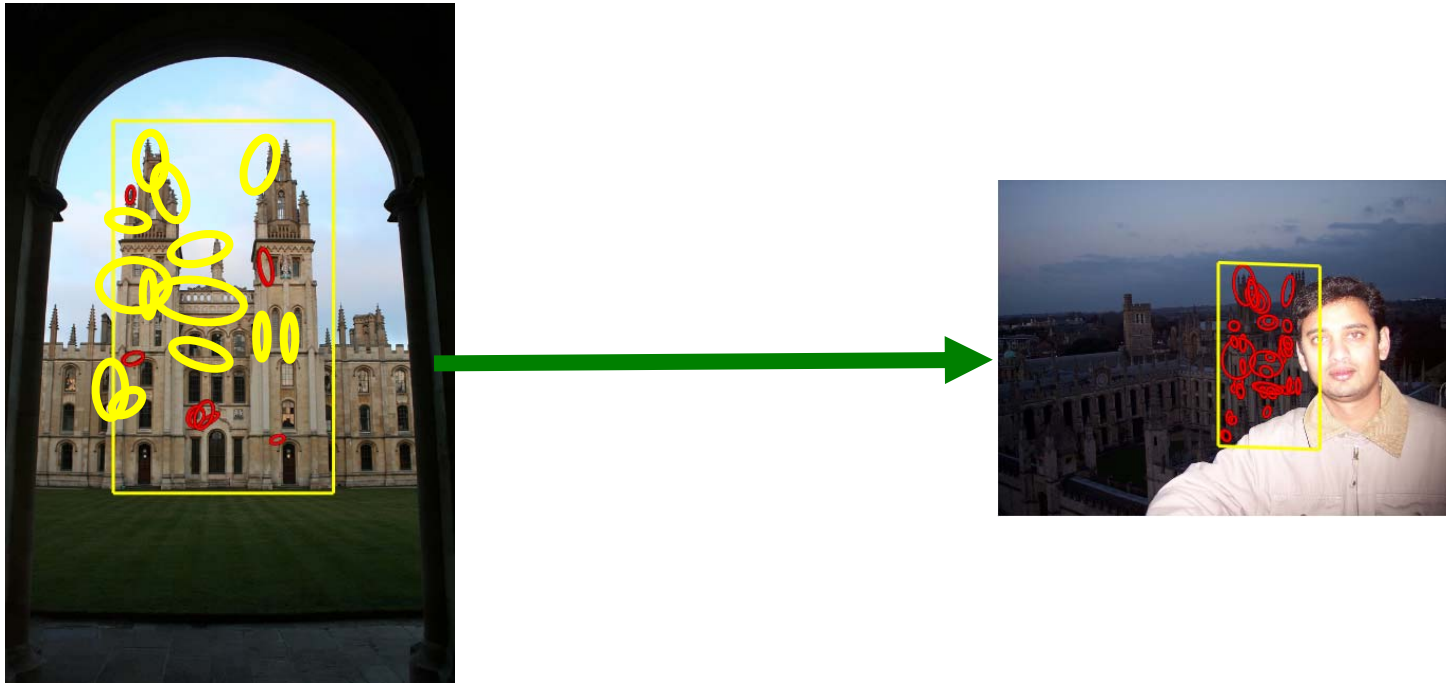


Query Expansion



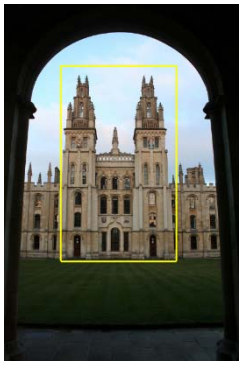


Query Expansion

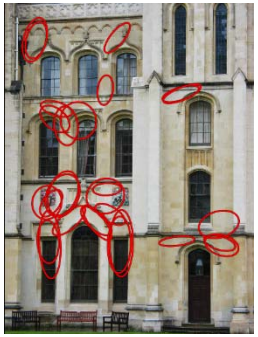
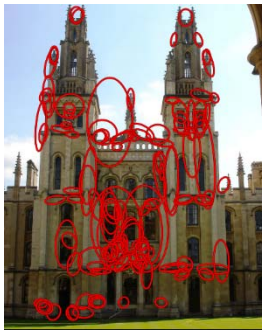
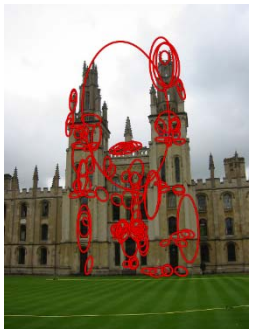


Query Expansion

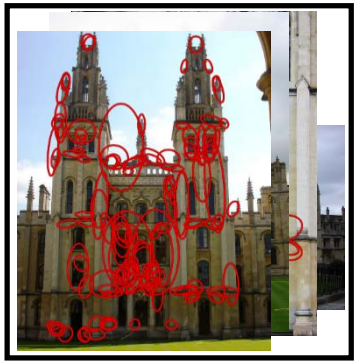
Query Image



Spatially verified retrievals with matching regions overlaid


...

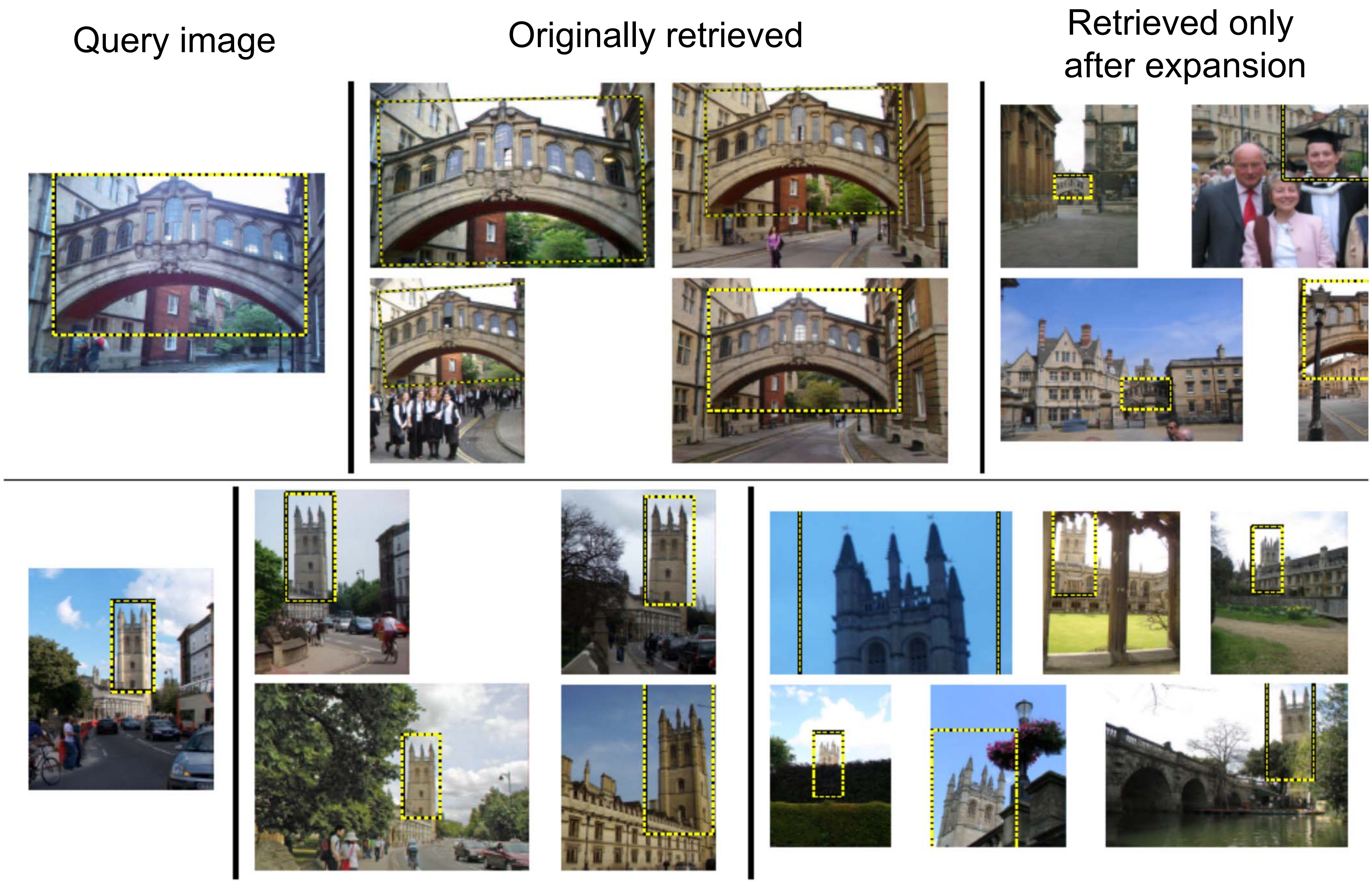


New expanded query

New expanded query is formed as

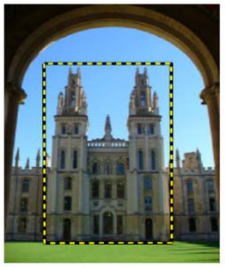
- the average of visual word vectors of spatially verified returns
- only inliers are considered
- regions are back-projected to the original query image

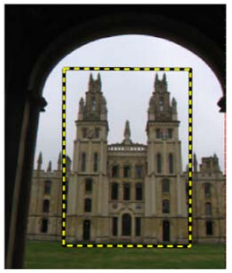
Query Expansion

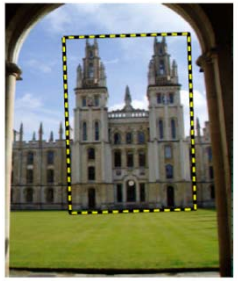


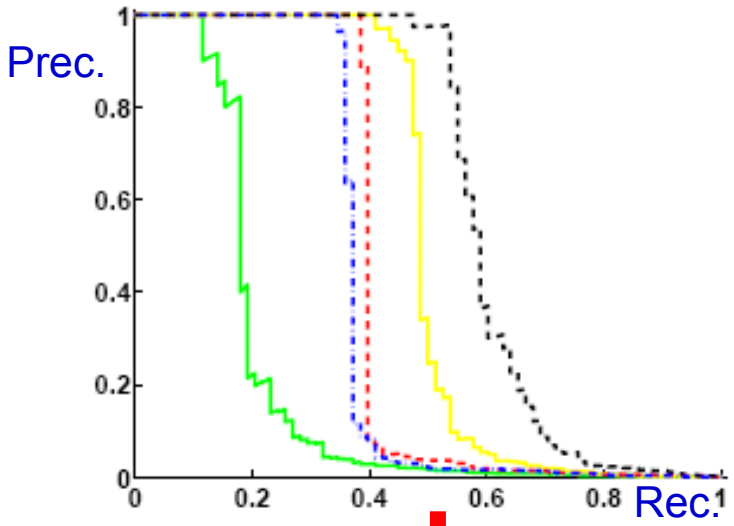
Query image

Original results

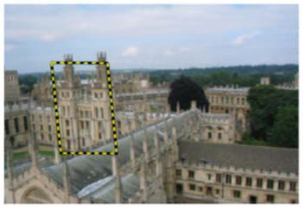


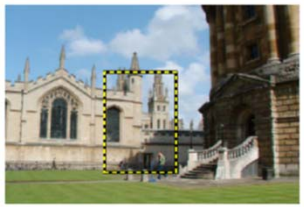


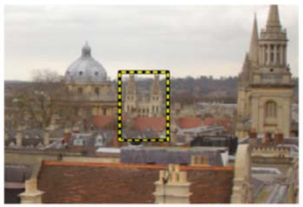


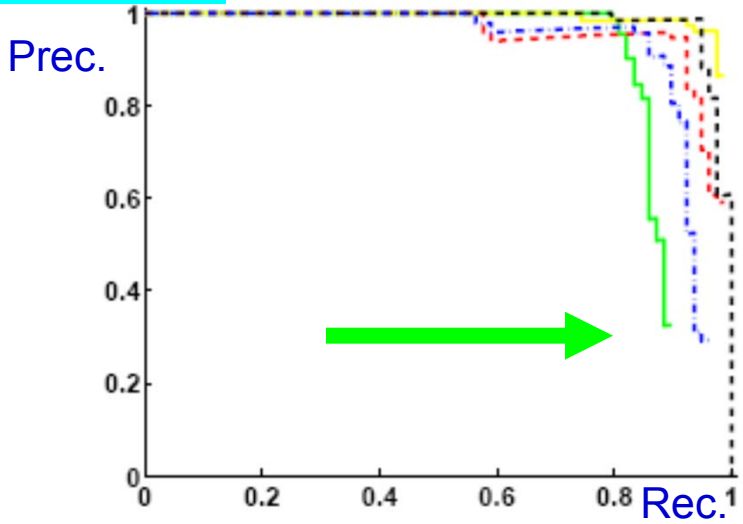


Expanded results (improved)





