Instance-level recognition

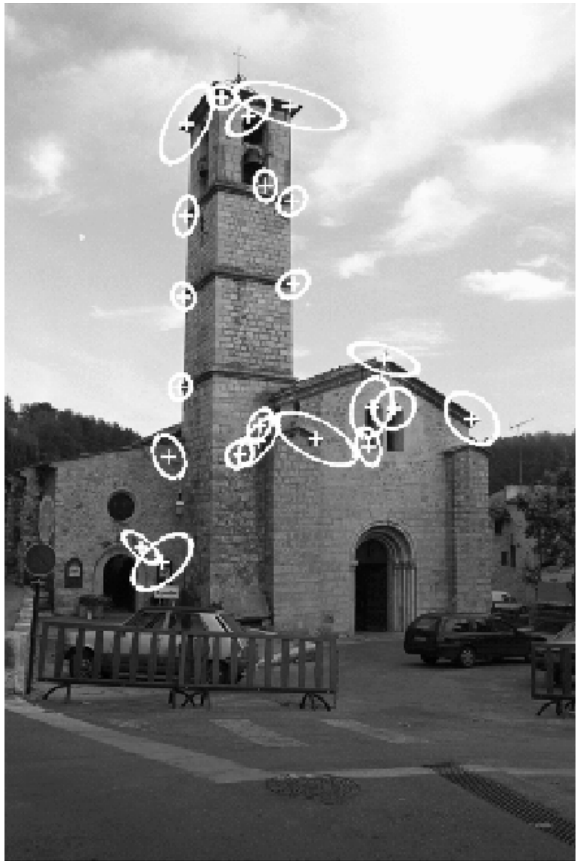
1) Local invariant features

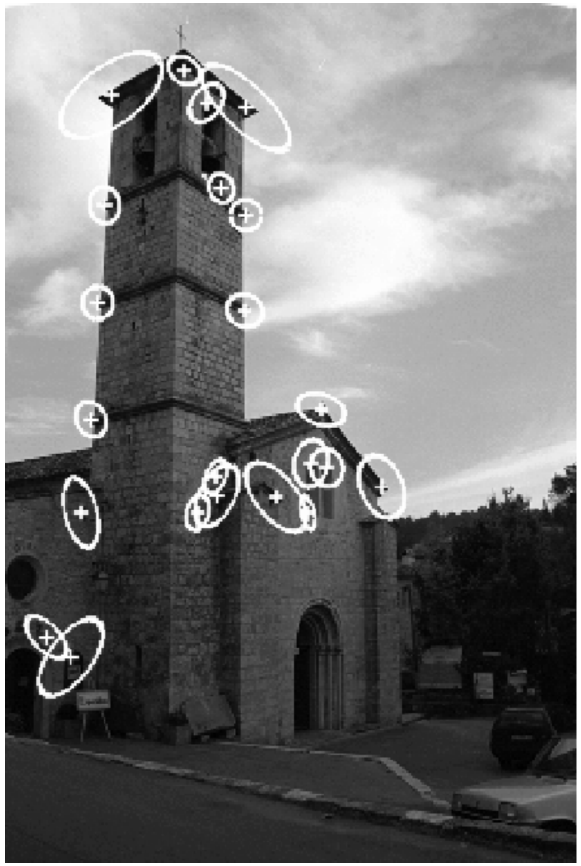
2) Matching and recognition with local features

3) Efficient visual search

4) Very large scale indexing

Matching of descriptors





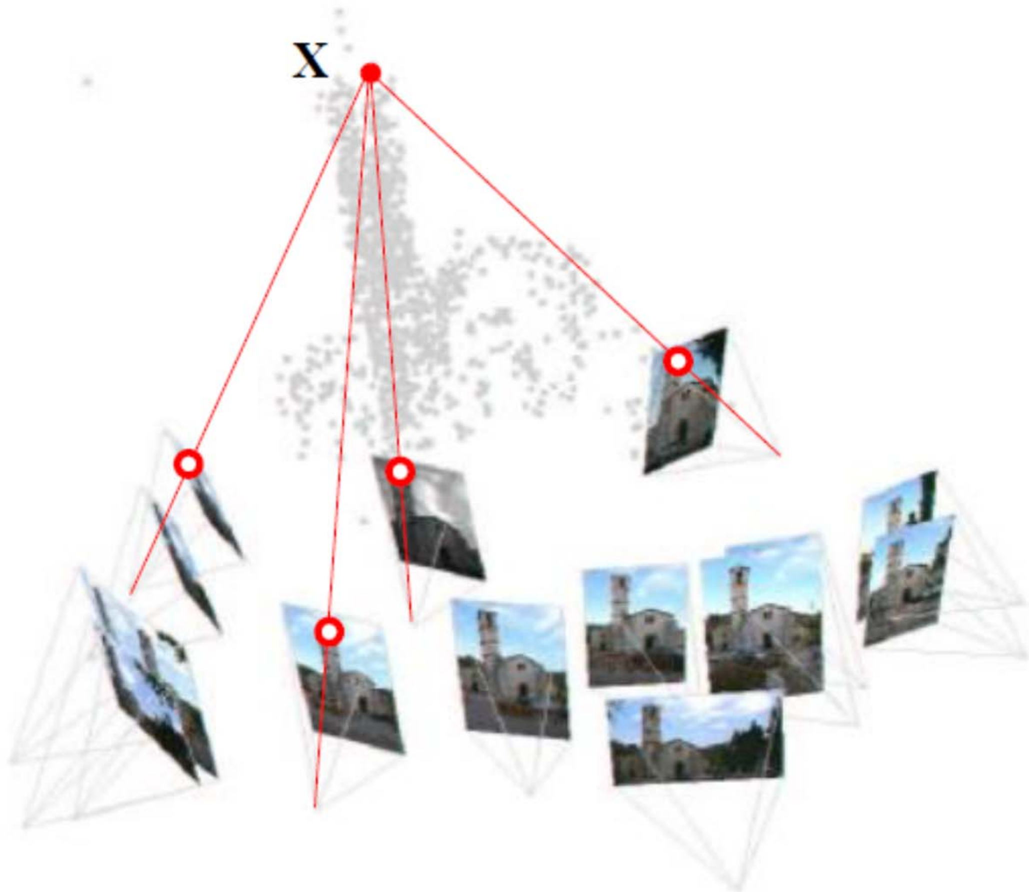
Matching and 3D reconstruction

- Establish correspondence between two (or more) images

[Schaffalitzky and Zisserman ECCV 2002]

Matching and 3D reconstruction

- Establish correspondence between two (or more) images



[Schaffalitzky and Zisserman ECCV 2002]

Building Rome in a Day

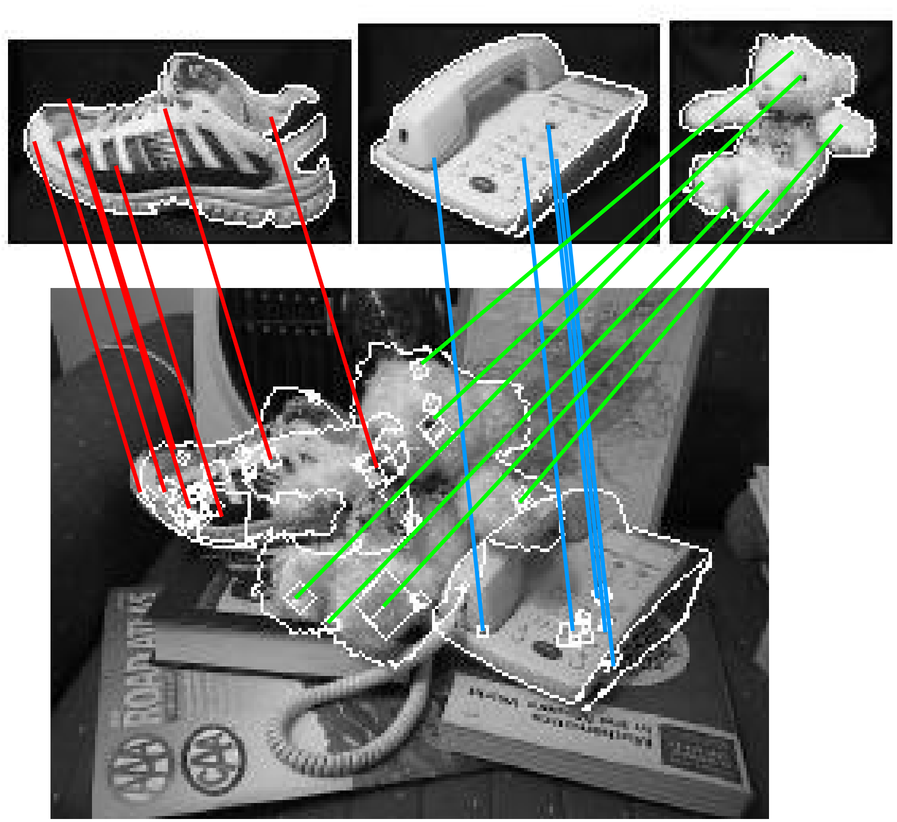
57,845 downloaded images, 11,868 registered images

[Agarwal, Snavely, Simon, Seitz, Szeliski, ICCV'09]

Object recognition

- Establish correspondence between the target image and (multiple) images in the model database

Model database

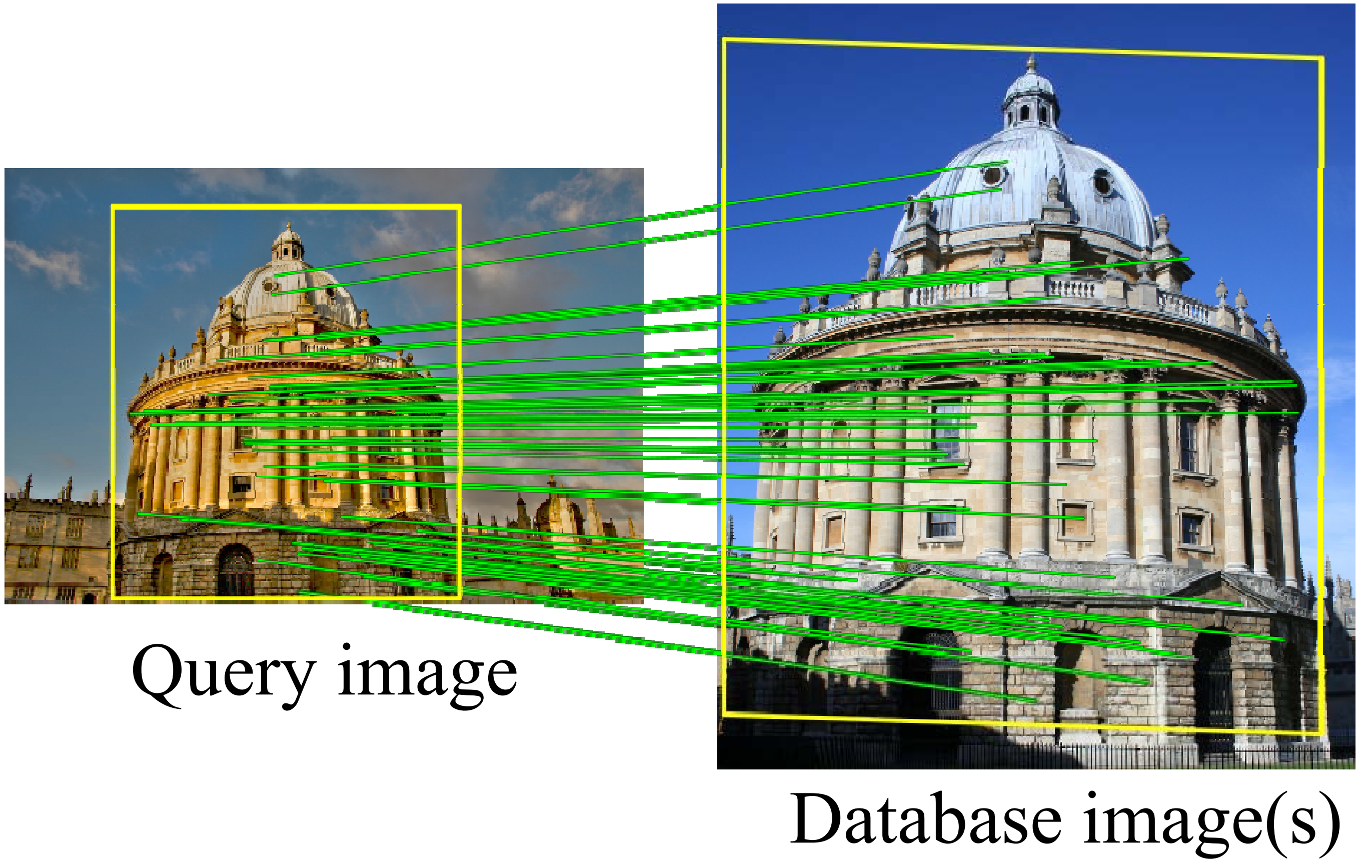


Target image

[D. Lowe, 1999]

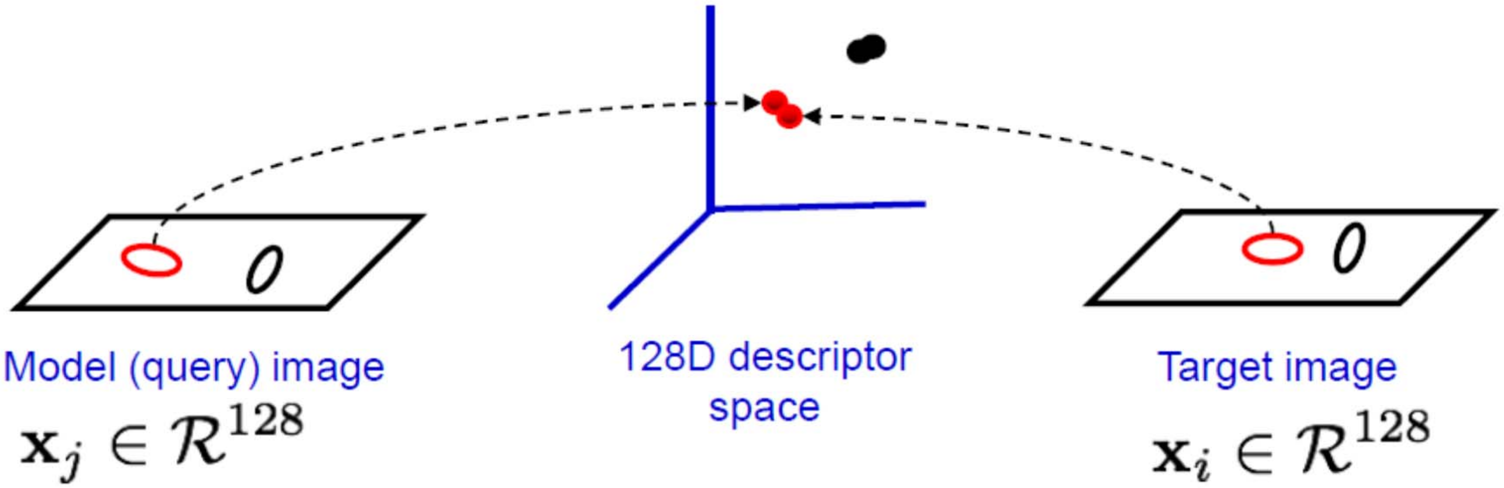
Visual search

- Establish correspondence between the query image and all images from the database depicting the same object or scene



Matching of descriptors

- Find the nearest neighbor in the second image for each descriptor, for example SIFT



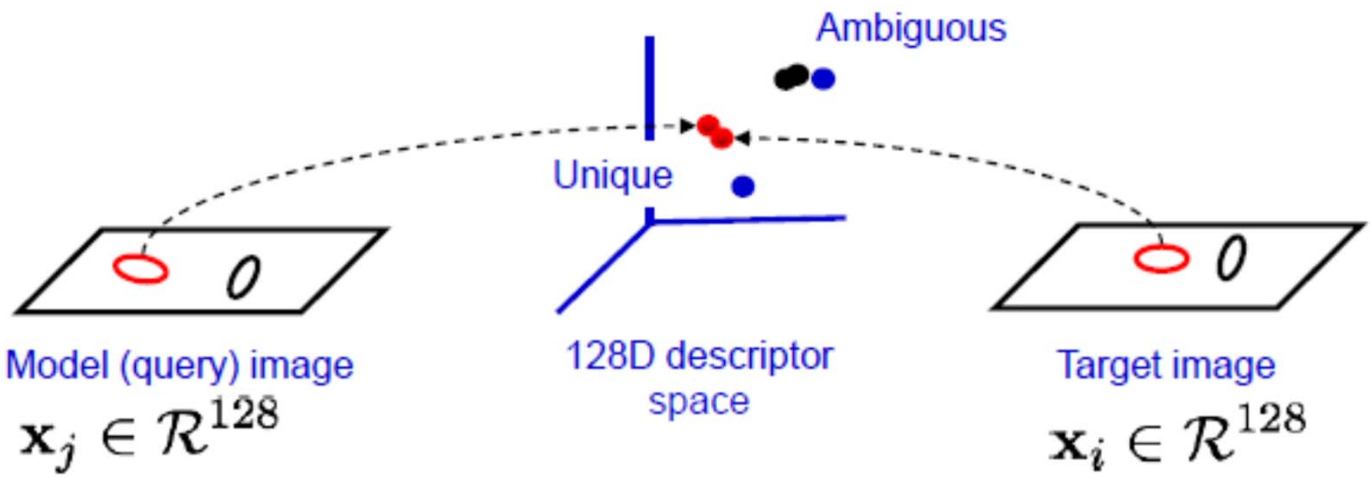
Need to solve some variant of the "nearest neighbor problem" for all feature vectors, $\mathbf{x}_j \in \mathcal{R}^{128}$, in the query image:

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

where, $\mathbf{x}_i \in \mathcal{R}^{128}$, are features in the target image.