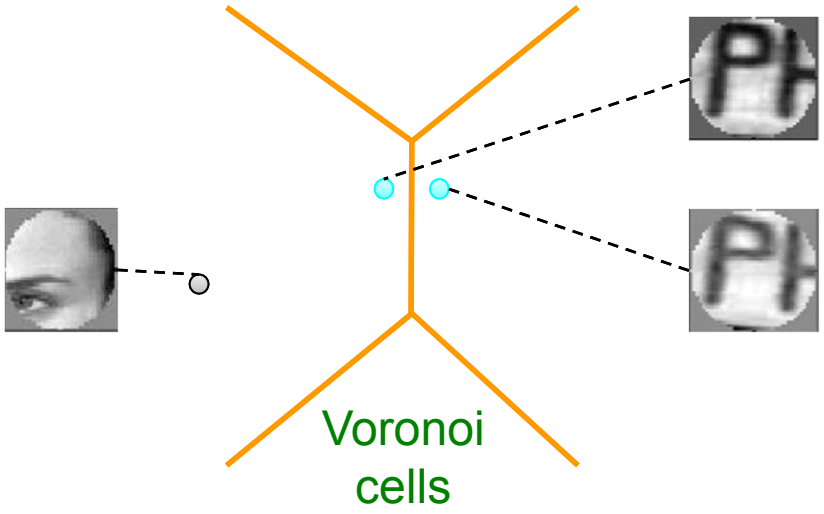




Quantization errors

Typically, quantization has a significant impact on the final performance of the system **[Sivic03,Nister06,Philbin07]**

Quantization errors split features that should be grouped together and confuse features that should be separated



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

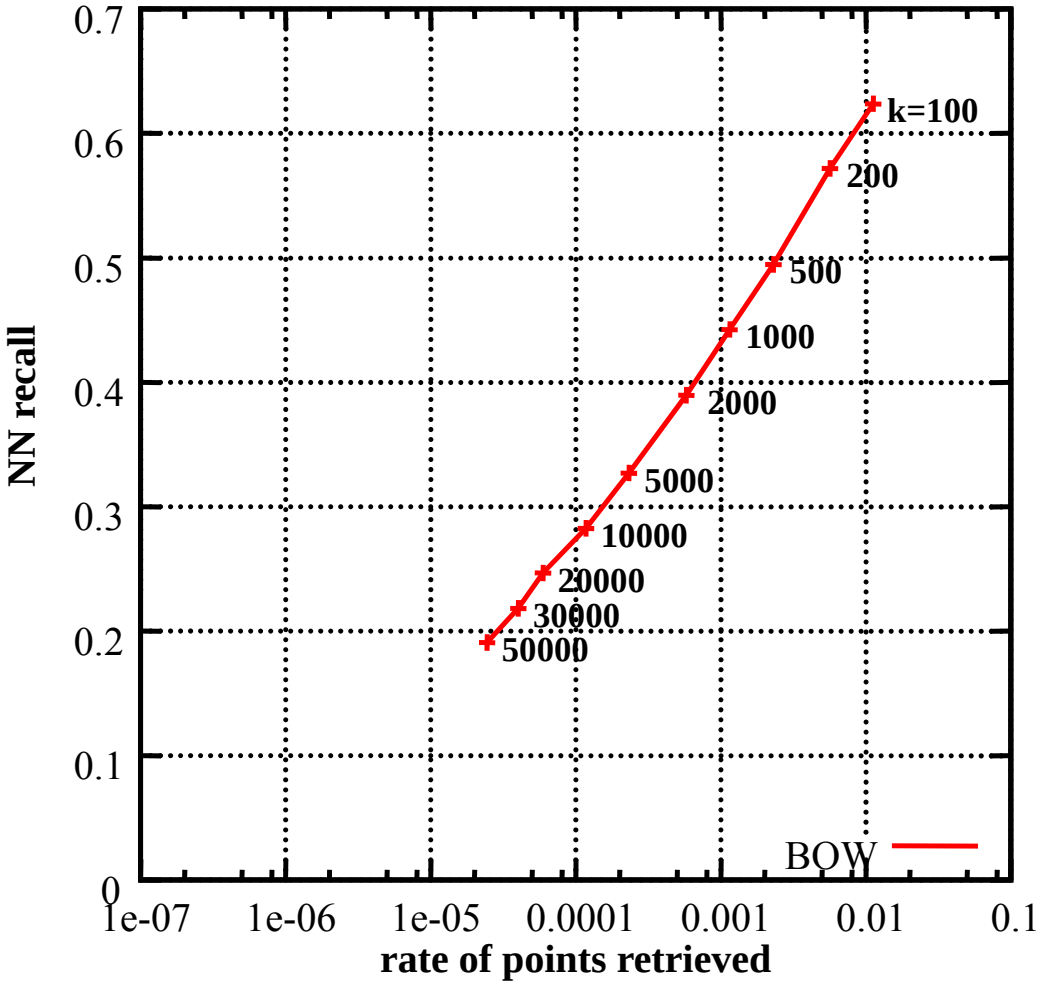
- ANN algorithms usually returns a short-list of nearest neighbors
  - this short-list is supposed to contain the NN with high probability
  - exact search may be performed to re-order this short-list

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

    against
  - **NN precision =** proportion of vectors in the short-list
    - the lower this proportion
      - the more information we have about the vector
      - 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



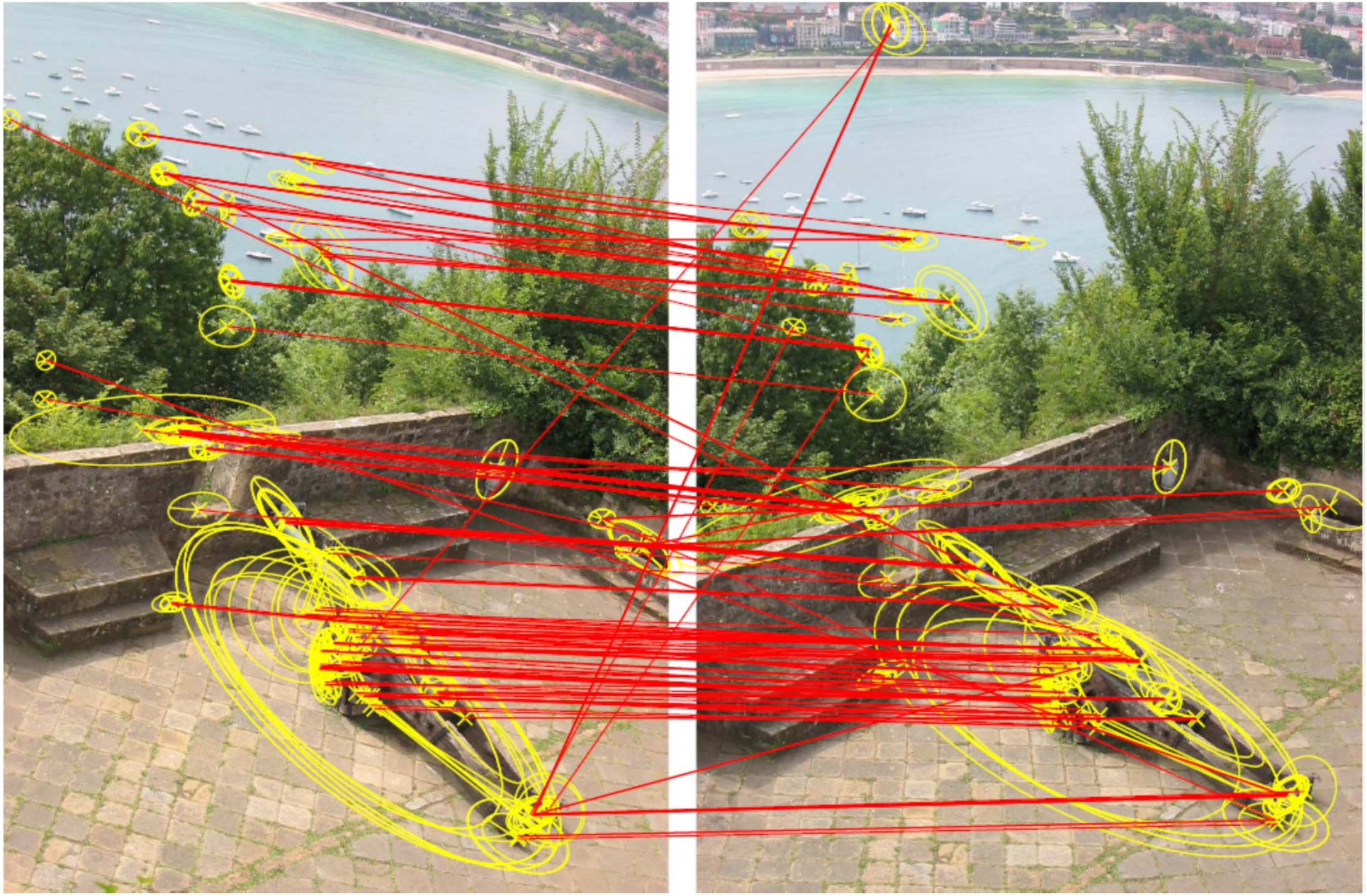
- ANN algorithms returns a list of potential neighbors

- **NN recall**
  = probability that *the* NN is in this list

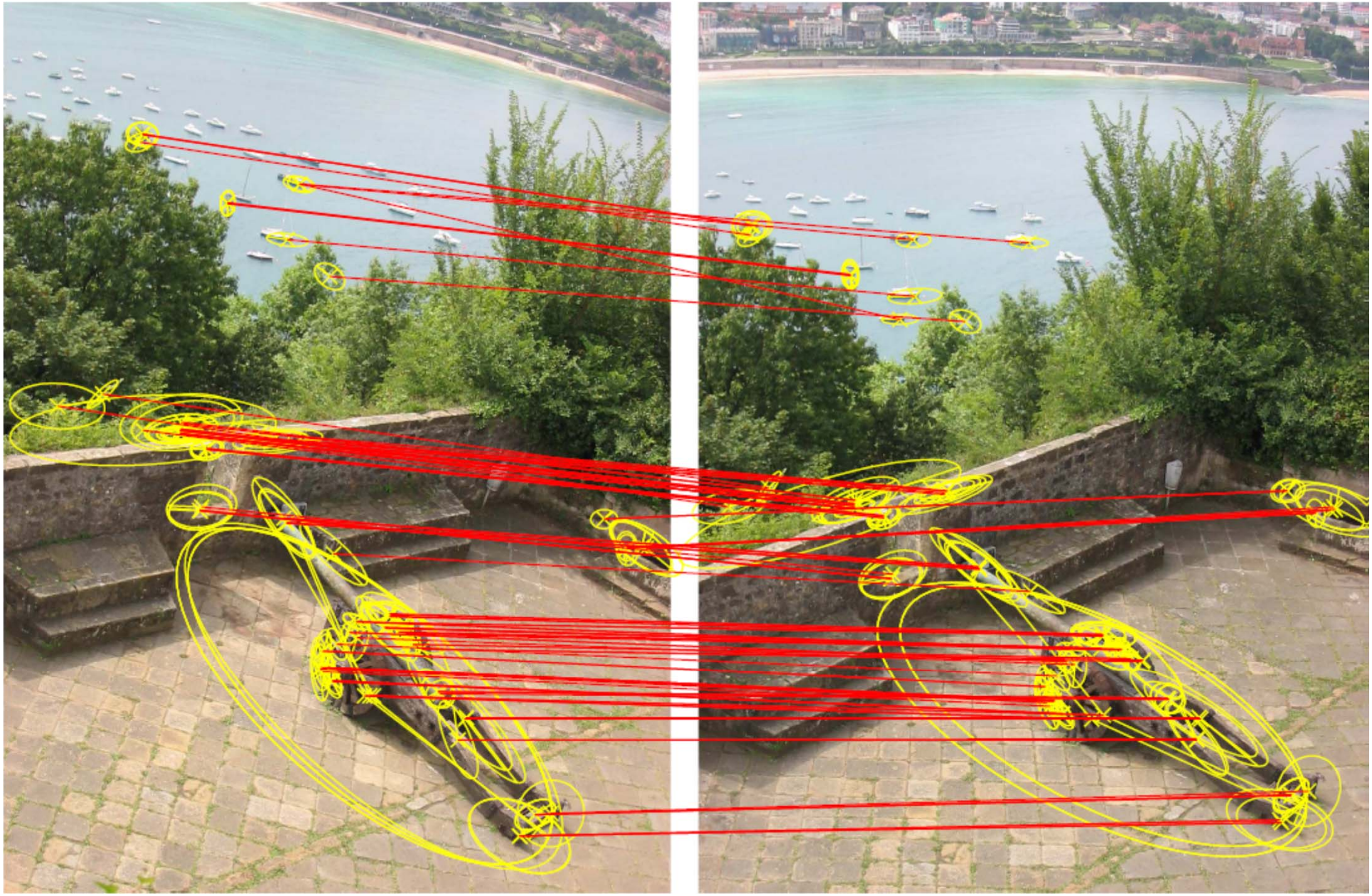
- **NN precision**:
  = proportion of vectors in the short-list

- 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

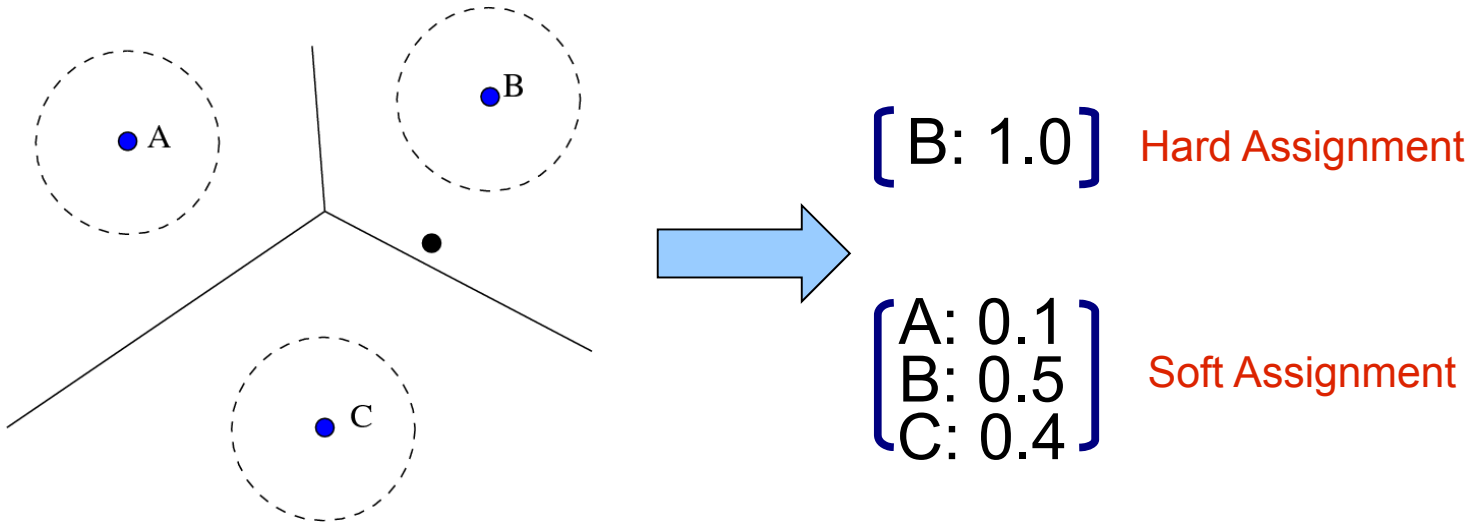
- The matching performed by BOF is weak
  - for a “small” visual dictionary: too many false matches
  - for a “large” visual dictionary: many true matches are missed

- 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]

Beyond bags-of-visual-words

- Soft-assign each descriptor to multiple cluster centers

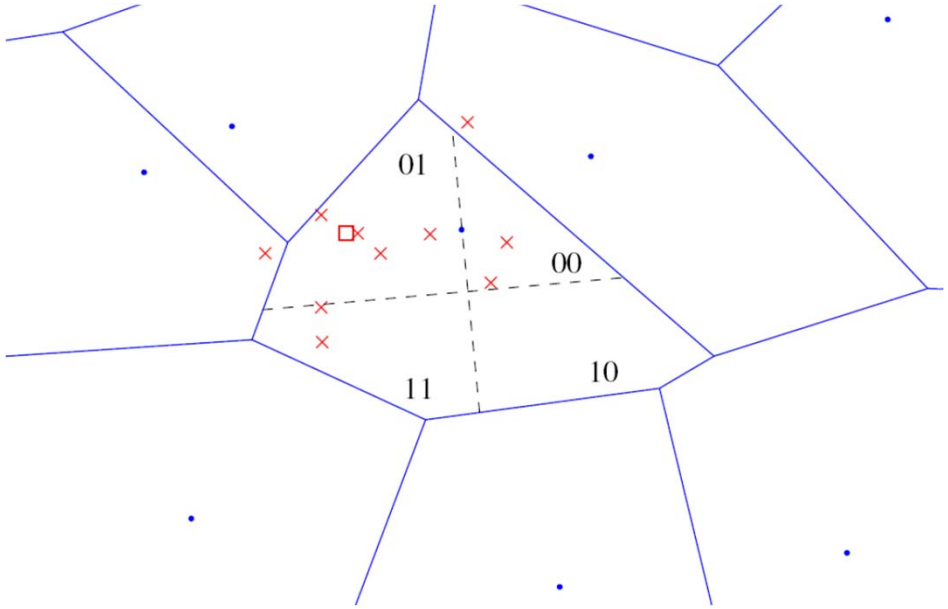
[Philbin et al. 2008, Van Gemert et al. 2008]



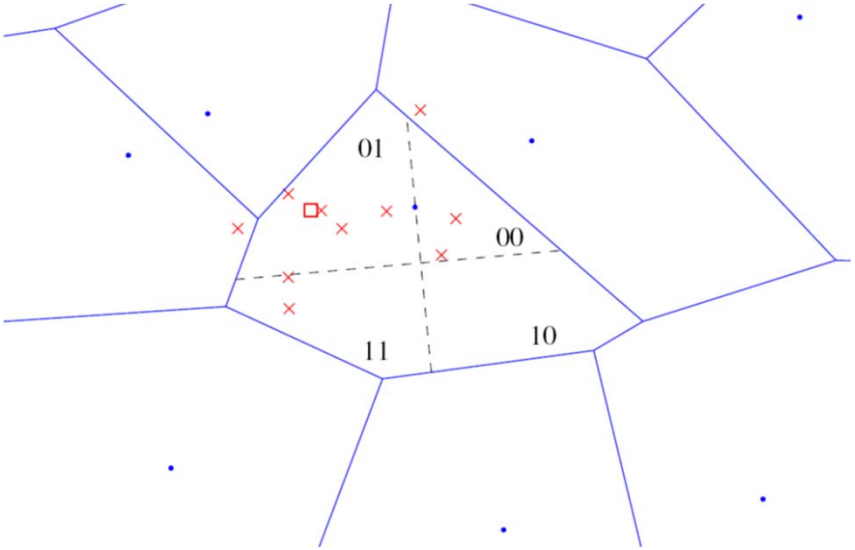
Beyond bag-of-visual-words

Hamming embedding [Jegou et al. 2008]

- Standard quantization using bag-of-visual-words
- Additional localization in the Voronoi cell by a binary signature



Hamming Embedding



Representation of a descriptor *x*

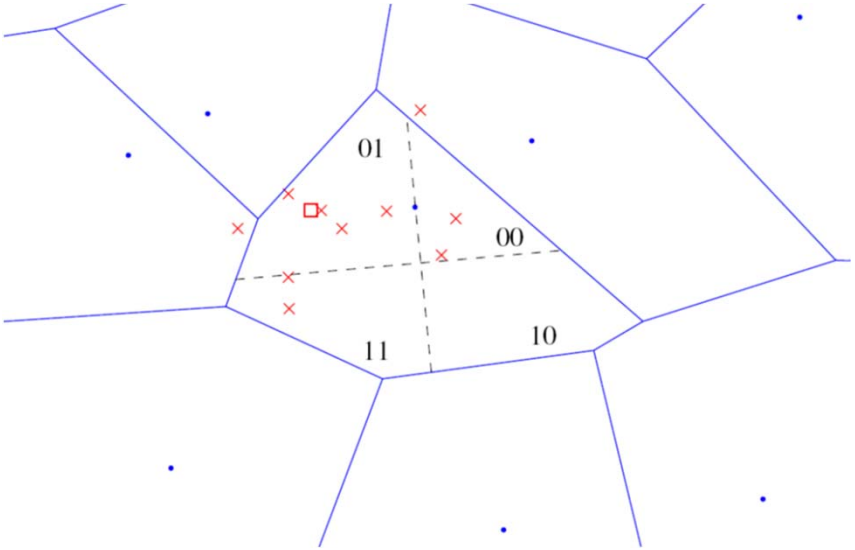
- – Vector-quantized to *q(x)* as in standard BOF
- \+ short binary vector *b(x)* for an additional localization in the Voronoi cell

Two descriptors x and y match iif

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

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

Hamming Embedding



•Nearest neighbors for Hamming distance ≈ those for Euclidean distance
→ 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