Matching of descriptors

- Pruning strategies
  - Ratio with respect to the second best match (d1/d2 << 1) [Lowe, '04]



If the 2nd nearest neighbour is much further than the 1st nearest neighbour, the match is more "unique" or discriminative.

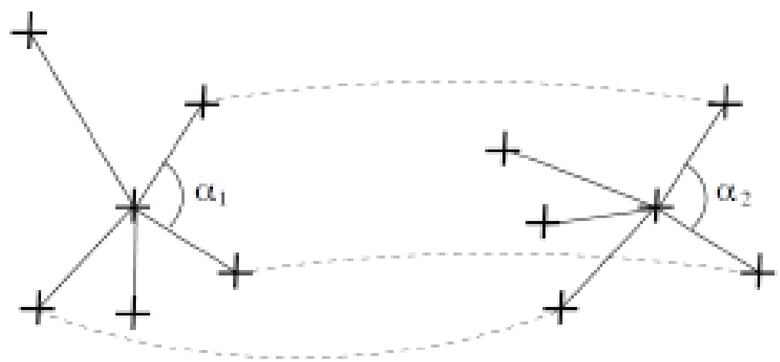
Measure this by the ratio: $r = d_{1NN} / d_{2NN}$

r is between 0 and 1

r is small the match is more unique.

Matching of descriptors

- 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

- 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)

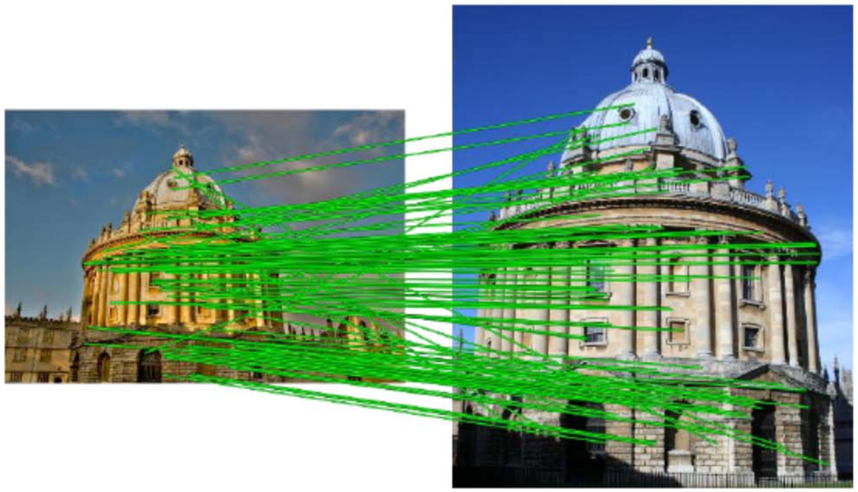
Matching of descriptors

- 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)

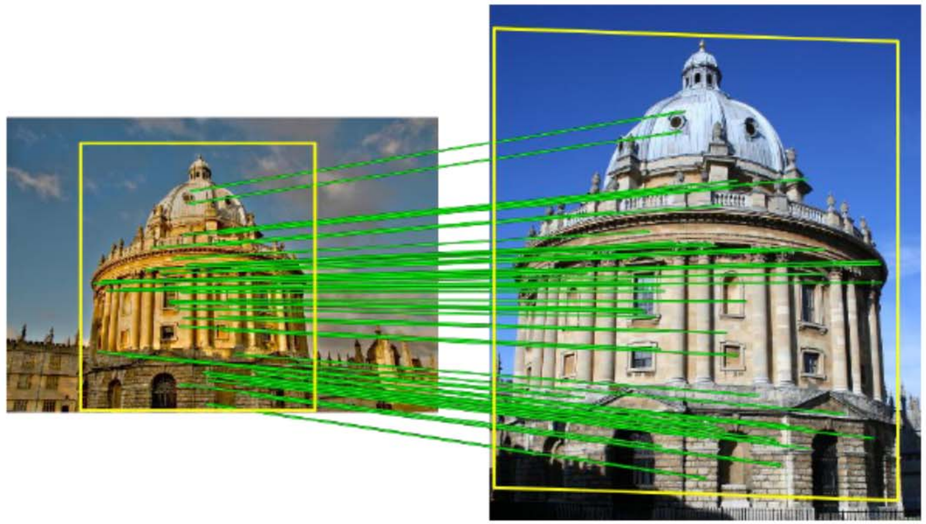
- Geometric verification with global constraint
  - All matches must be consistent with a global geometric transformation
  - However, there are many incorrect matches
  - Need to estimate simultaneously the geometric transformation and the set of consistent matches

Geometric verification with global constraint

- Example of a geometric verification

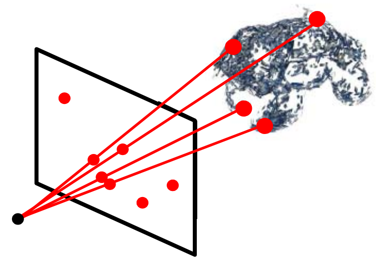


Tentative matches



Matches consistent with an affine transformation

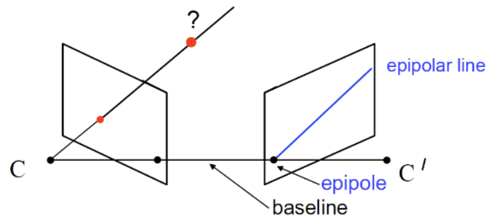
Examples of global constraints

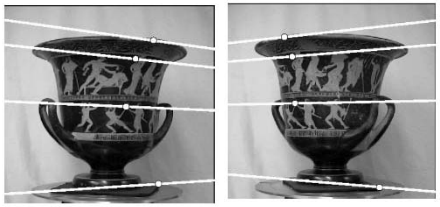


1 view and known 3D model.

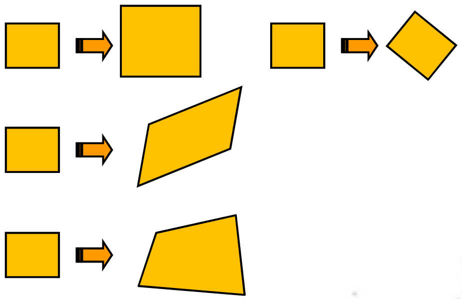
- Consistency with a (known) 3D model.

2 views





- Epipolar constraint
- **2D transformations**
  - Similarity transformation
  - Affine transformation
  - Projective transformation



N-views

Are images consistent with a 3D model?

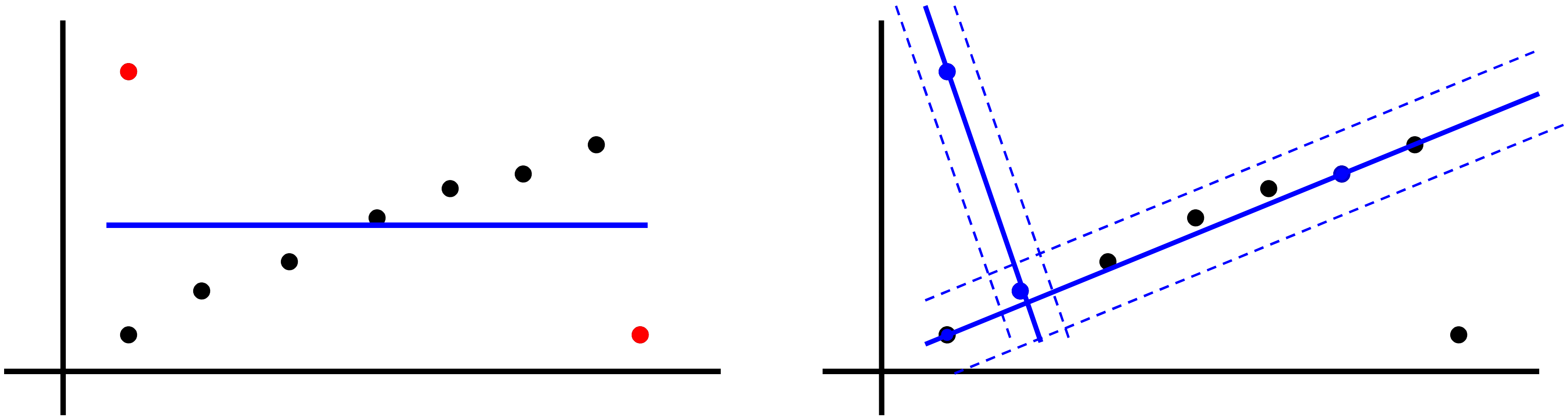
Matching of descriptors

- Geometric verification with global constraint
  - All matches must be consistent with a global geometric transformation
  - However, there are many incorrect matches
  - Need to estimate simultaneously the geometric transformation and the set of consistent matches

- Robust estimation of global constraints
  - RANSAC (RANdom Sampling Consensus) [Fishler&Bolles'81]
  - Hough transform [Lowe'04]

RANSAC: Example of robust line estimation

Fit a line to 2D data containing outliers



There are two problems

1. a line fit which minimizes perpendicular distance
2. a classification into inliers (valid points) and outliers

Solution: use robust statistical estimation algorithm RANSAC

(RANdom Sample Consensus) [Fishler & Bolles, 1981]

Slide credit: A. Zisserman

RANSAC robust line estimation

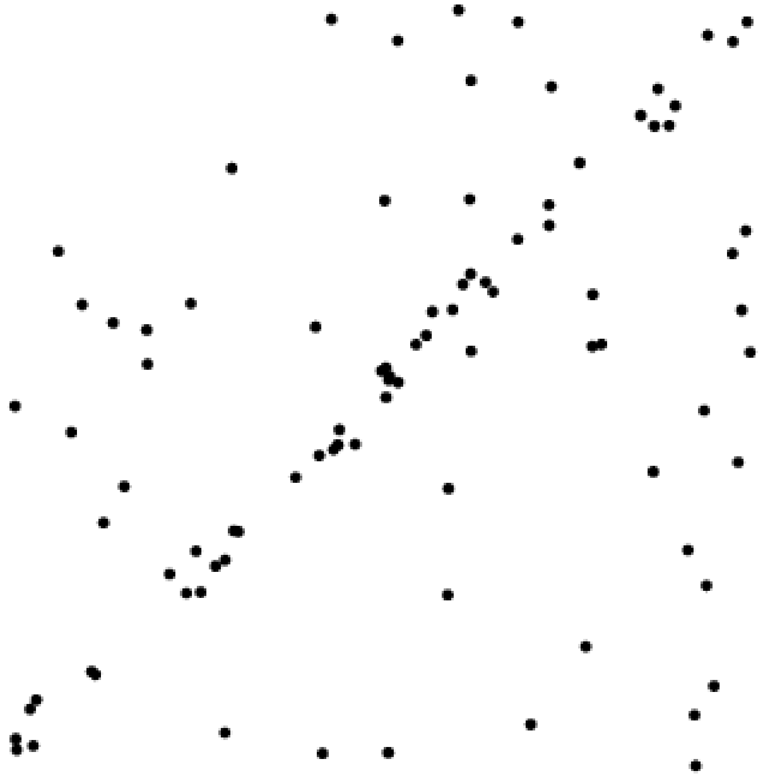
Repeat

1. Select random sample of 2 points
2. Compute the line through these points
3. Measure support (number of points within threshold distance of the line)

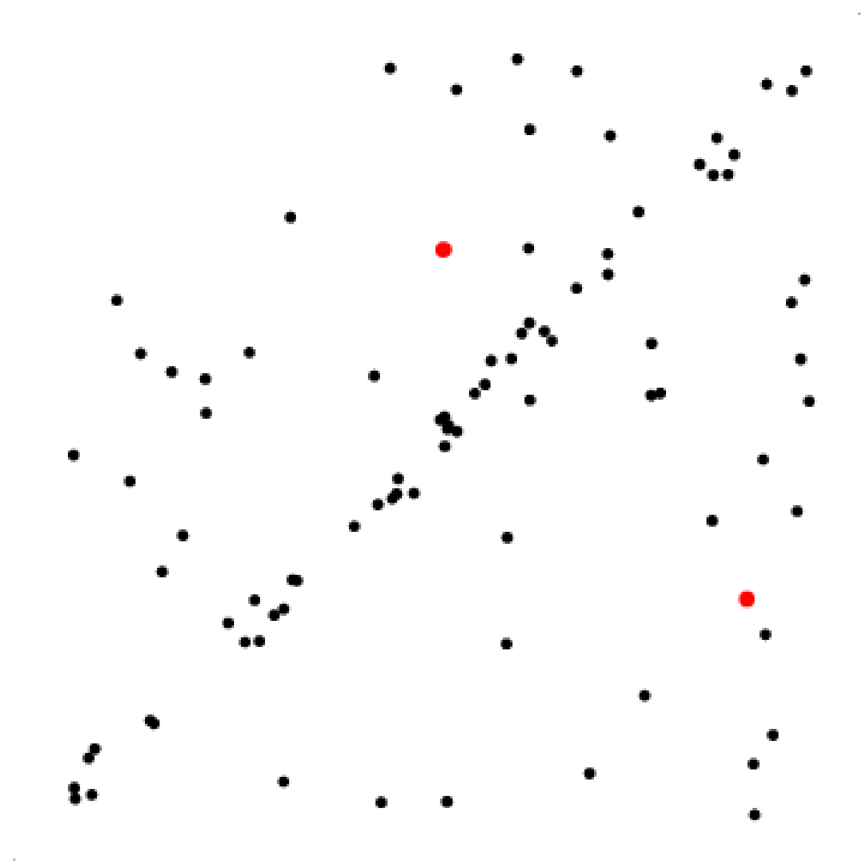
Choose the line with the largest number of inliers

- Compute least squares fit of line to inliers (regression)