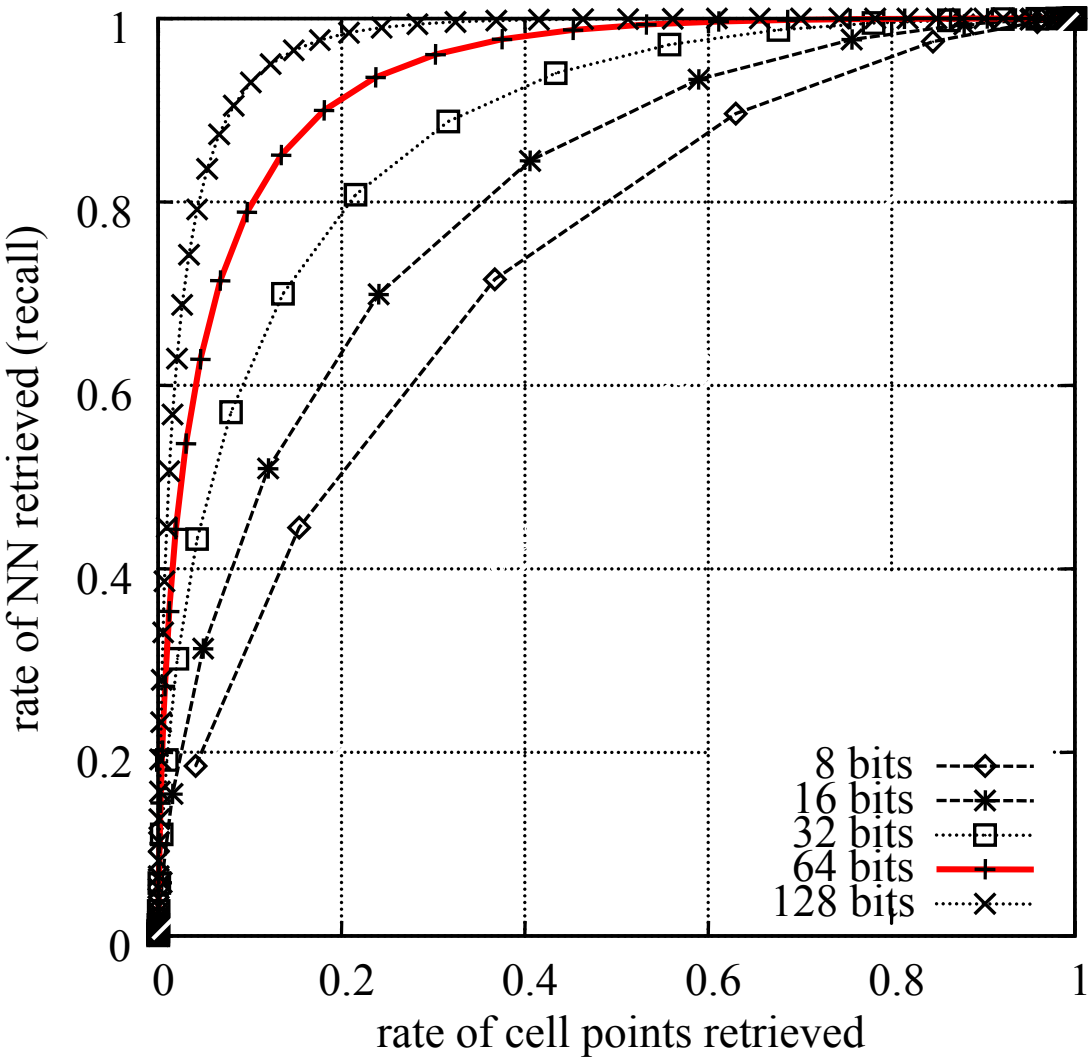
- **Off-line** (given a quantizer)
  - draw an orthogonal projection matrix P of size $d_b \times d$

  $\rightarrow$ this defines d_b random projection directions
  - for each Voronoi cell and projection direction, compute the median value for a training 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, ECCV'08, ICJV'10]

Hamming neighborhood

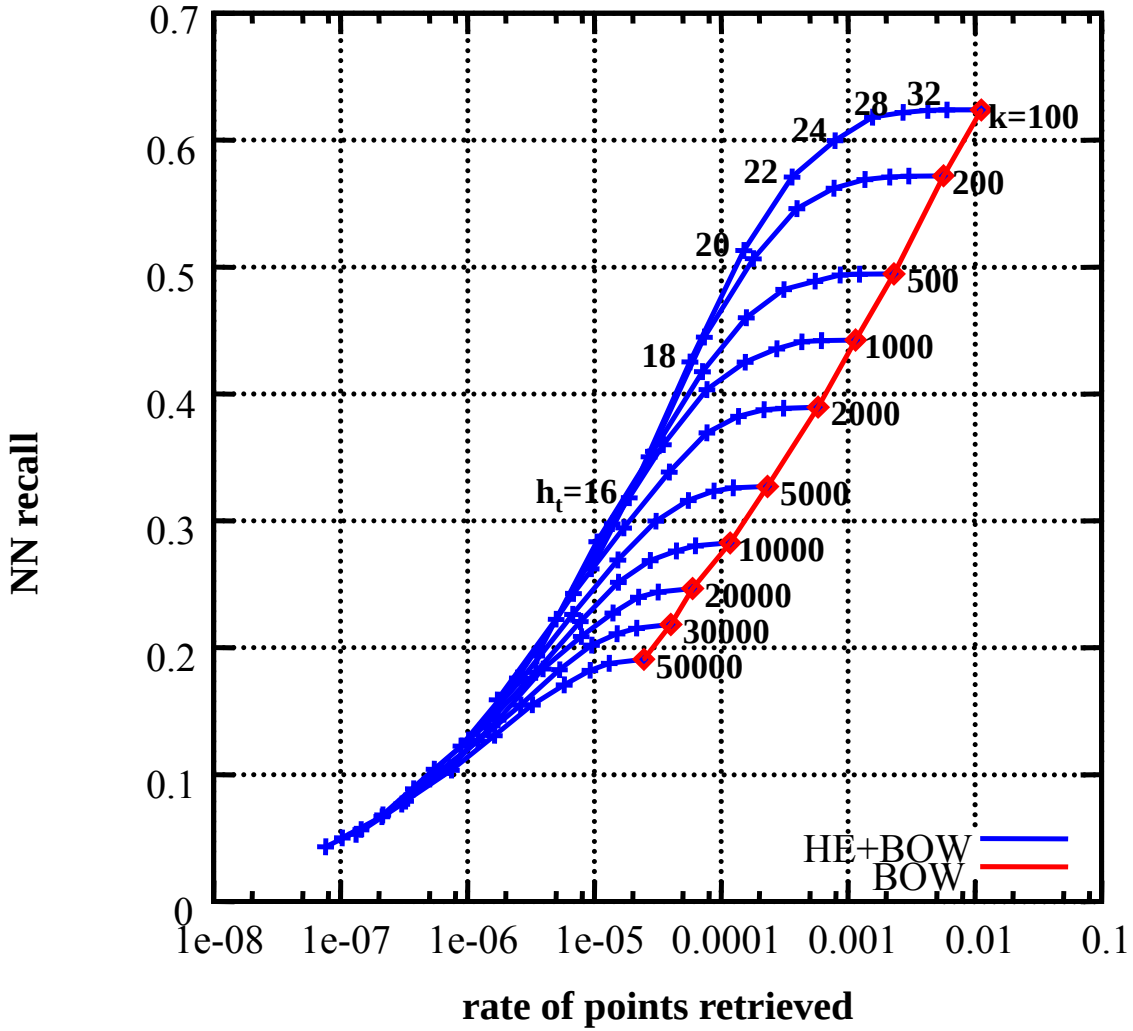


Trade-off between memory usage and accuracy

→More bits yield higher accuracy

In practice, 64 bits (8 byte)

ANN evaluation of Hamming Embedding



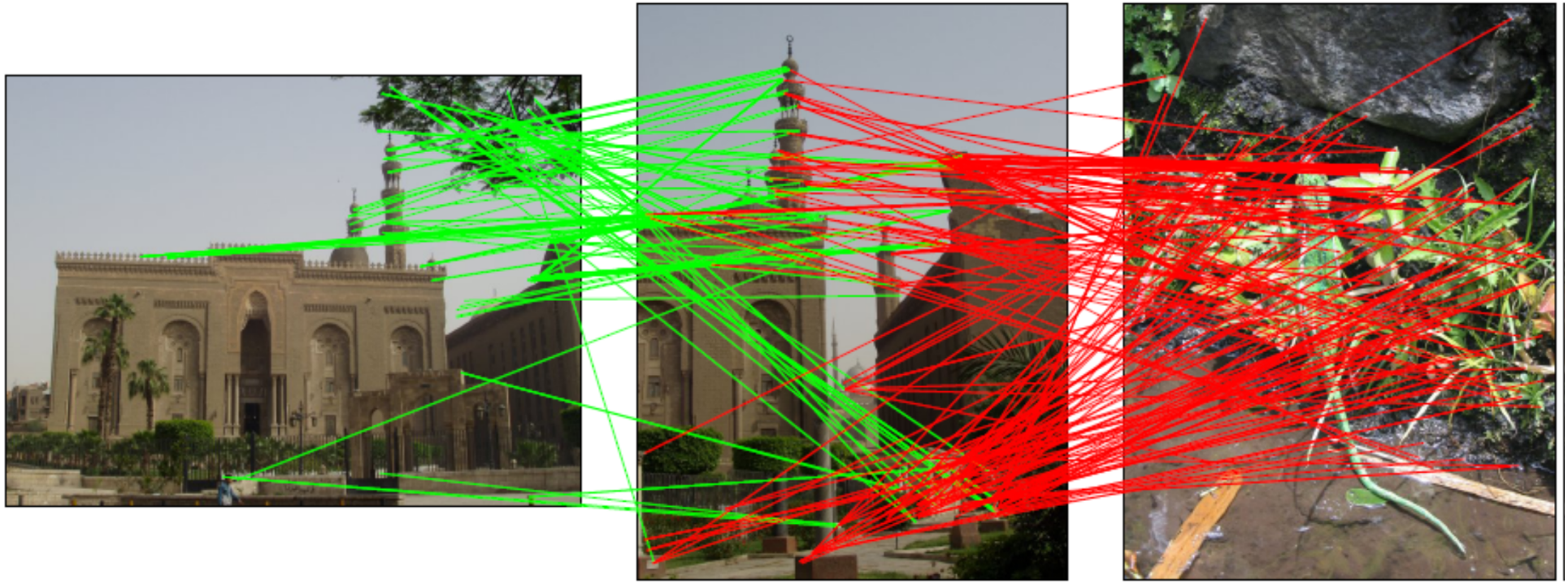
compared to BOW: at least 10 times less points in the short-list for the same level of NN recall

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

Matching points - 20k word vocabulary

201 matches

240 matches

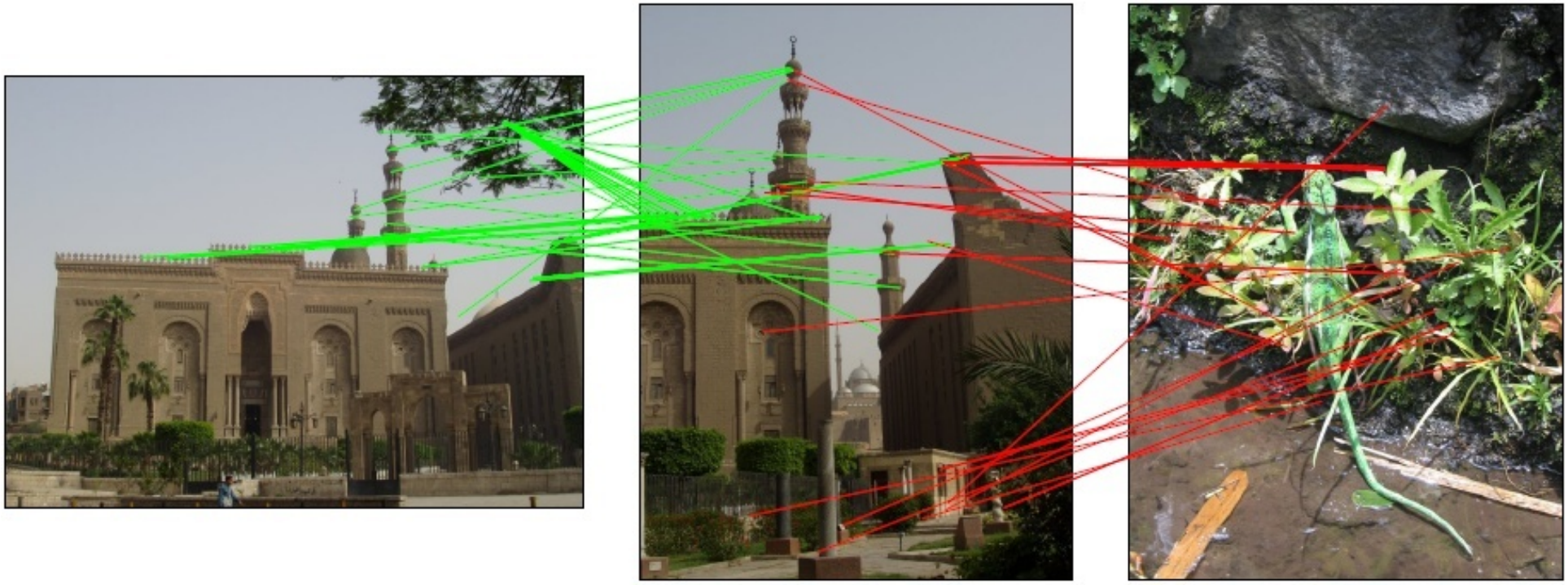


Many matches with the non-corresponding image!

Matching points - 200k word vocabulary

69 matches

35 matches

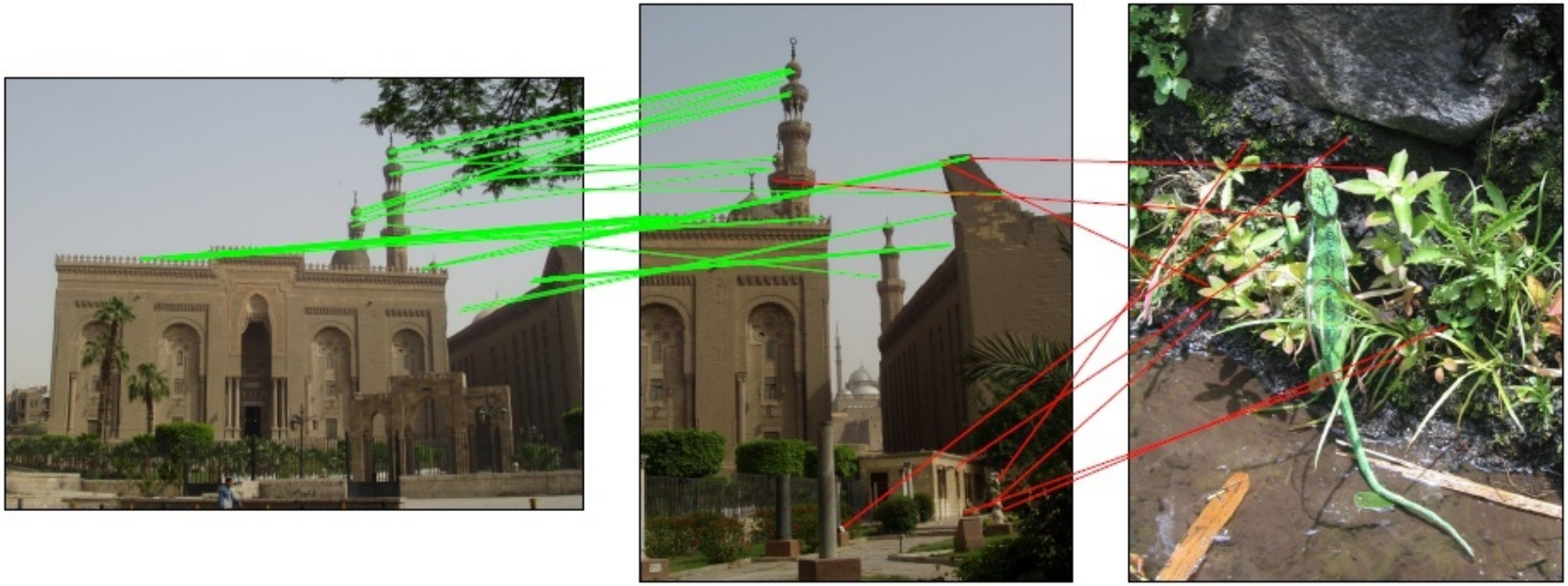


Still many matches with the non-corresponding one

Matching points - 20k word vocabulary + HE

83 matches

8 matches



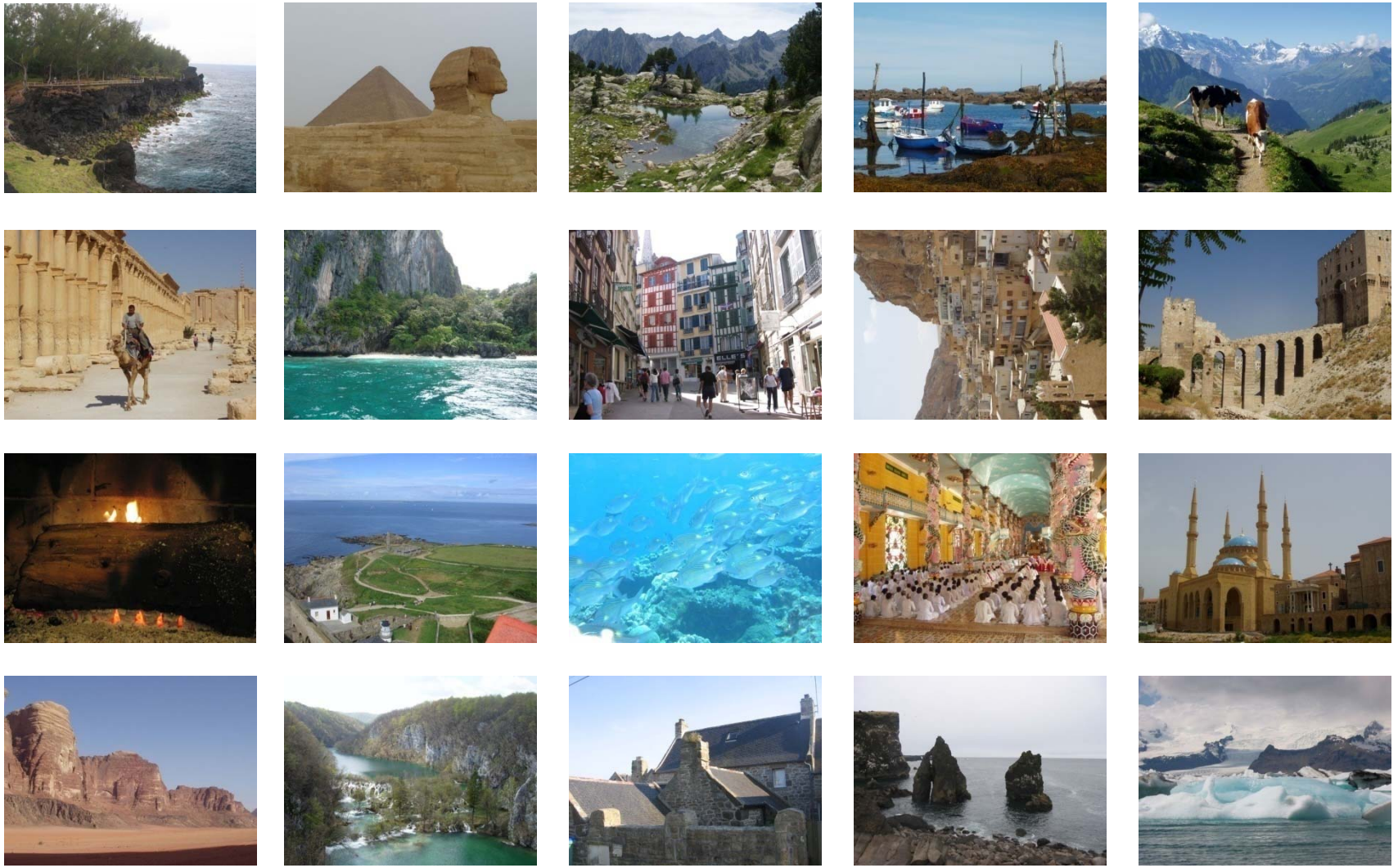
10x more matches with the corresponding image!