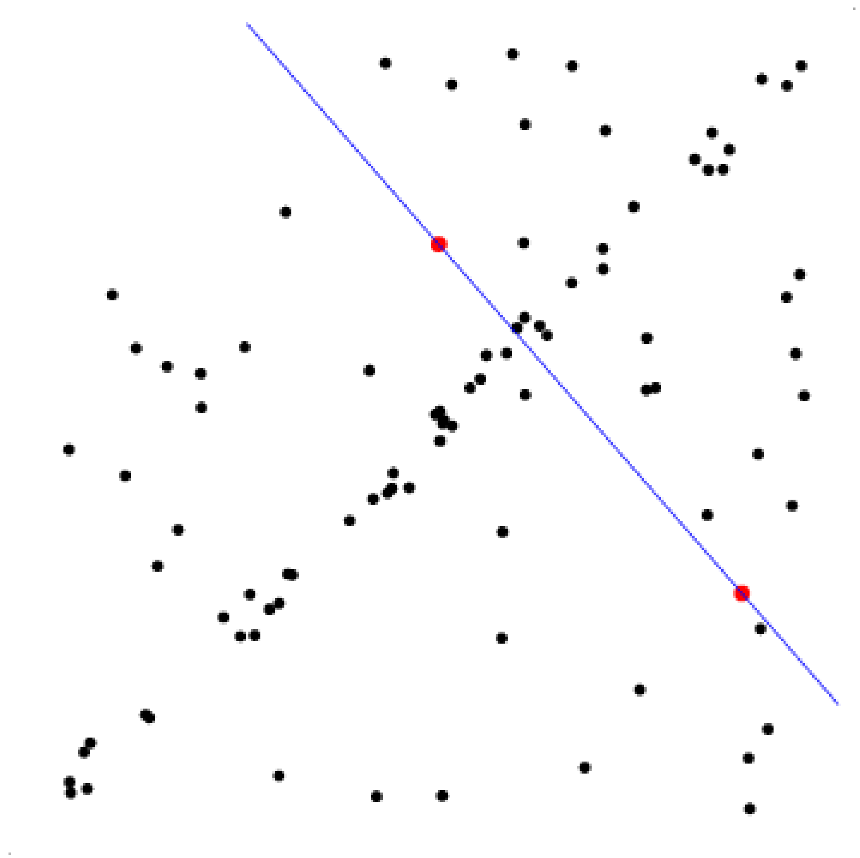
Slide credit: A. Zisserman



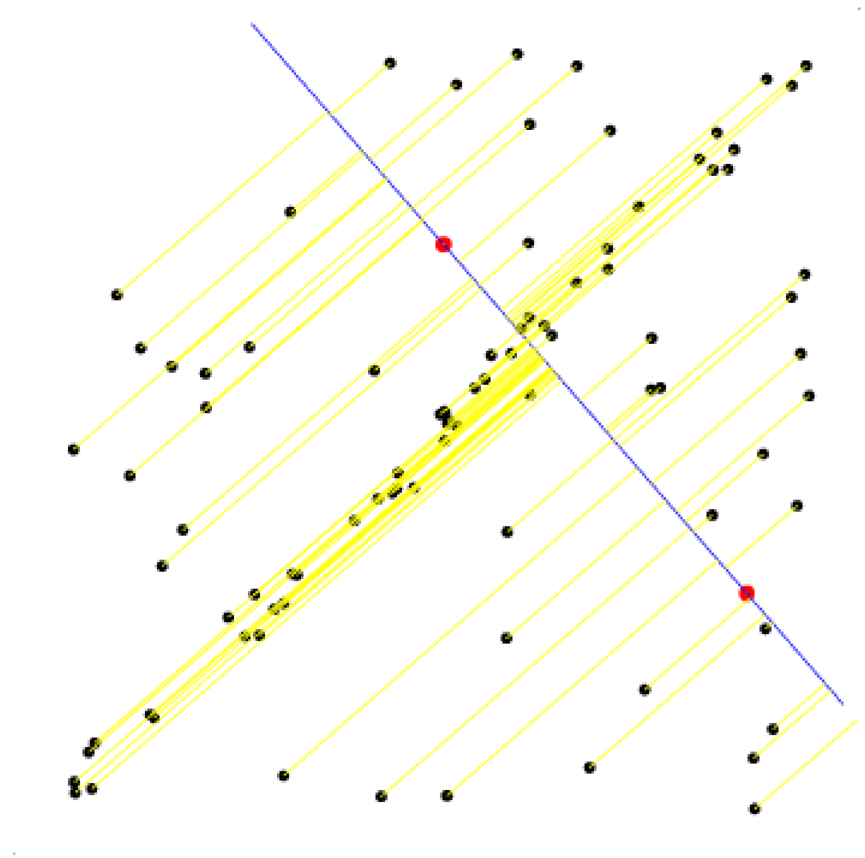
Slide credit: O. Chum



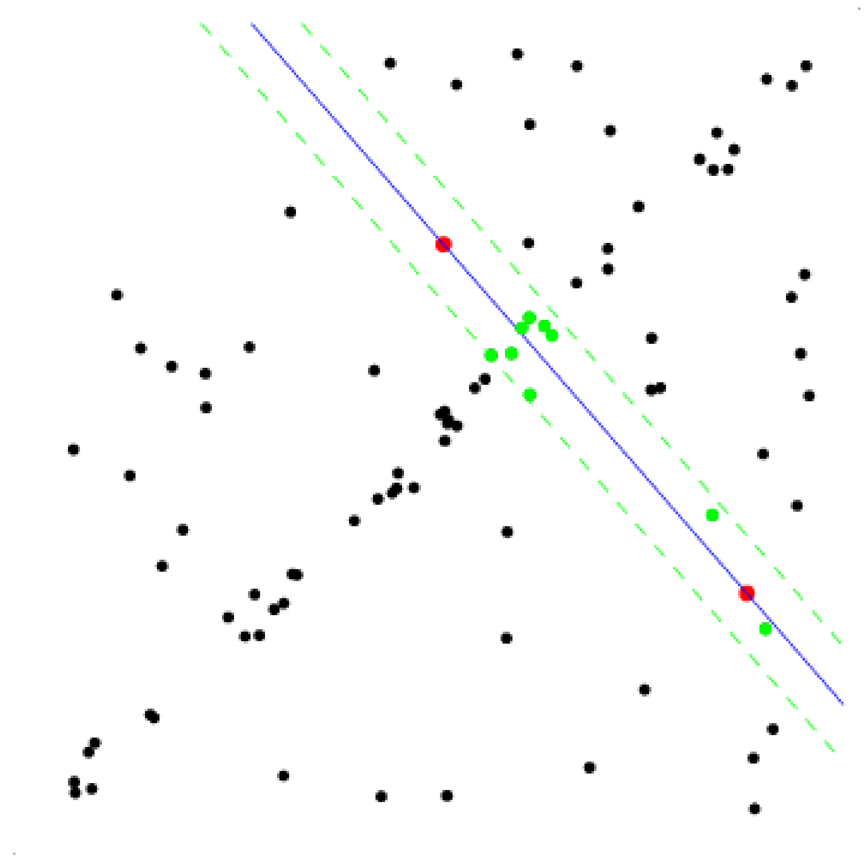
Slide credit: O. Chum



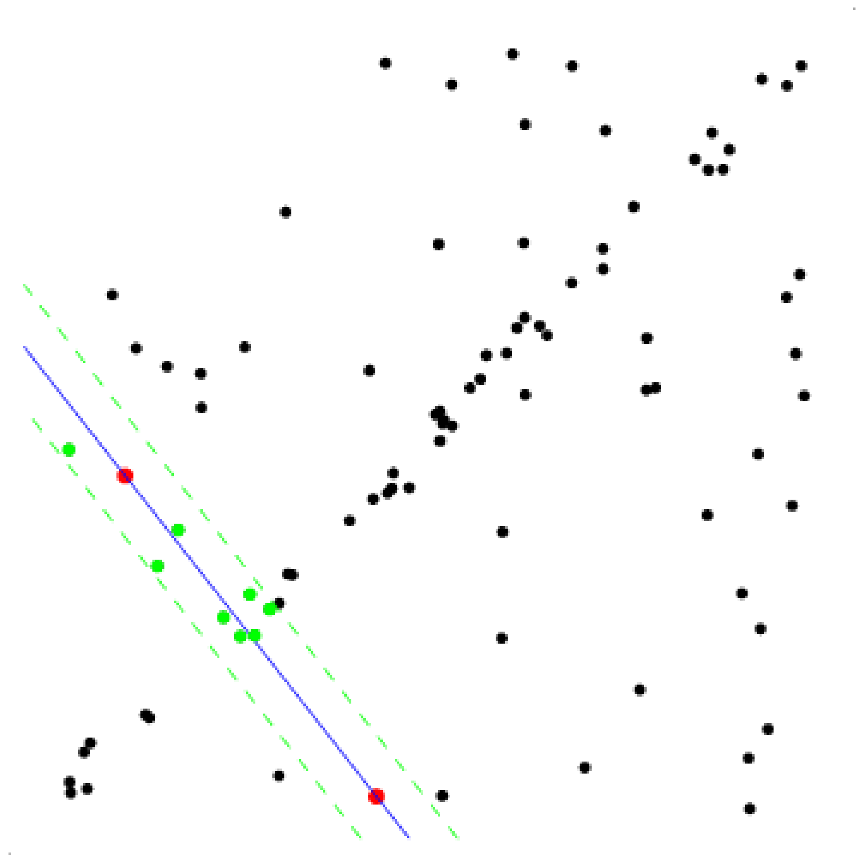
Slide credit: O. Chum



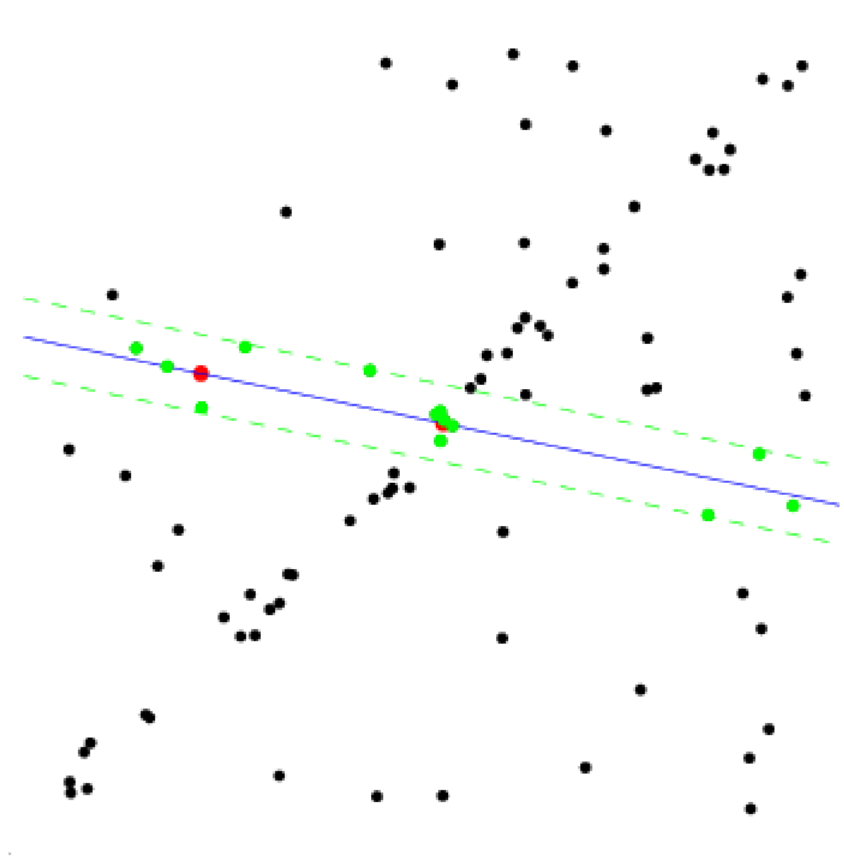
Slide credit: O. Chum



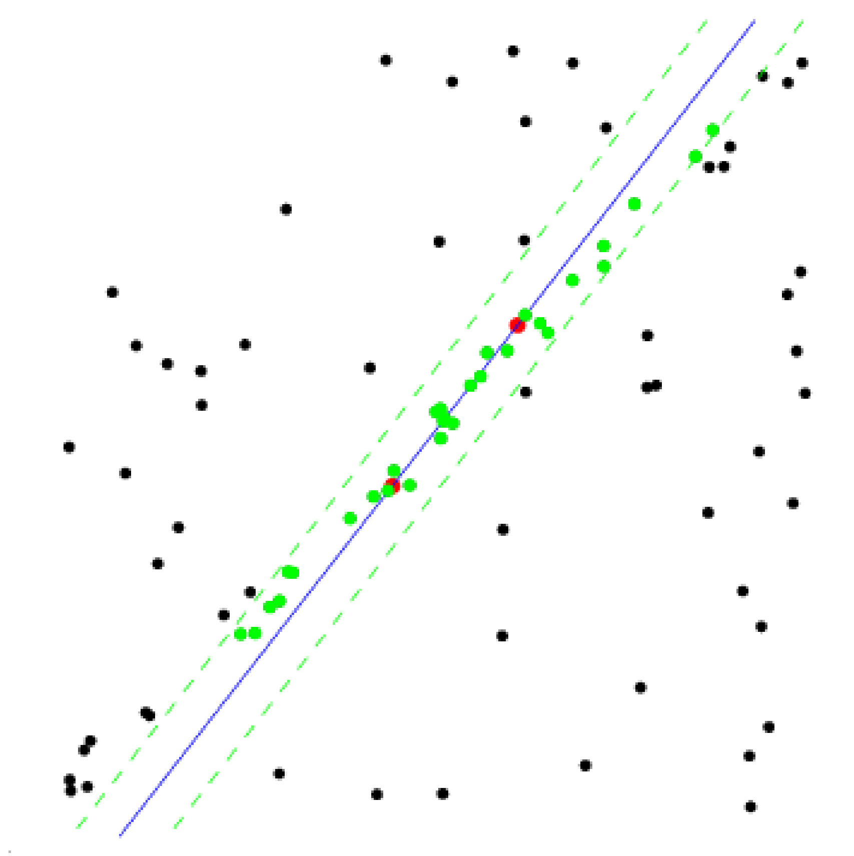
Slide credit: O. Chum



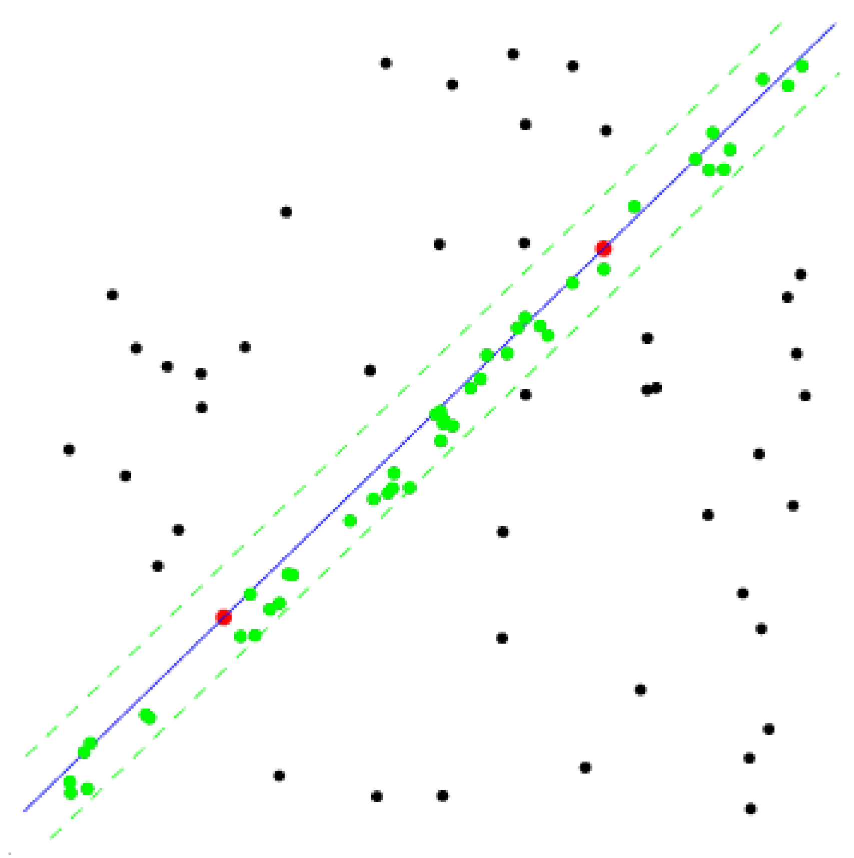
Slide credit: O. Chum



Slide credit: O. Chum



Slide credit: O. Chum

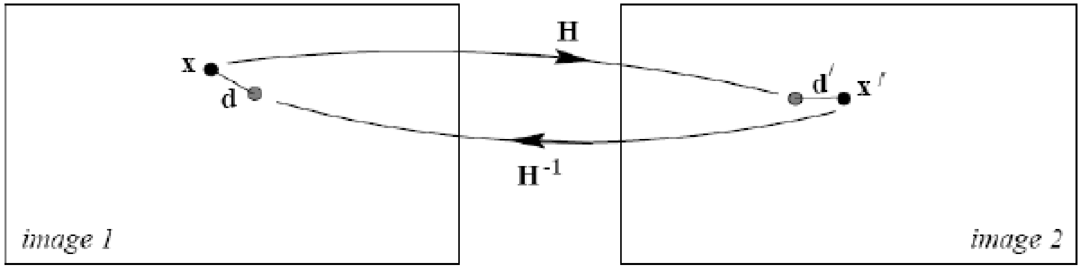


Slide credit: O. Chum

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

Matching of descriptors

- Geometric verification with global constraint
  - All matches must be consistent with a global geometric transformation
  - However, there are many incorrect matches
  - Need to estimate simultaneously the geometric transformation and the set of consistent matches
- Robust estimation of global constraint
  - RANSAC (RANdom Sampling Consensus) [Fishler&Bolles'81]
  - **Hough transform [Lowe'04]**