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

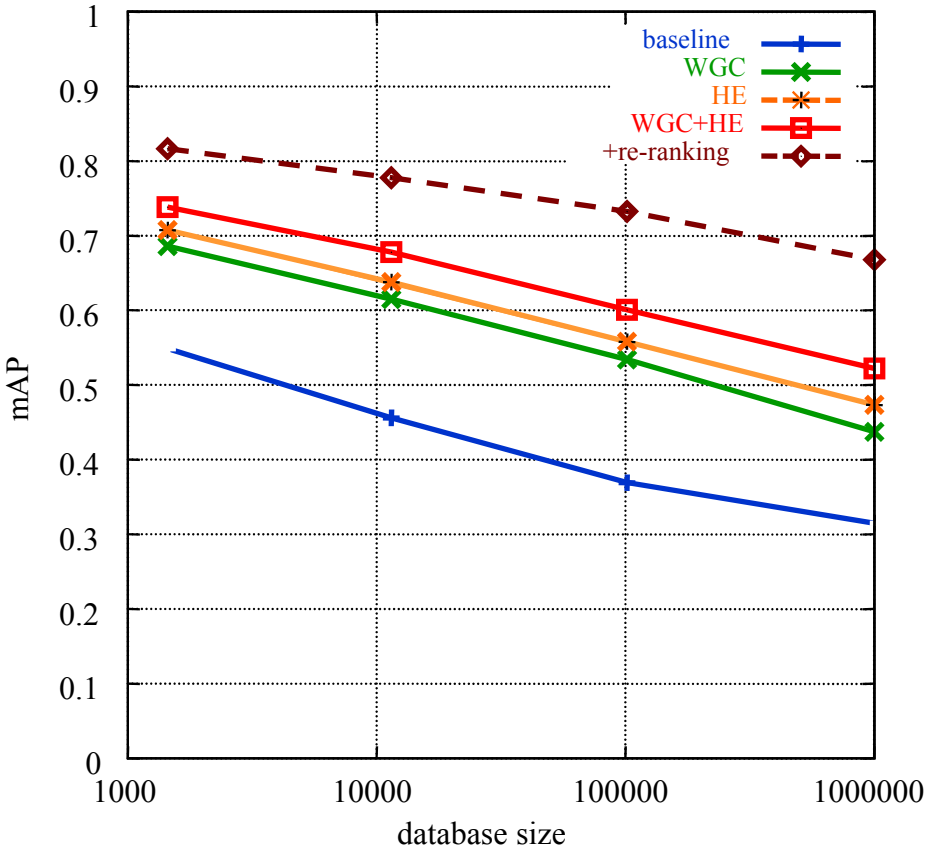






Experimental evaluation

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



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

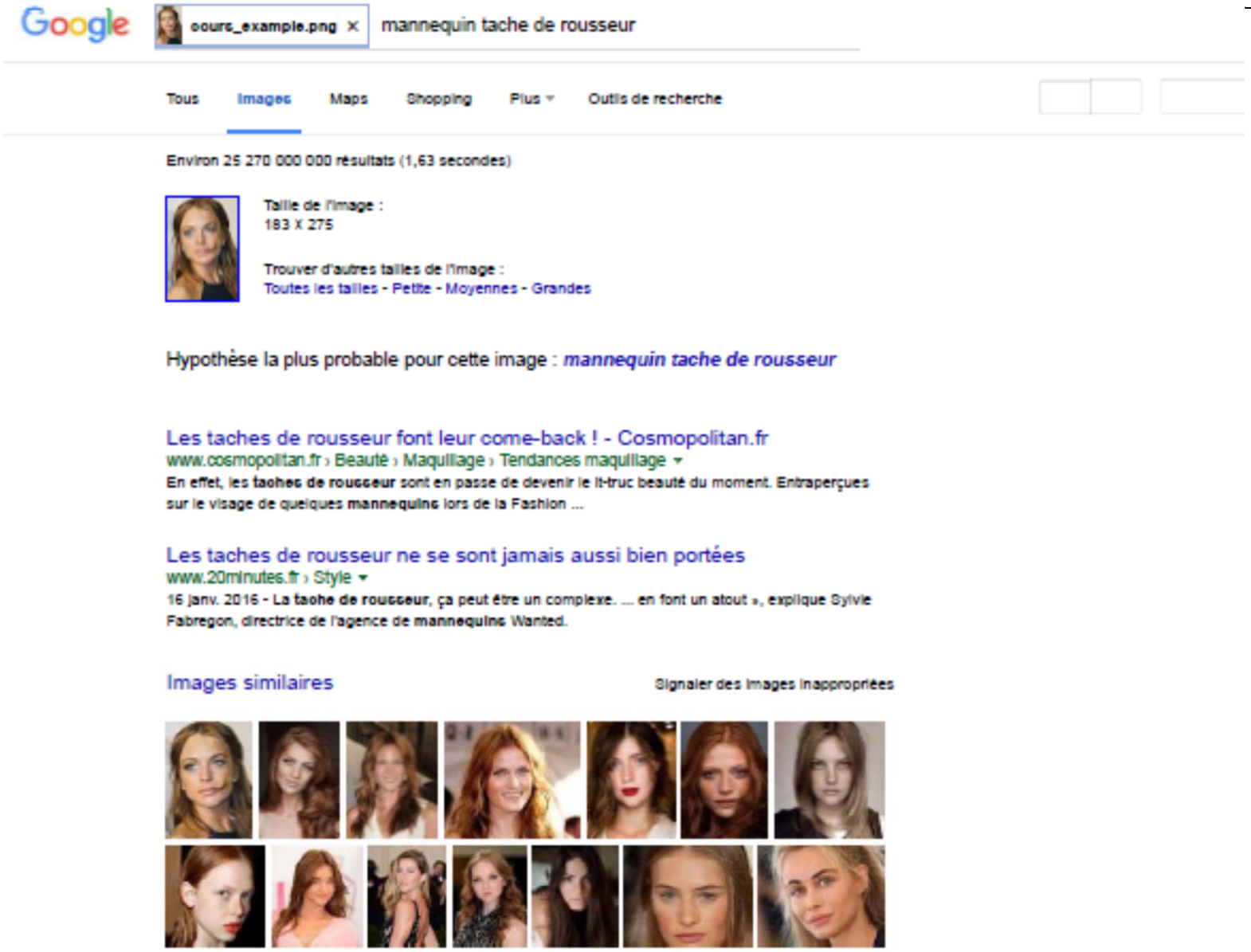
Results – Venice Channel

Image retrieval - products

- Search for places and particular objects
  - For example on a smart phone

Courtesy Google

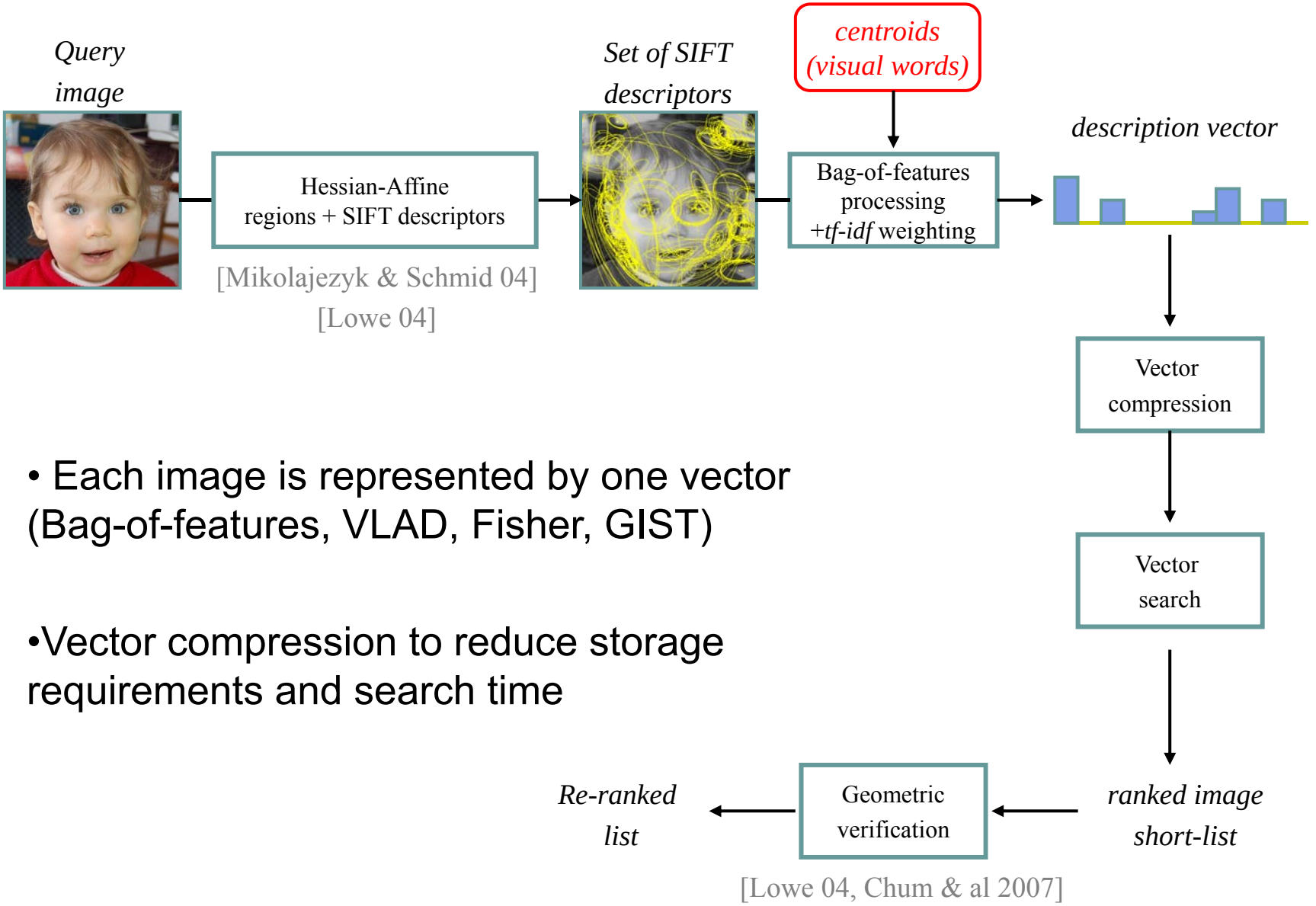
Google image search



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

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

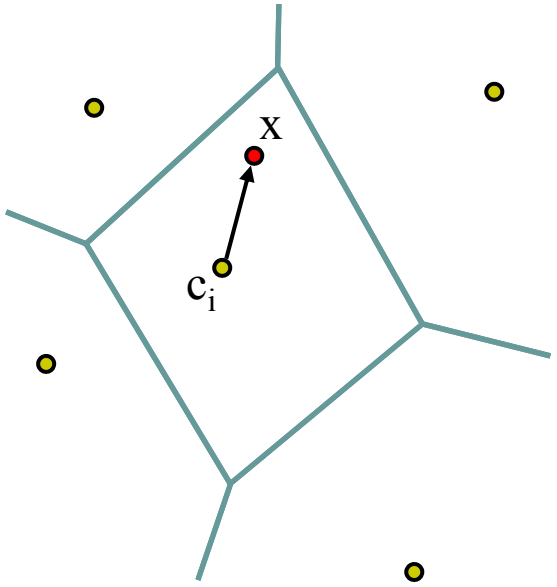
- Vector of locally aggregated descriptors (VLAD) [Jegou et al. 10]
  - non sparse vector
  - fast to compute
  - excellent results with a small vector dimensionality

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

VLAD : vector of locally aggregated descriptors

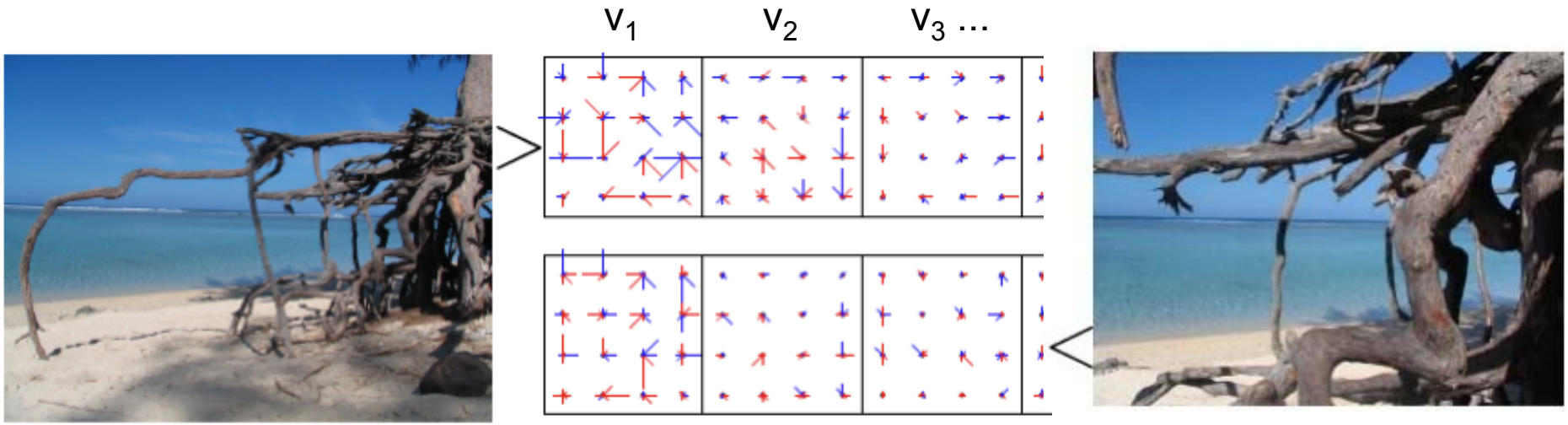
- Determine a vector quantifier (*k*-means)
  - output: *k* centroids (visual words): $c_1,\ldots,c_i,\ldots c_k$
  - centroid c_i has dimension *d*
- For a given image
  - assign each descriptor to closest center c_i
  - accumulate (sum) descriptors per cell

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

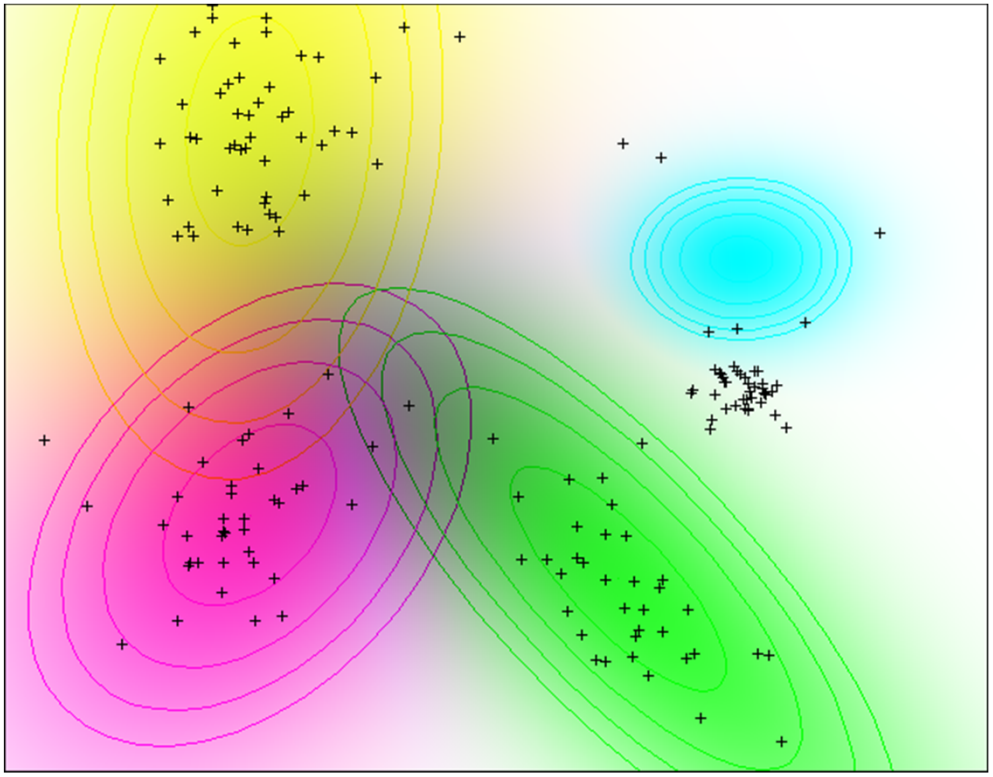


SIFT-like representation per centroid (+ components: blue, - components: red)

- 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 variance (diagonal)

Translated cluster → large derivative on μ_i for this component

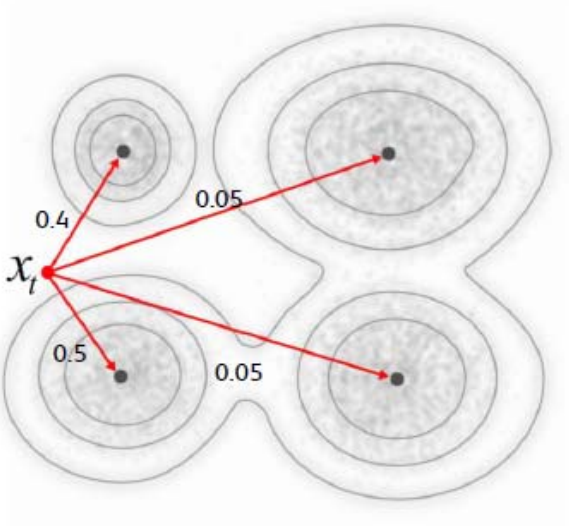
[Perronnin & Dance 07]

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

GIST descriptor

Models different frequency channels,
global descriptor of the image content

Torralba et al. (2003)

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					

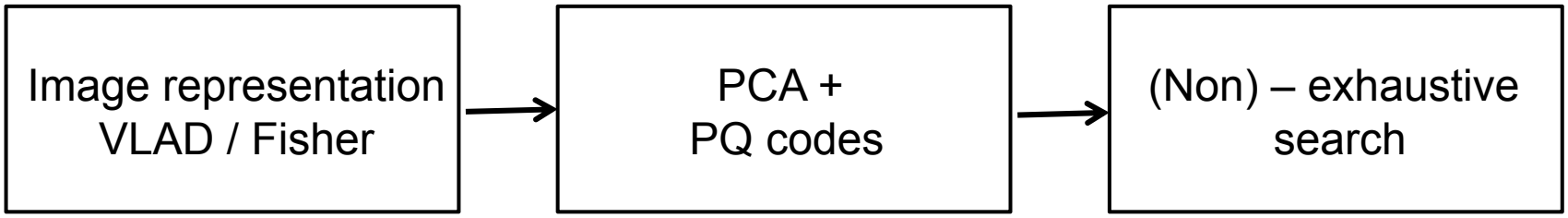
- Observations:
  - Fisher, VLAD better than BoF for a given descriptor size
  - Choose a small D if output dimension D' is small

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

Compact image representation

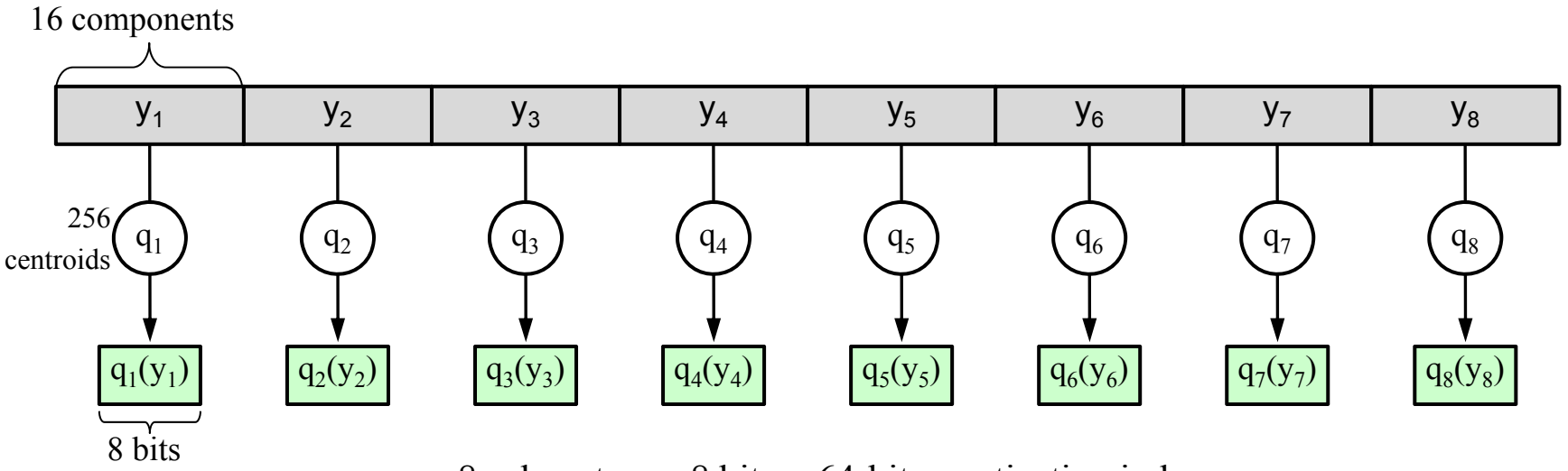
- Aim: improving the tradeoff between
  - search speed
  - memory usage
  - search quality

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



Product quantization for nearest neighbor search

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

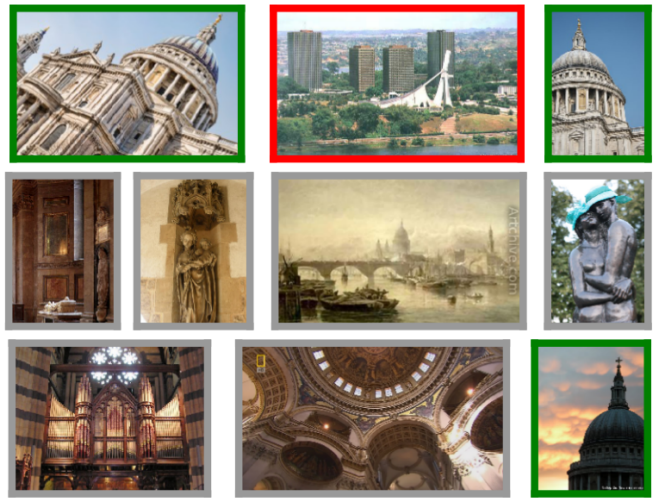
[Jegou, Douze, Schmid, PAMI'11]

Deep image retrieval [Gordo et al. 2016]

- Learn to represent images for retrieval
- Requires train data
- Network which focuses on retrieval

Obtaining Training Data: NN require a lot of data

- Public dataset of landmark images
  - ~200K images
  - ~600 different landmarks
  - **Extremely noisy! Needs automated cleaning procedure**



[Babenko et al, Neural codes @ ECCV14]

xerox

Obtaining Training Data: NN require a lot of data

- Public dataset of landmark images
  - ~200K images
  - ~600 different landmarks
  - **Extremely noisy! Needs automated cleaning procedure**

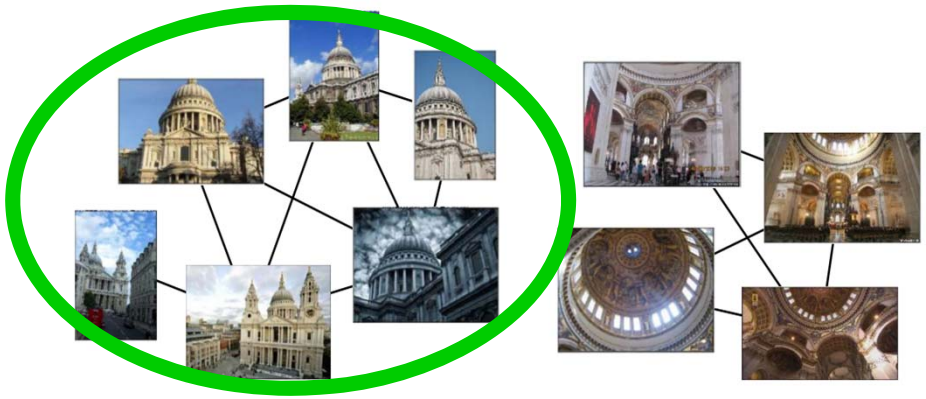
1. Keypoint matching of all image pairs of each landmark
   - Hessian-Affine keypoints + SIFT descriptors
   - First-to-second neighbor ratio rule
   - Geometrical verification with Affine transformation

xerox

Obtaining Training Data: NN require a lot of data

- Public dataset of landmark images
  - ~200K images
  - ~600 different landmarks
  - **Extremely noisy! Needs automated cleaning procedure**

2. Verification of groups
   - Construct a graph
   - Prune low-weight connections
   - Extract the largest connected component