Strategy 2: Hough transform

- General outline:
  - Discretize parameter space into bins
  - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  - Find bins that have the most votes

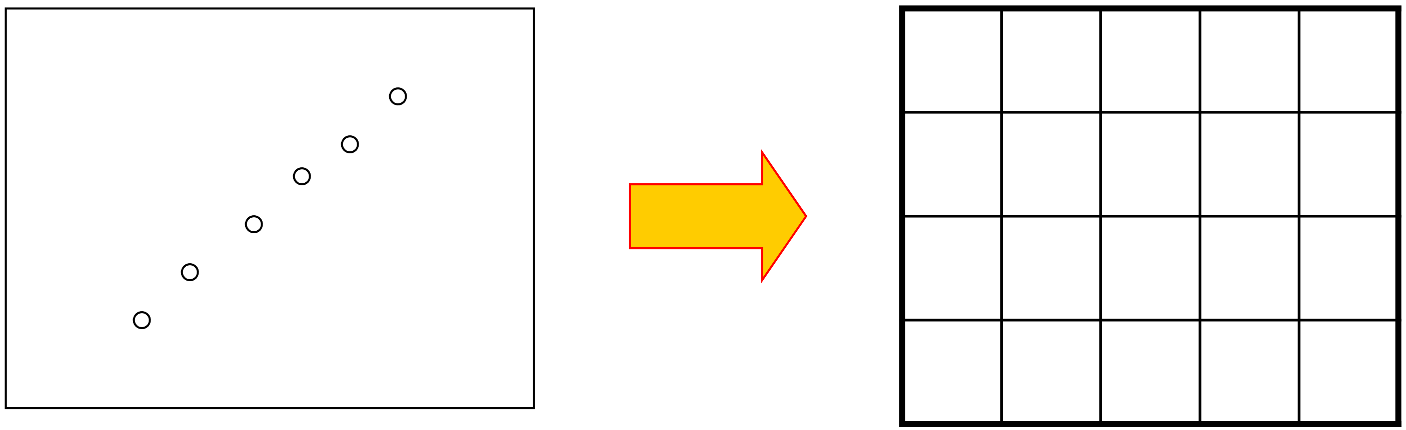


Image space　　　Hough parameter space

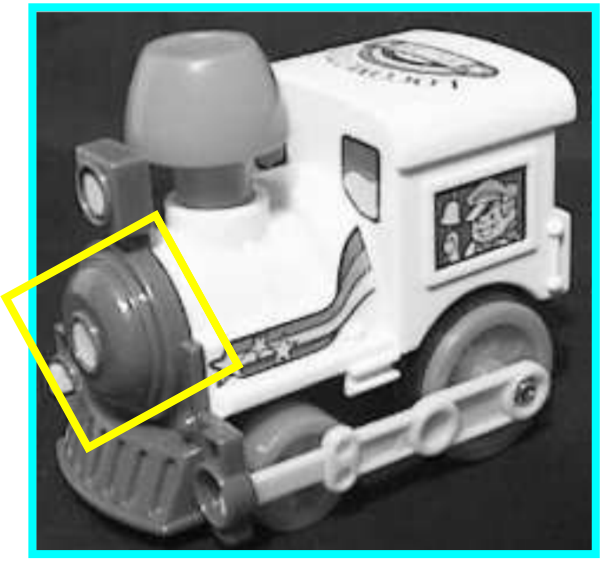
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Hough transform for object recognition

Suppose our features are scale- and rotation-covariant

- Then a single feature match provides an alignment hypothesis (translation, scale, orientation)

model



Target image

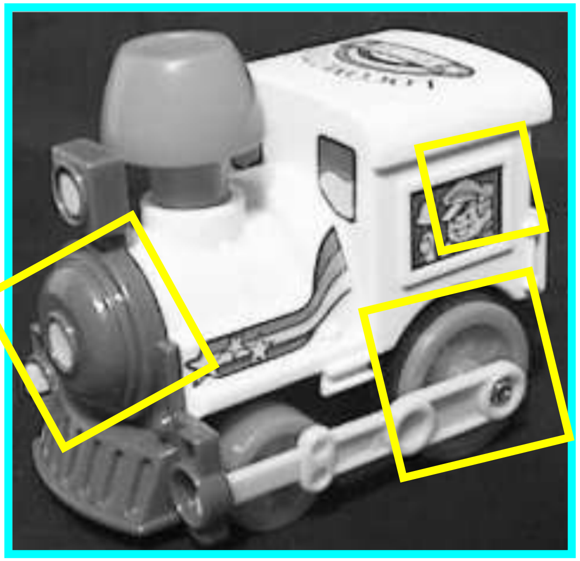
David G. Lowe. **"Distinctive image features from scale-invariant keypoints",** *IJCV* 60 (2), pp. 91-110, 2004.

Hough transform for object recognition

Suppose our features are scale- and rotation-covariant

- Then a single feature match provides an alignment hypothesis (translation, scale, orientation)
- Of course, a hypothesis obtained from a single match is unreliable
- Solution: Coarsely quantize the transformation space. Let each match vote for its hypothesis in the quantized space.

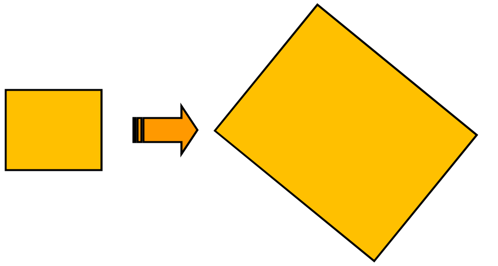
model



David G. Lowe. **"Distinctive image features from scale-invariant keypoints",** *IJCV* 60 (2), pp. 91-110, 2004.

Similarity transformation is specified by four parameters: scale factor s, rotation θ, and translations t_x and t_y.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = sR(\theta) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



Recall, each SIFT detection has: position (x_i, y_i), scale s_i, and orientation θ_i.

How many correspondences are needed to compute similarity transformation?

Compute similarity transformation from a single correspondence:

$$(x_A, y_A, s_A, \theta_A) \leftrightarrow (x'_A, y'_A, s'_A, \theta'_A)$$

$$\theta = \theta'_A - \theta_A$$

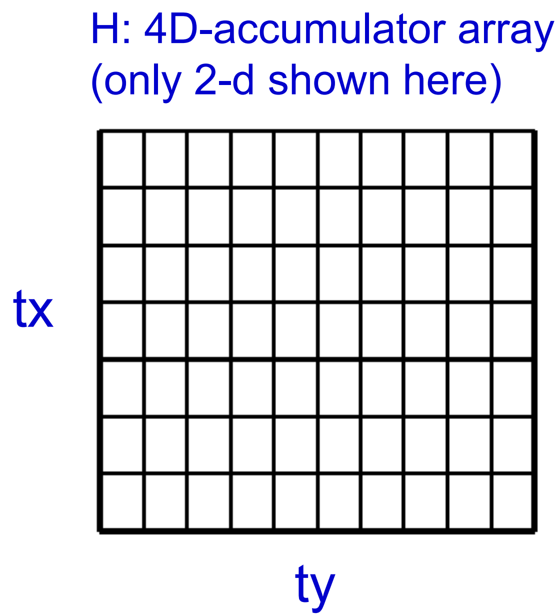
$$t_x = x'_A - x_A$$

$$t_y = y'_A - y_A$$

$$s = s'_A / s_A$$

Basic algorithm outline

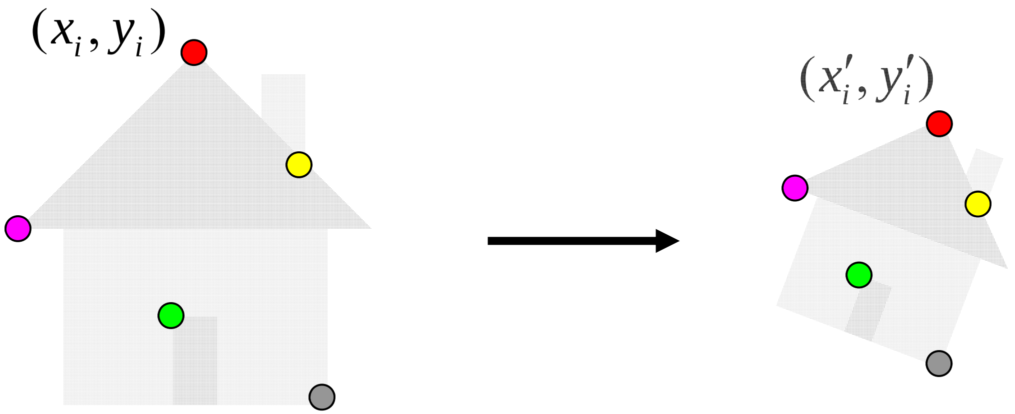
1. Initialize accumulator H to all zeros
2. For each tentative match
   compute transformation hypothesis: tx, ty, s, θ
   H(tx,ty,s,θ) = H(tx,ty,s,θ) + 1
   end
   end
3. Find all bins (tx,ty,s,θ) where H(tx,ty,s,θ) has at least three votes



- Correct matches will consistently vote for the same transformation while mismatches will spread votes.
- Cost: Linear scan through the matches (step 2), followed by a linear scan through the accumulator (step 3).

Fitting an affine transformation

Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{bmatrix}$$

Linear system with six unknowns

Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters

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]

Summary

Finding correspondences in images is useful for

- Image matching, panorama stitching
- Object recognition
- Large scale image search: next part of the lecture

Beyond local point matching

- Semi-local relations
- Global geometric relations:
  - Epipolar constraint
  - 3D constraint (when 3D model is available)
  - 2D tnfs: Similarity / Affine / Homography
- Algorithms:
  - RANSAC
  - Hough transform

$$\boxed{\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0}$$

$$\boxed{\mathbf{x} = \mathrm{P}\mathrm{X}}$$

$$\boxed{\mathbf{x}' = \mathrm{H}\mathbf{x}}$$

Instance-level recognition

1) Local invariant features

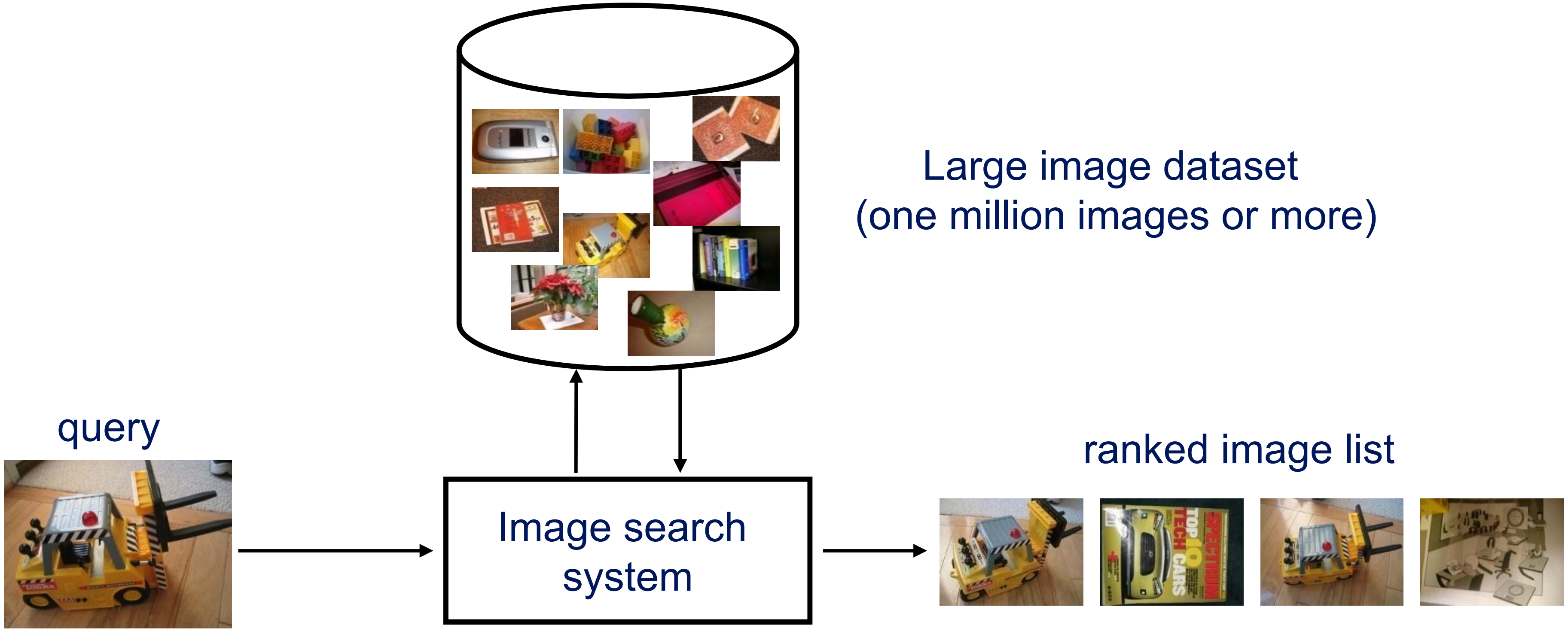
2) Matching and recognition with local features

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

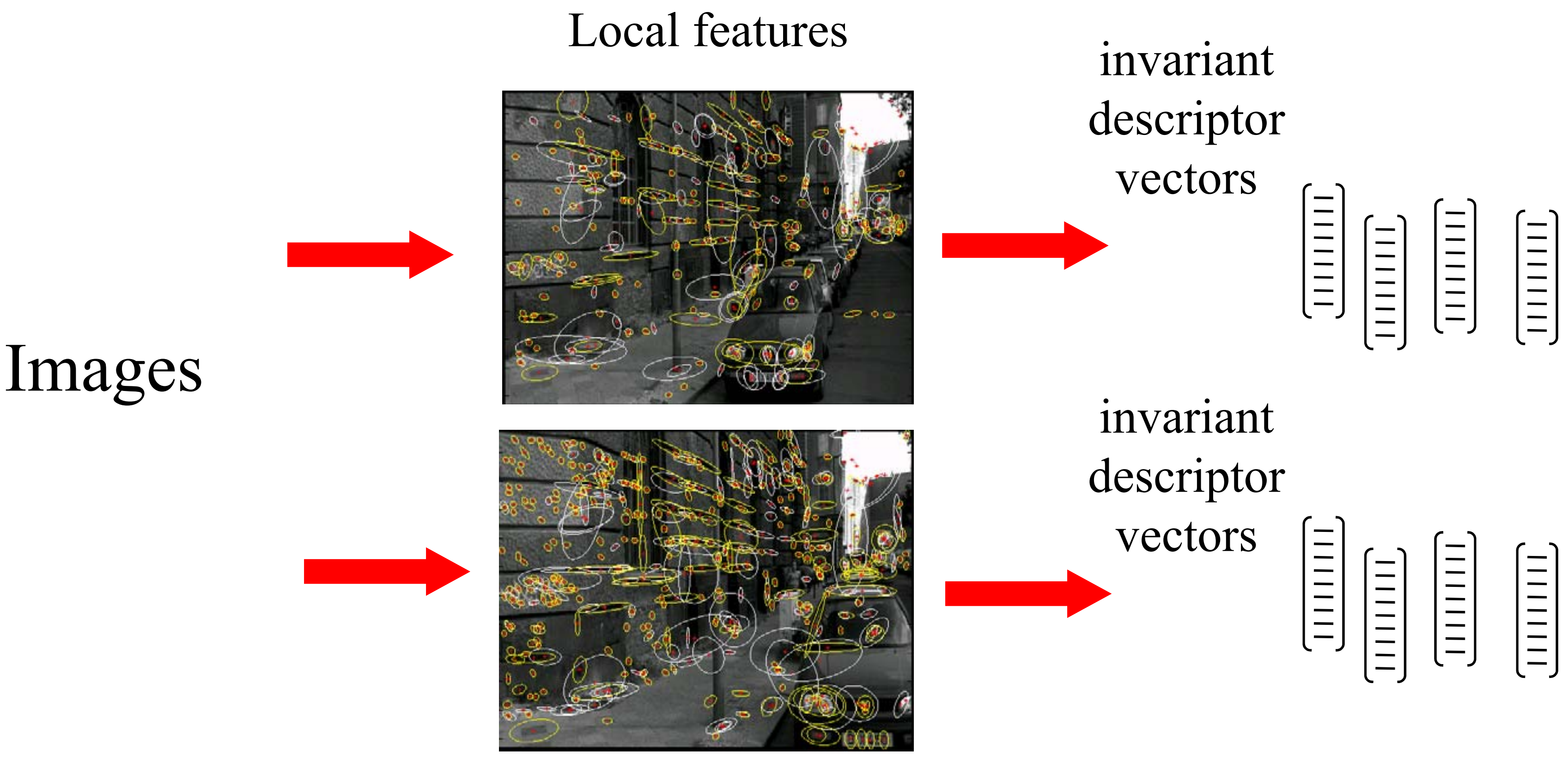
Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors

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

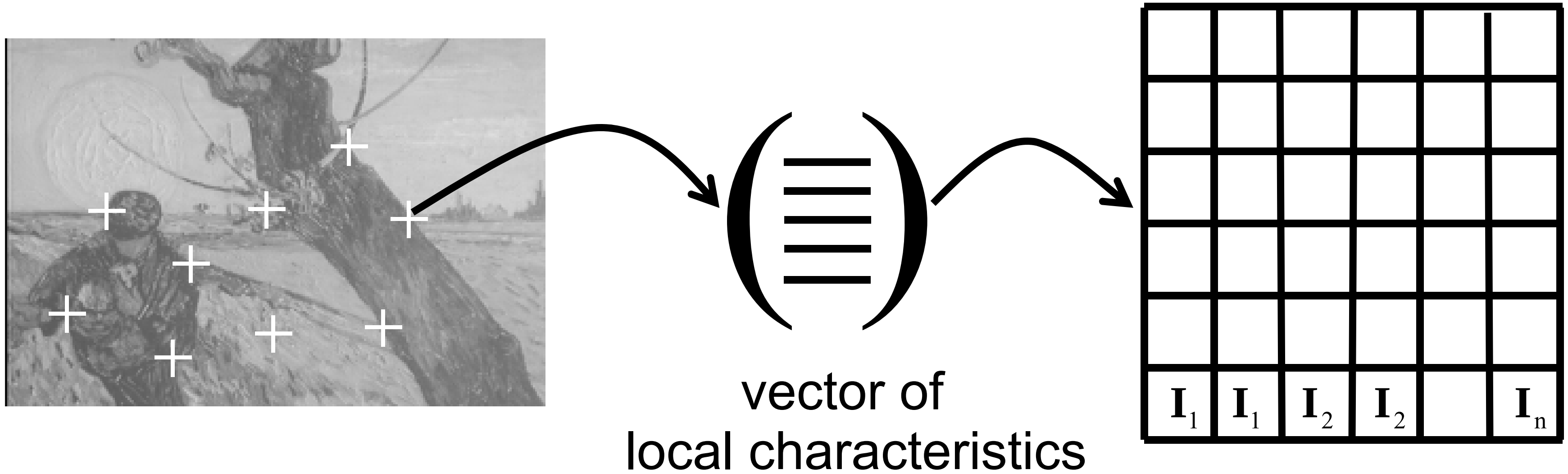
   (Bag-of-words representation)

Strategy 1: Efficient approximate NN search

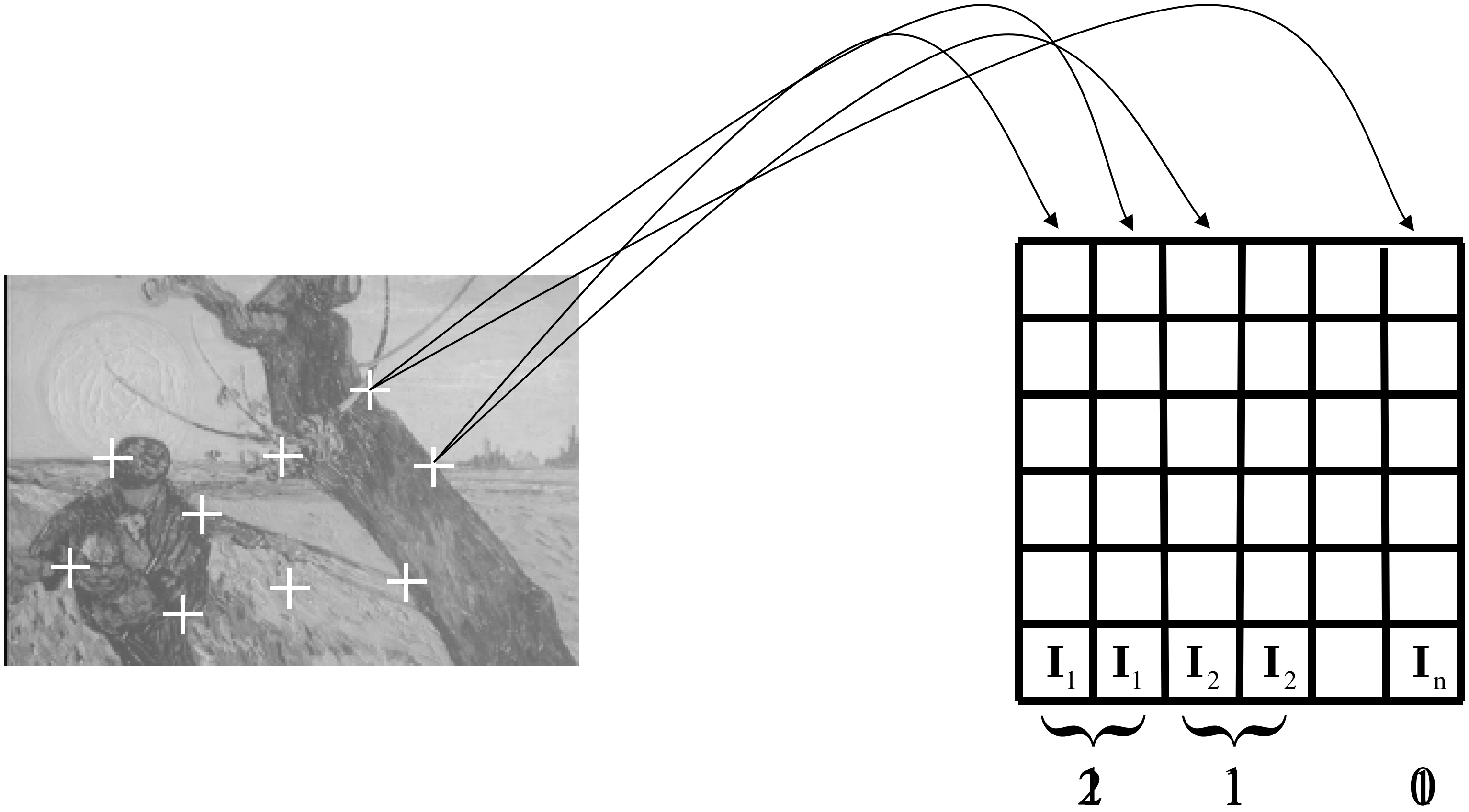


1. Compute local features in each image independently
2. Describe each feature by a descriptor vector
3. Find nearest neighbour vectors between query and database
4. Rank matched images by number of (tentatively) corresponding regions
5. Verify top ranked images based on spatial consistency

Voting algorithm



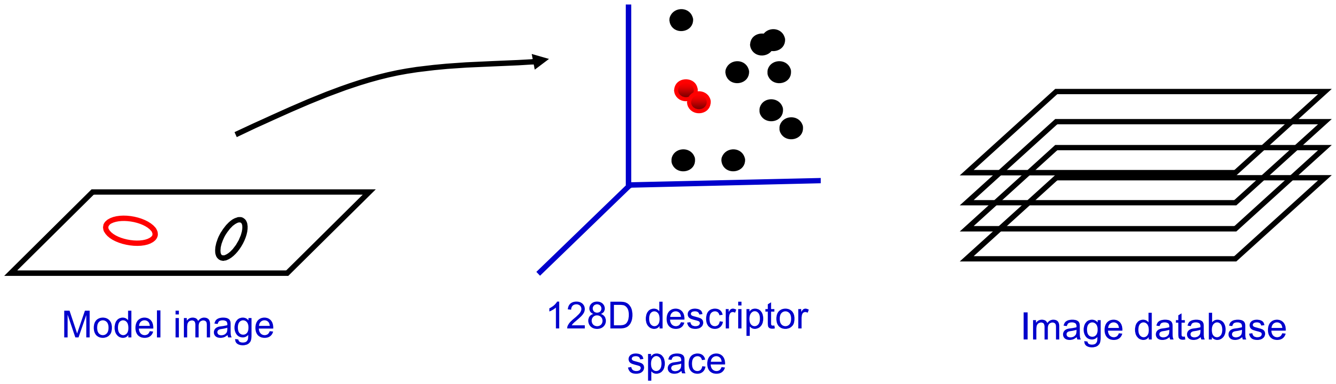
Voting algorithm



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:

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

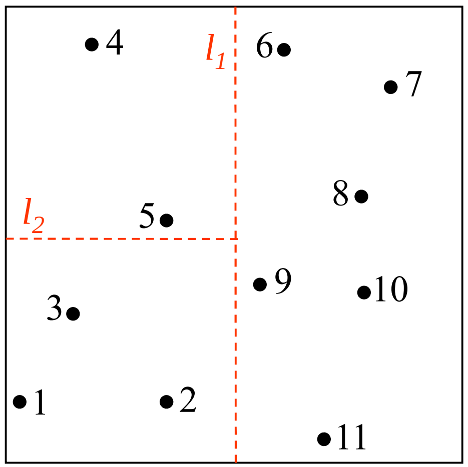
where $\mathbf{x_i}$ are features in database images.

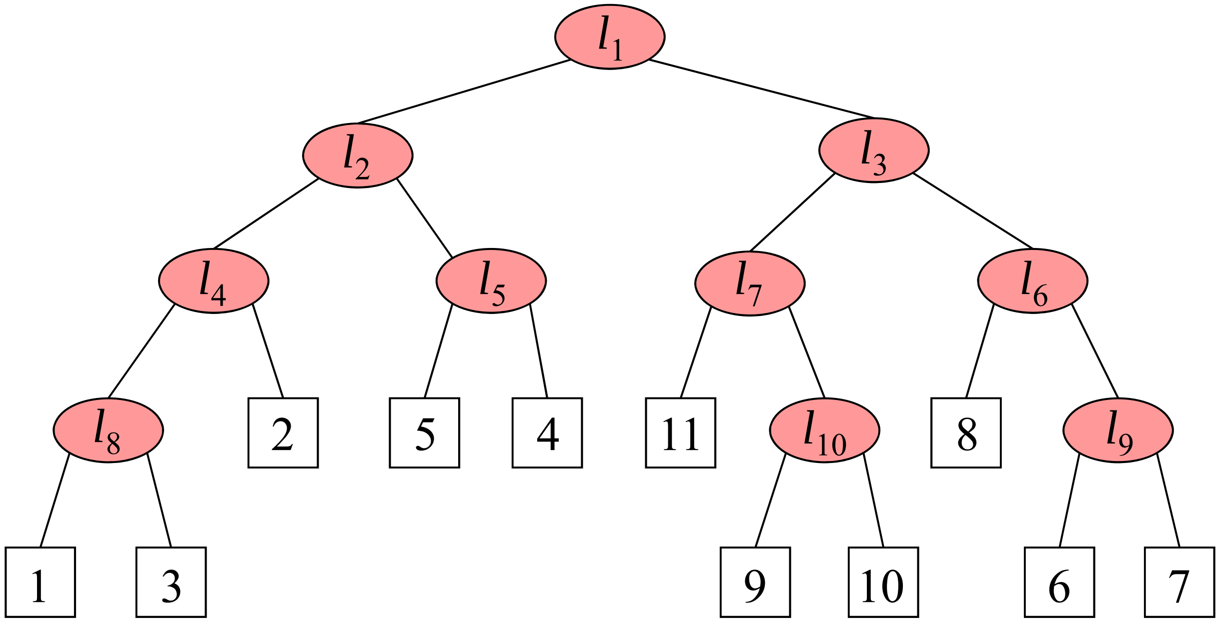
Nearest-neighbour matching is the major computational bottleneck

- 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

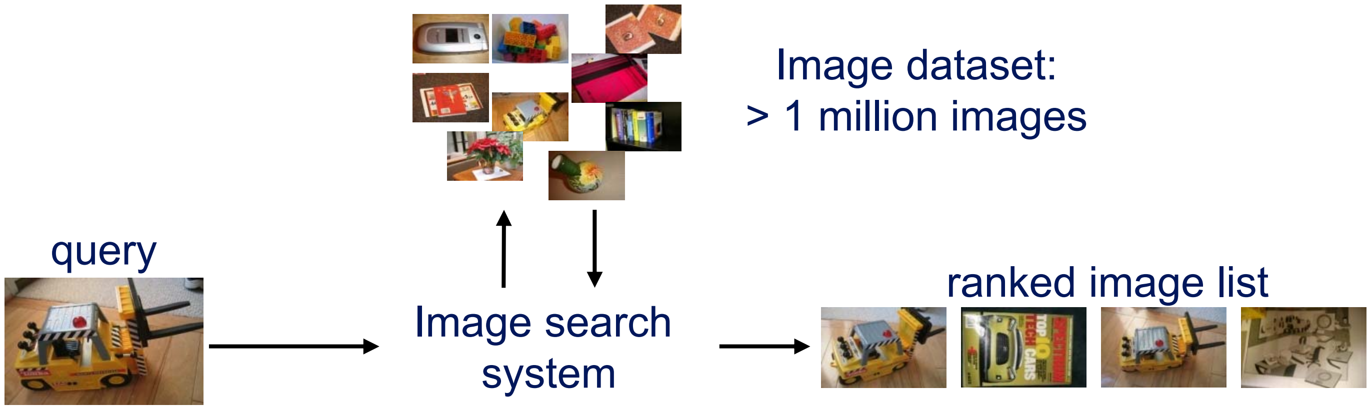
K-d tree

- 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!