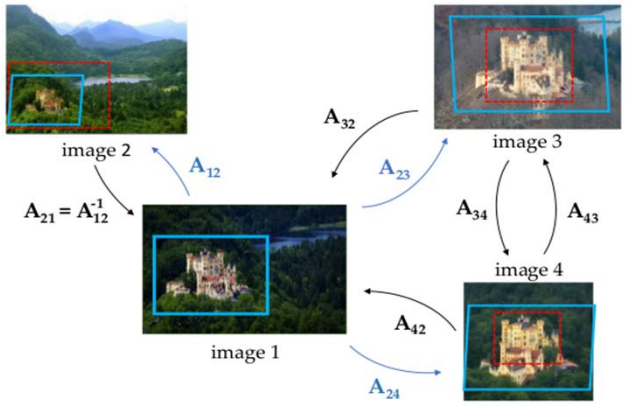


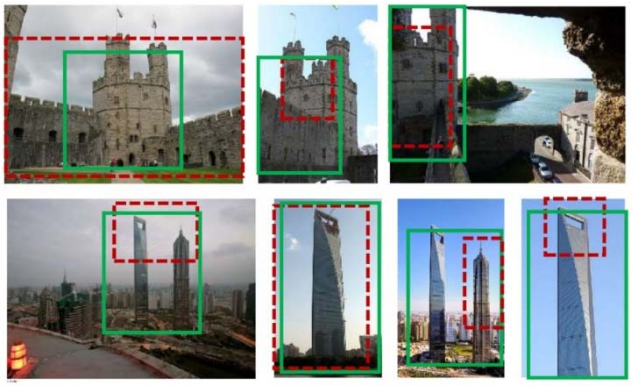
xerox

Obtaining Training Data : NN require a lot of data

- Public dataset of landmark images
  - ~200K images
  - ~600 different landmarks
  - **Extremely noisy! Needs automated cleaning procedure**

3. Bounding box prediction
   - Diffusion process that iteratively adjust landmark location using neighbors





xerox

Obtaining Training Data : NN require a lot of data

- Public dataset of landmark images
  - ~200K images
  - ~600 different landmarks
  - **Extremely noisy! Needs automated cleaning procedure**

At the end:

- Clean dataset
  - 40K spatially verified images
  - Approximate bounding box annotations
  - **No overlap with Oxford5K / Paris6K / Holidays datasets**

Note: list of verified images and corresponding Bounding Boxes available

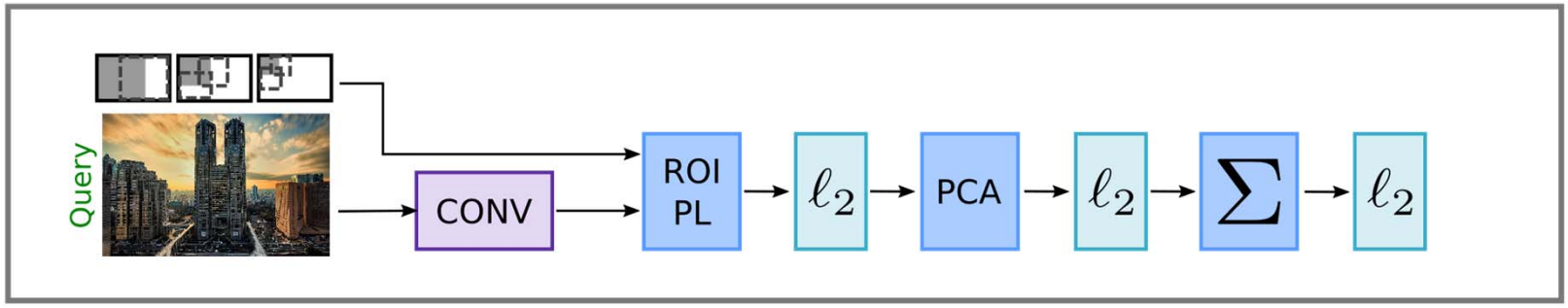
xerox

Baseline approach

- We base our work on the recent R-MAC descriptor

[Particular object retrieval with integral max-pooling of CNN activations. Giorgos Tolias, Ronan Sicre, and Hervé Jégou @ ICLR16]

Advantages: no aspect ratio distortion, high resolution images.



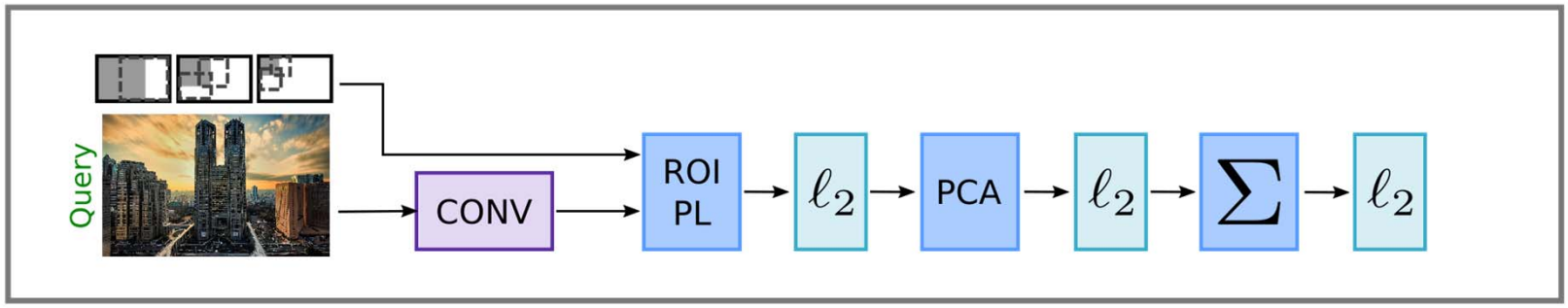
xerox

Baseline approach

- We base our work on the recent R-MAC descriptor

  [Particular object retrieval with integral max-pooling of CNN activations. Giorgos Tolias, Ronan Sicre, and Hervé Jégou @ ICLR16]

  Advantages: no aspect ratio distortion, high resolution images.

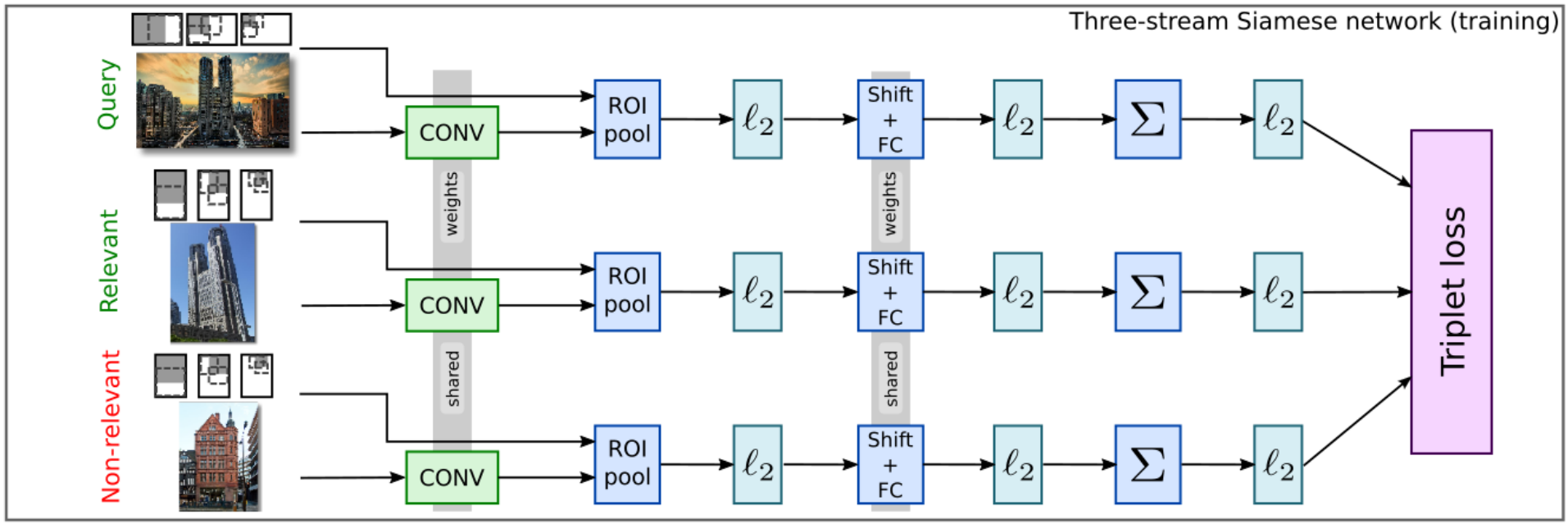


- Observation
  - Every step is differentiable ⇒ the descriptor can be trained end-to-end

xerox

Proposed approach

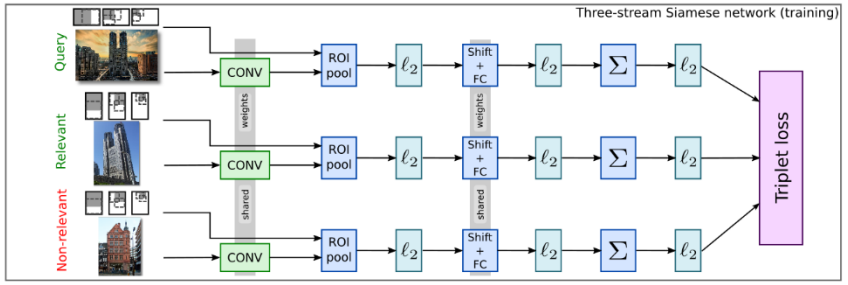
- Learning to rank
  - We propose to use a three-stream Siamese Network



xerox

Proposed approach

- Learning to rank
  - We propose to use a three-stream Siamese Network
  - We use a triplet loss suited for the ranking task
    - Explicitly enforces that, given a query, a relevant element is closer to the query than an non-relevant one:



$$L(I_q, I^+, I^-) = \frac{1}{2}\max(0, m + \|q - d^+\|^2 - \|q - d^-\|^2),$$

Qualitative results – Network activations

Where do conv5 neurons fire?

[#33] Baseline VGG16:

[#33] After training:

xerox

Qualitative results – Network activations

Where do conv5 neurons fire?

[#19] Baseline VGG16:

[#19] After training:

xerox

Experiments: Standard Benchmarks

- Compared to global descriptors: clearly outperforms previous methods

	Method	Dim.	Datasets				
			Oxf5k	Par6k	Oxf105k	Par106k	Holidays
Global descriptors	[Jégou and Zisserman (2014)]	1024	56.0	-	50.2	-	72.0
	[Jégou and Zisserman (2014)]	128	43.3	-	35.3	-	61.7
	[Gordo et al (2012)]	512	-	-	-	-	79.0
	[Babenko et al (2014)]	128	55.7*	-	52.3*	-	75.9/78.9$^{\triangleright}$
	[Gong et al (2014)]	2048	-	-	-	-	80.8
	[Babenko and Lempitsky (2015)]	256	53.1	-	50.1	-	80.2$^{\triangleright}$
	[Ng et al (2015)]	128	59.3*	59.0*	-	-	83.6
	[Paulin et al (2015)]	256K	56.5	-	-	-	79.3
	[Perronnin and Larlus (2015)]	4000	-	-	-	-	84.7
	[Tolias et al (2016)]	512	66.9	83.0	61.6	75.7	85.2†/86.9$^{\dagger,\triangleright}$
	[Kalantidis et al (2015)]	512	68.2	79.7	63.3	71.0	84.9
	[Arandjelovic et al (2016)]	4096	71.6	79.7	-	-	83.1/87.5$^{\triangleright}$
	[Radenovic et al (2016)]	512	79.7	83.8	73.9	76.4	82.5$^{\triangleright}$
	Previous state of the art		79.7 Radenovic et al (2016)	83.8 Radenovic et al (2016)	73.9 Radenovic et al (2016)	76.4 Radenovic et al (2016)	84.9 Kalantidis et al (2015)
	Ours	2048	**86.1**	**94.5**	**82.8**	**90.6**	**94.8**$^{\triangleright}$