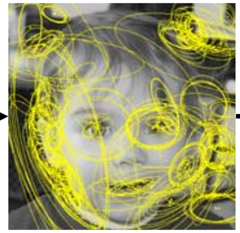
- Database representation in RAM:
  - Size of descriptors : 1 TB, search+memory intractable

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

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

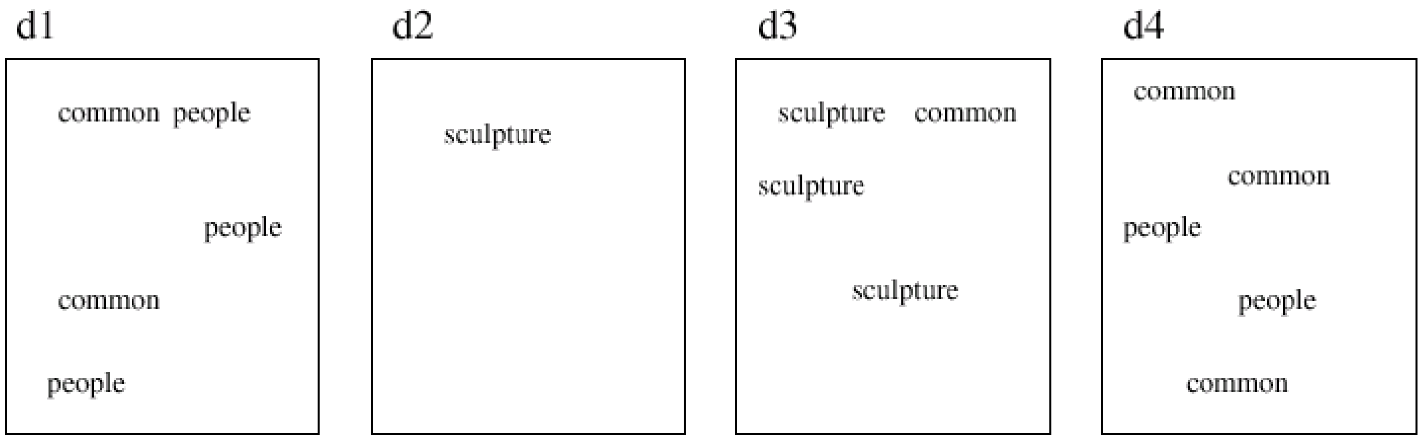
[Chum & al. 2007]

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:

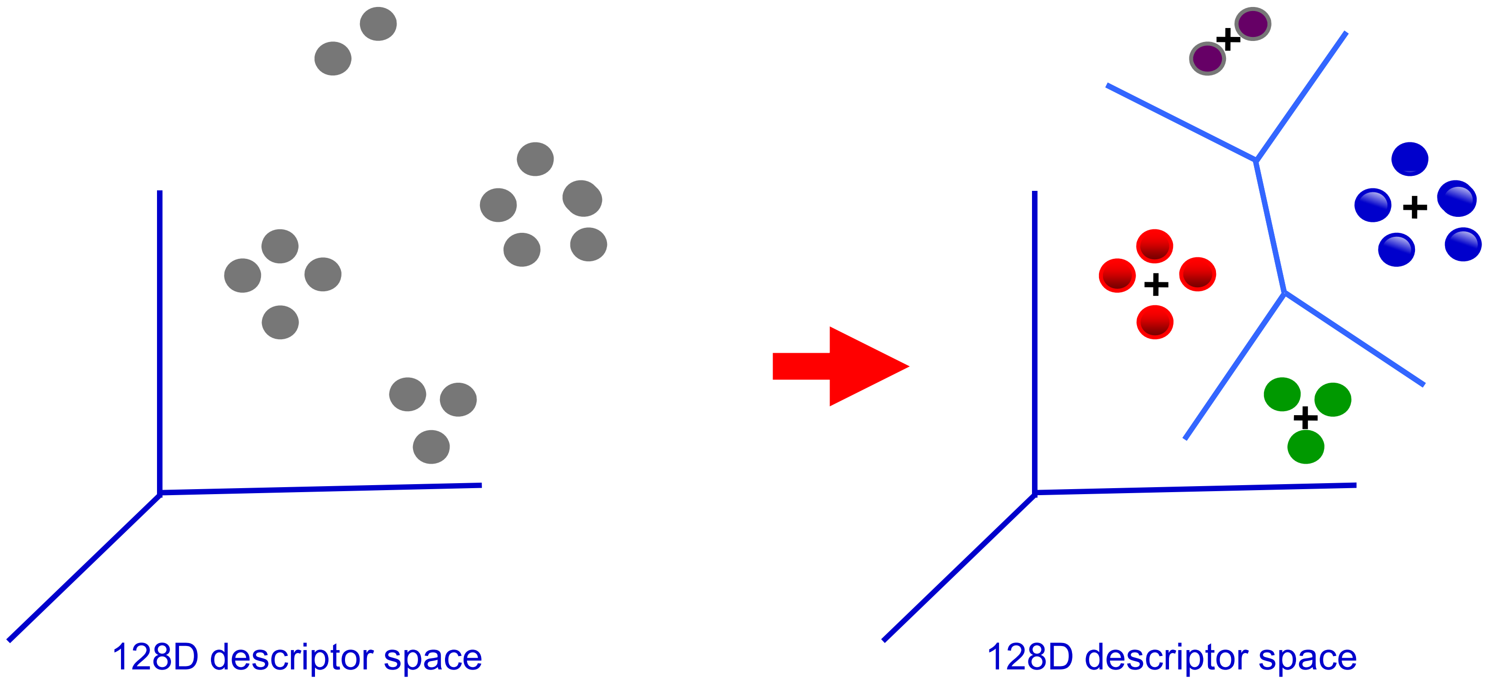


Inverted file:

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"

Build a visual vocabulary



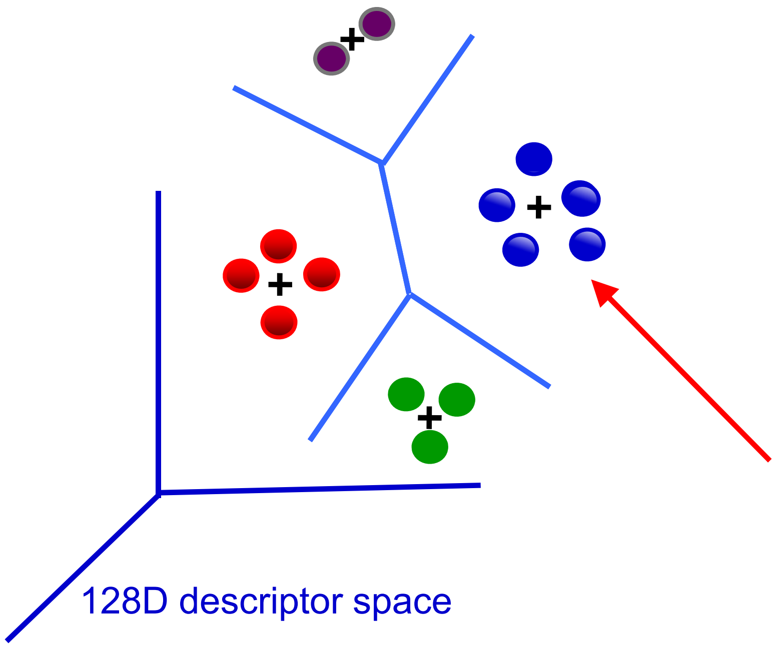
Vector quantize descriptors

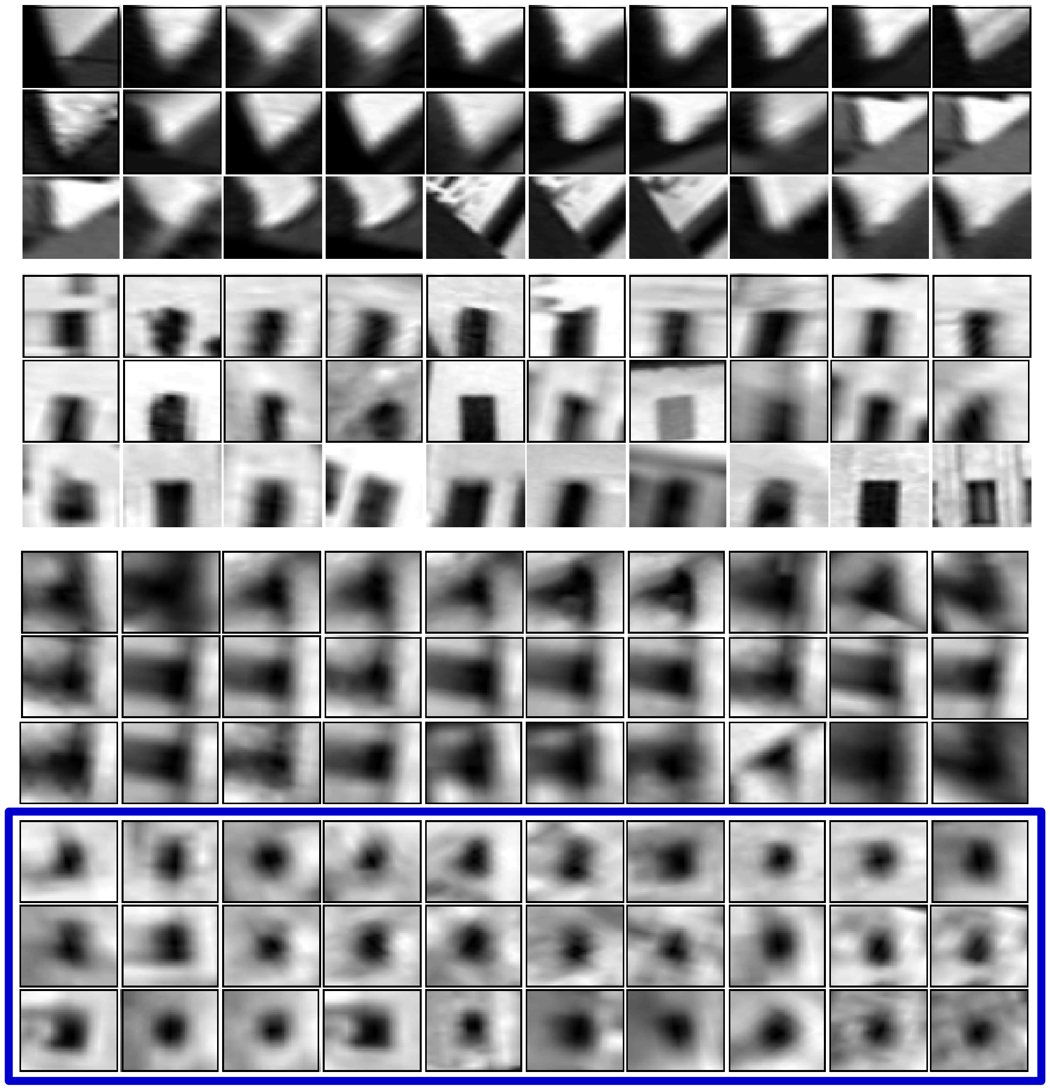
- Compute SIFT features from a subset of images
- K-means clustering (need to choose K)

[Sivic and Zisserman, ICCV 2003]

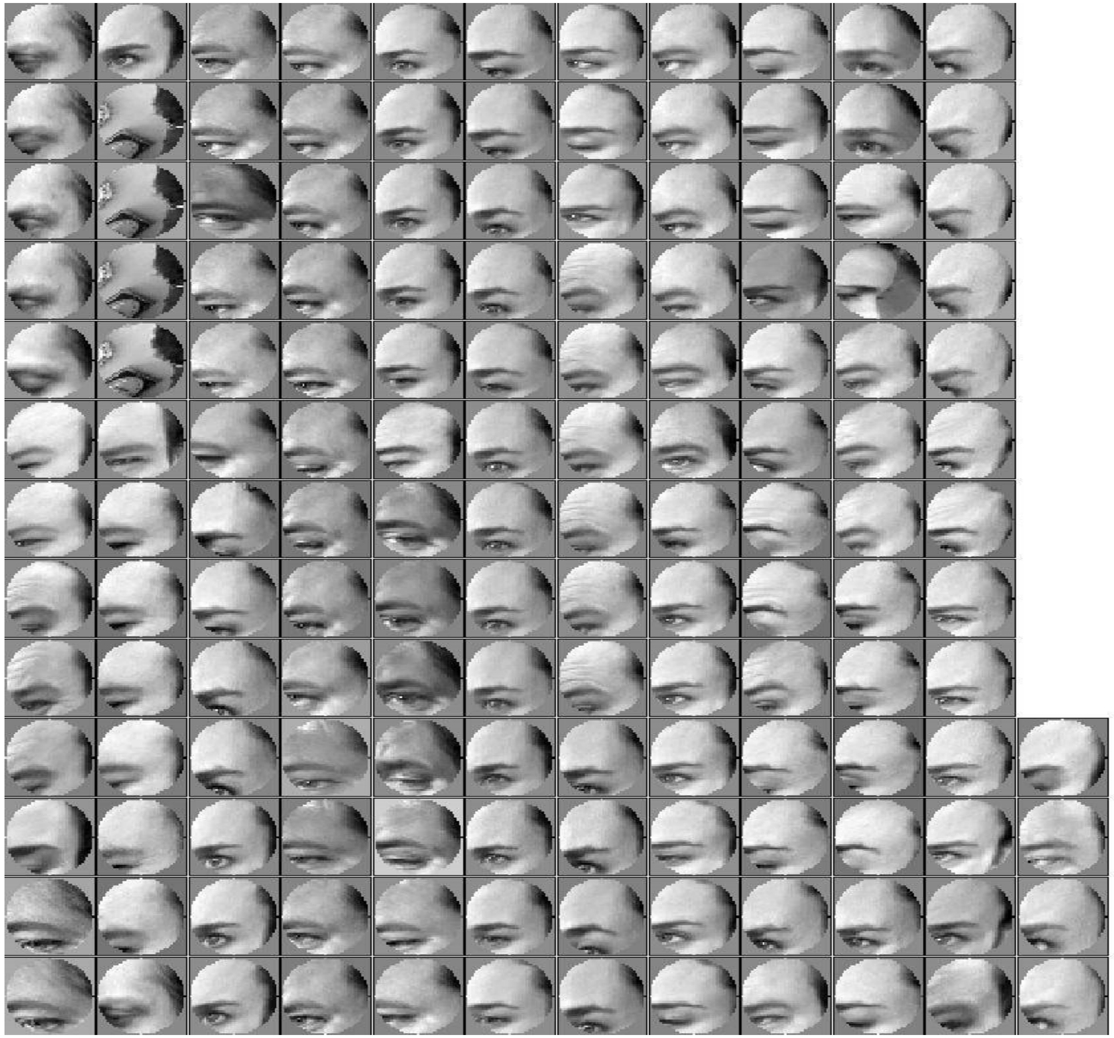
Visual words

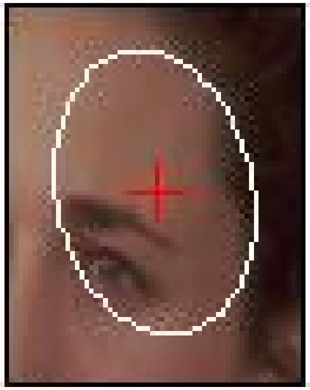
Example: each group of patches belongs to the same visual word



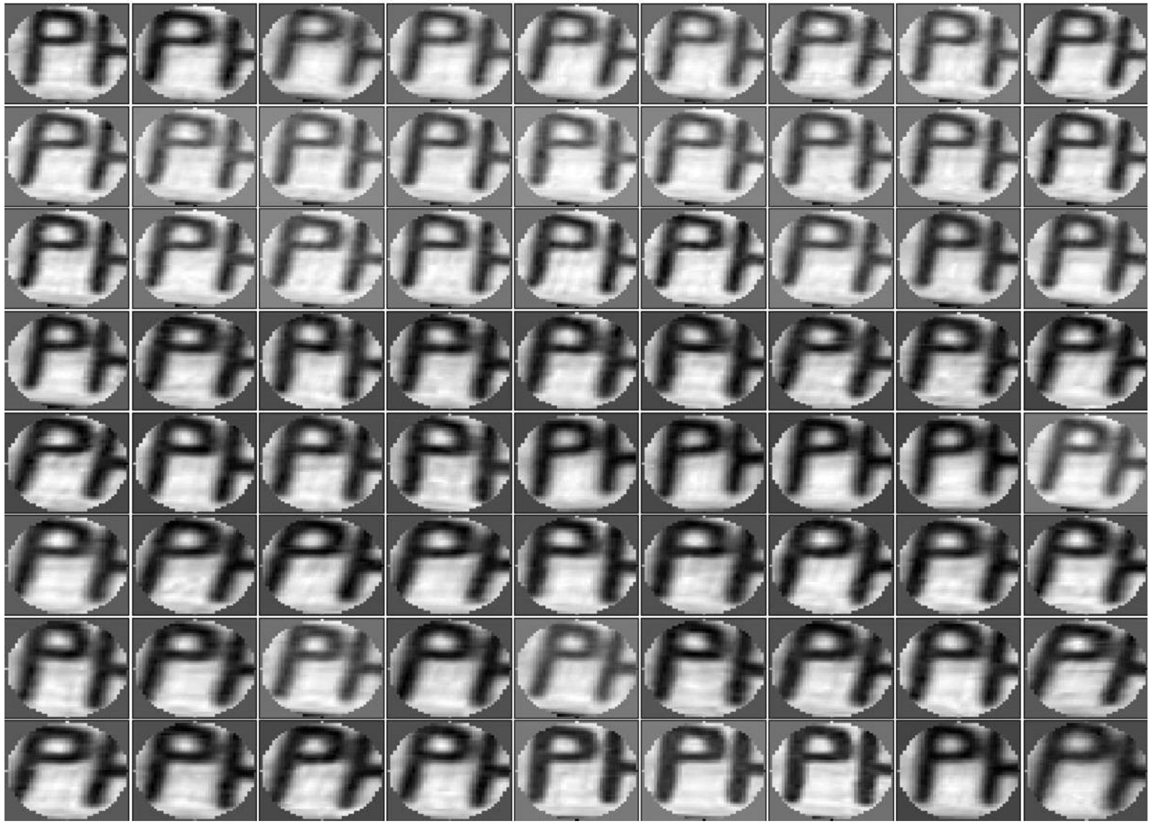


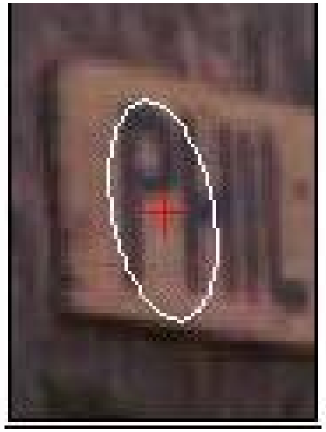
Samples of visual words (clusters on SIFT descriptors):





Samples of visual words (clusters on SIFT descriptors):



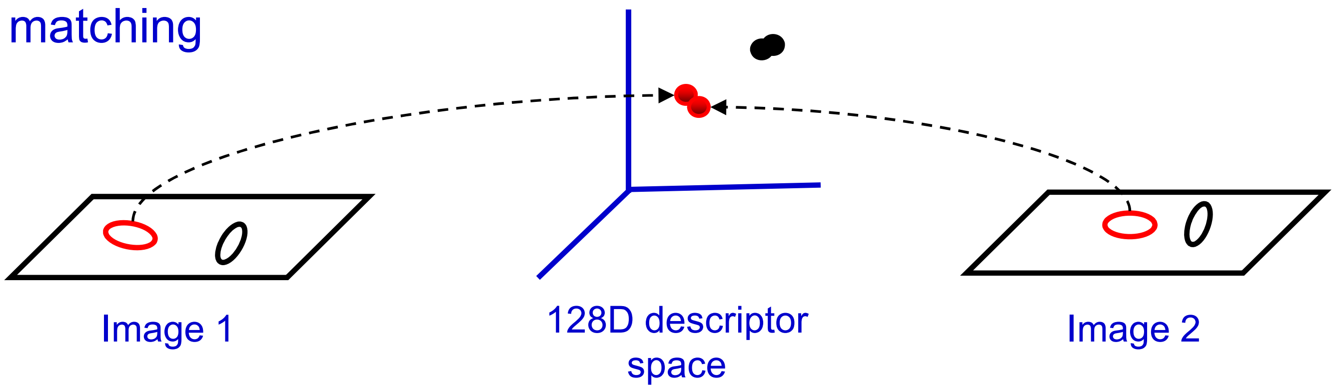


Visual words: quantize descriptor space

Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames

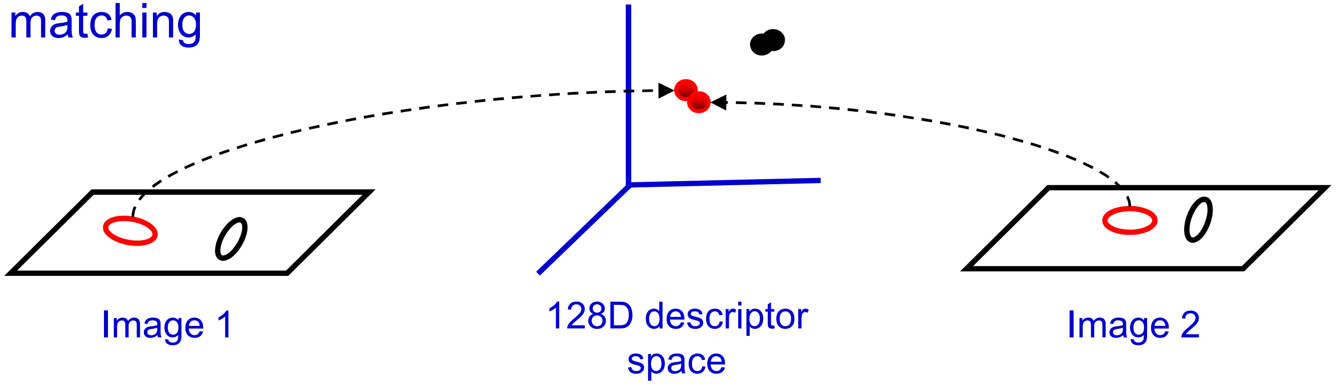


Visual words: quantize descriptor space

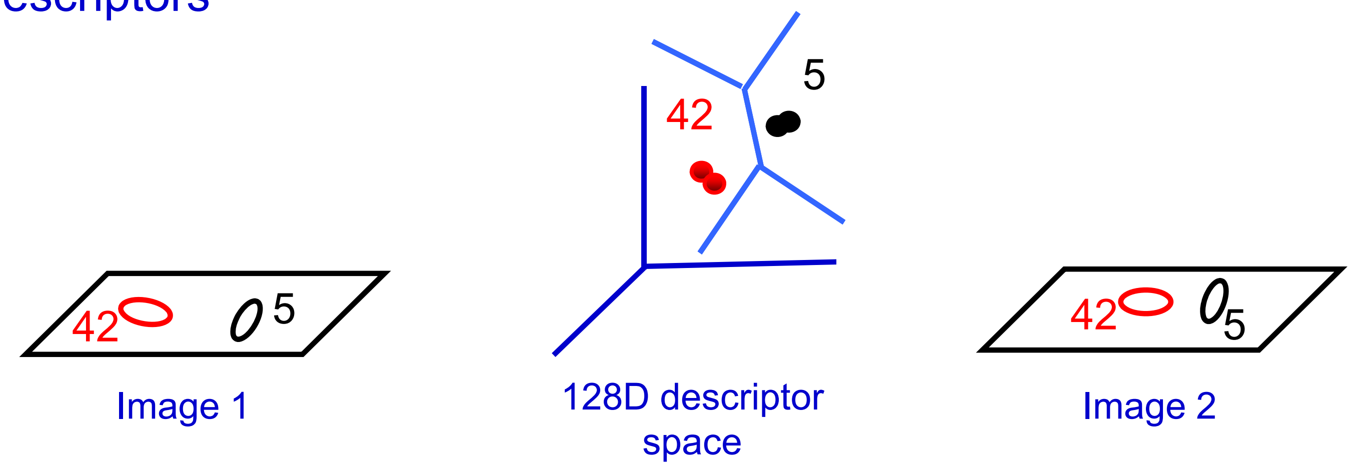
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

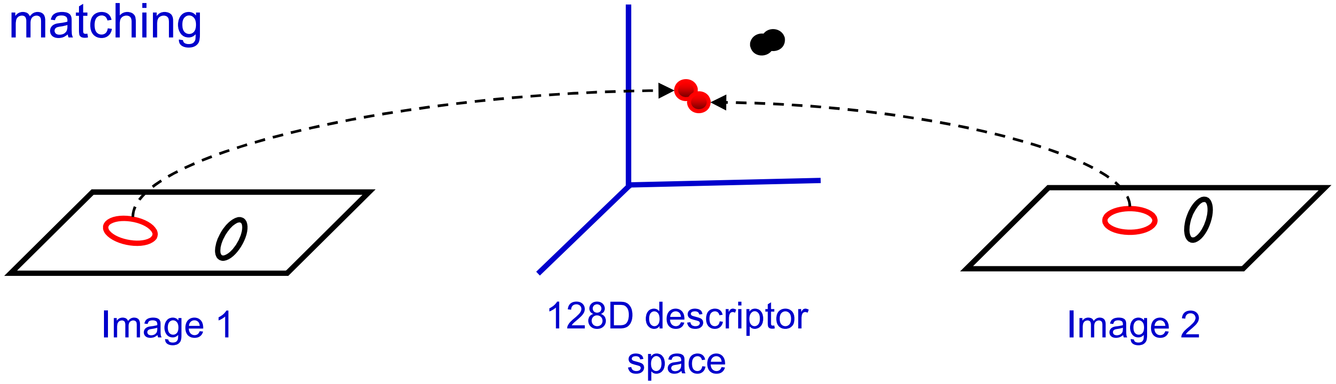


Visual words: quantize descriptor space

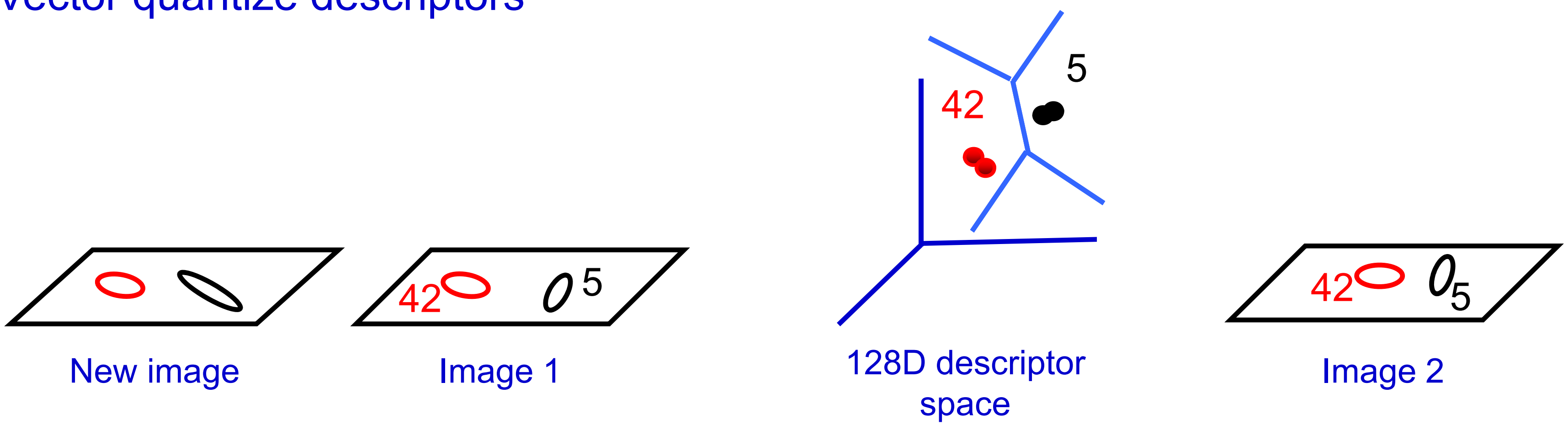
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

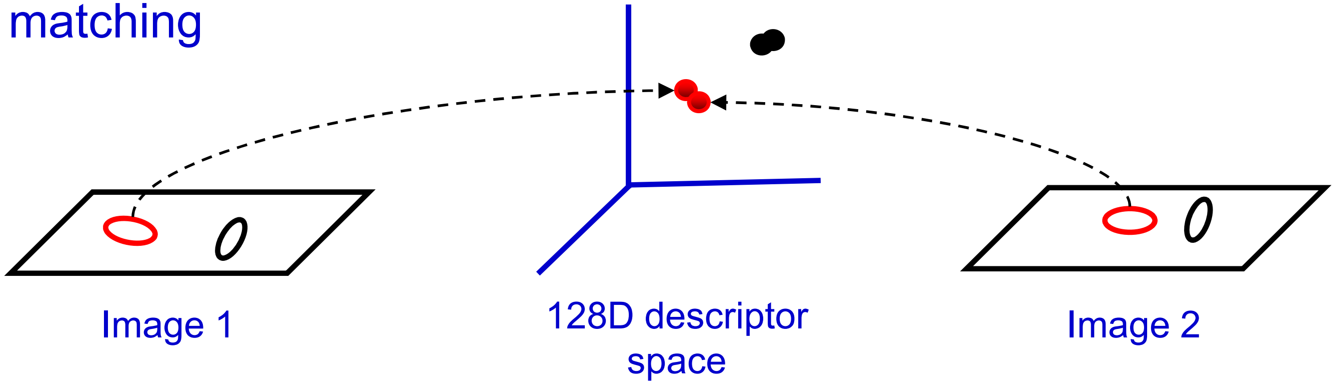


Visual words: quantize descriptor space

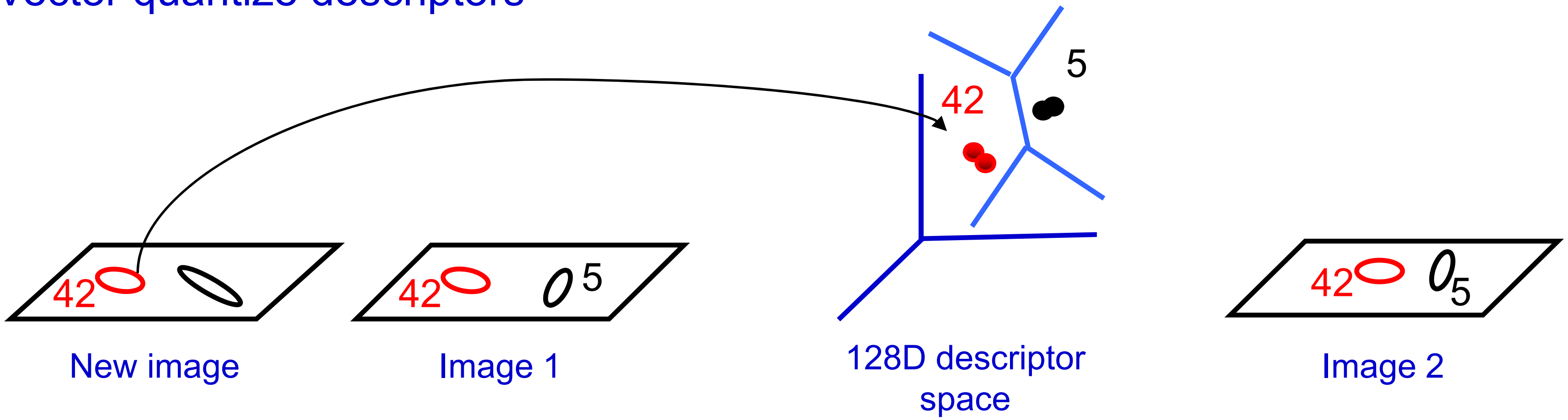
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

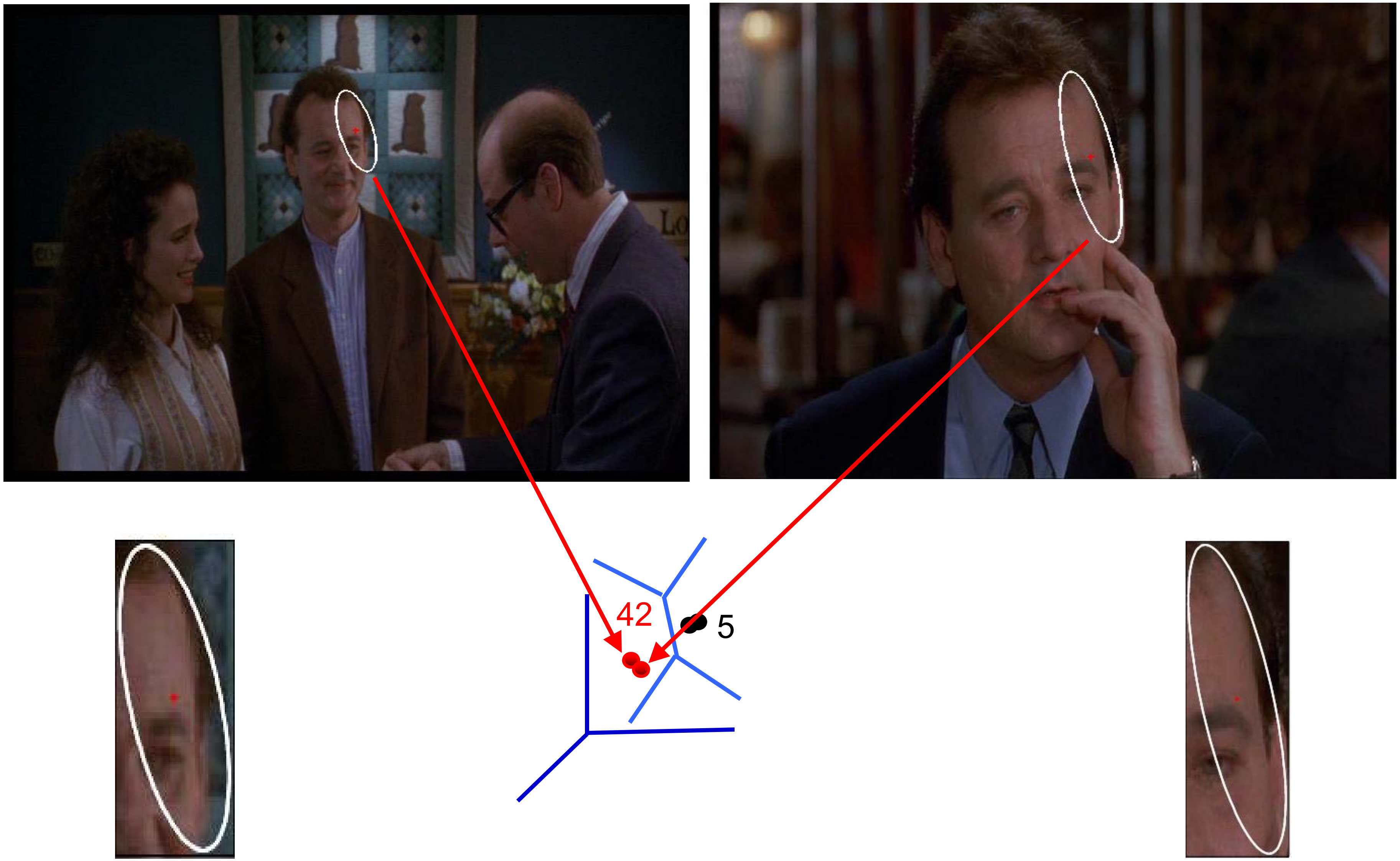
- expensive to do for all frames



Vector quantize descriptors



Vector quantize the descriptor space (SIFT)

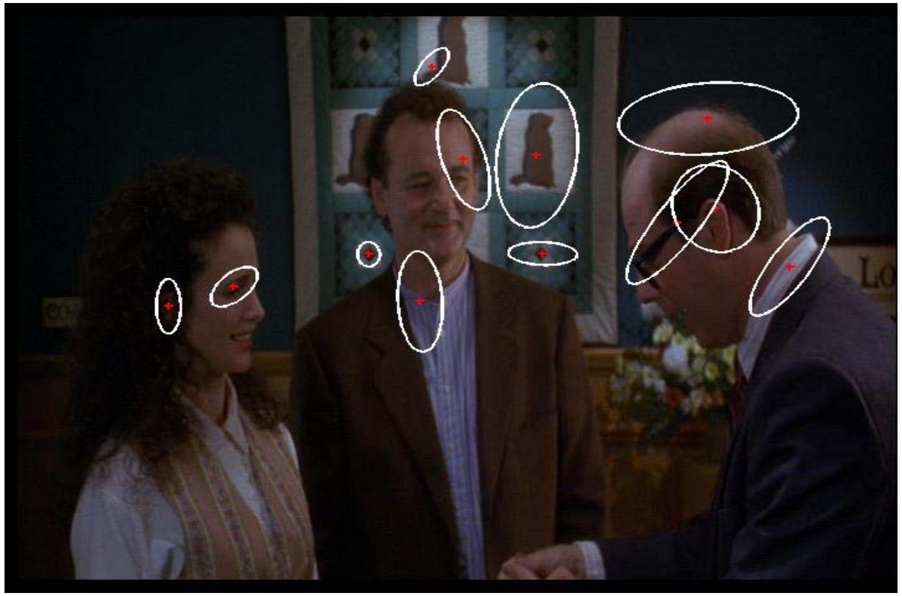


The same visual word

Representation: bag of (visual) words

Visual words are ‘iconic’ image patches or fragments

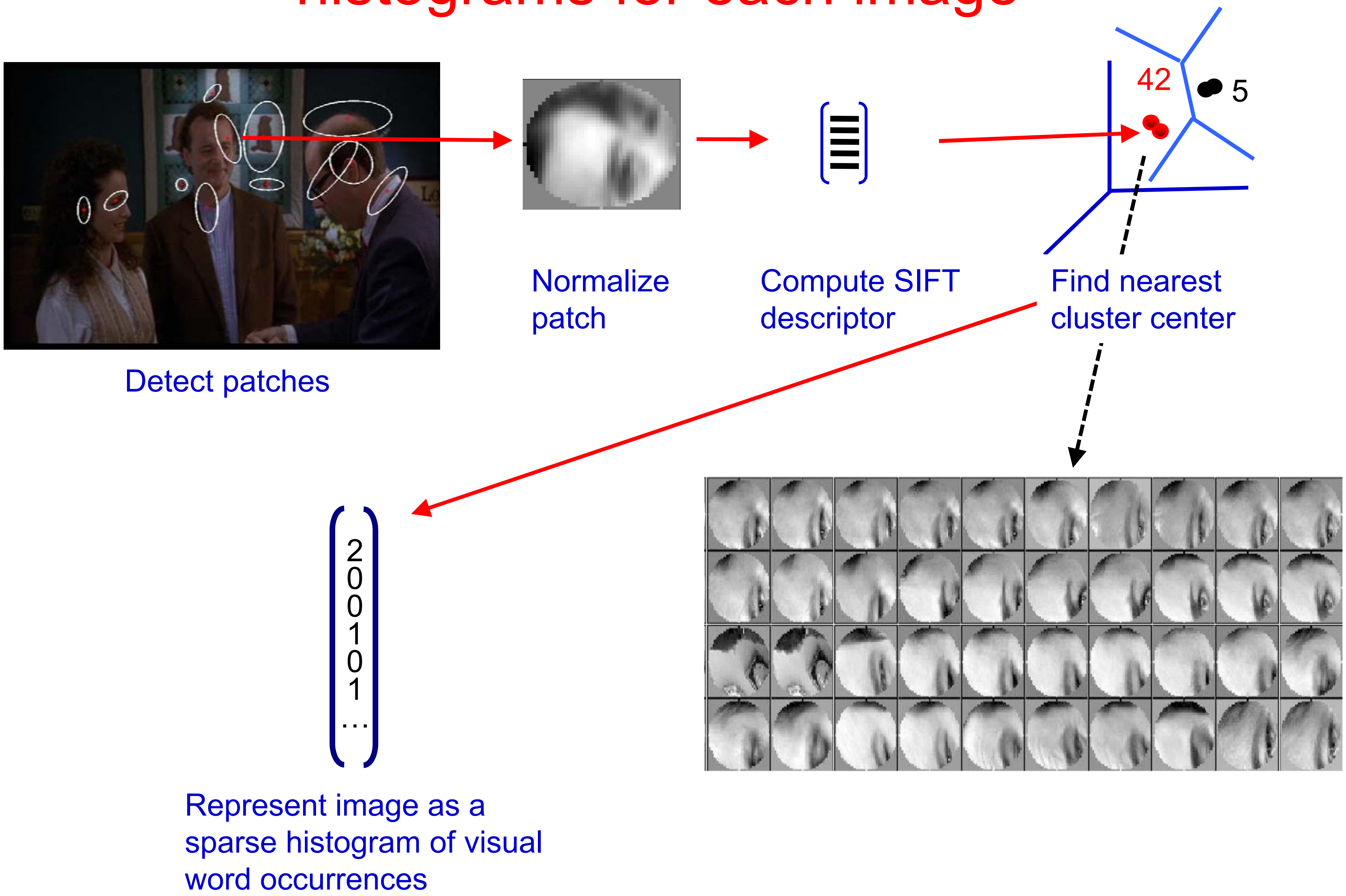
- represent their frequency of occurrence
- but not their position



Image

Collection of visual words

Offline: Assign visual words and compute histograms for each image



Offline: create an index

frame #5

frame #10

Word number	Posting list
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

(i.e. like the “book index”)

At run time

frame #5

frame #10

Word number	Posting list
1	5,10, ...
2	10,...
...	...

- User specifies a query region
- Generate a short-list of images using 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
    3. Compute similarity for images sharing at least one word

At run time

frame #5

frame #10

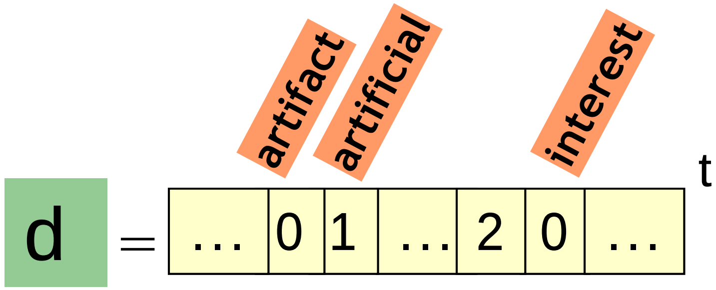
Word number		Posting list
1	→	5,10, ...
2	→	10,...
...		...

- 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)

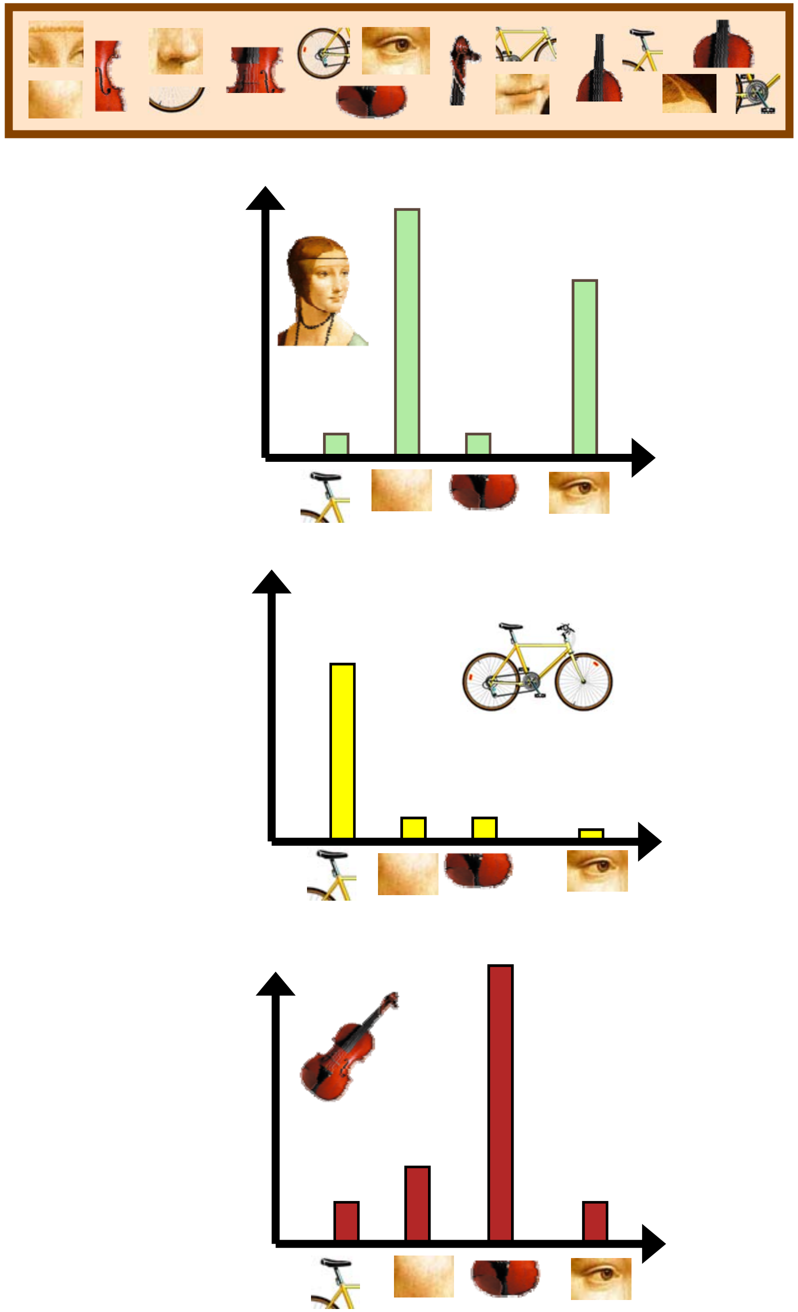
Another interpretation: Bags of visual words

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

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

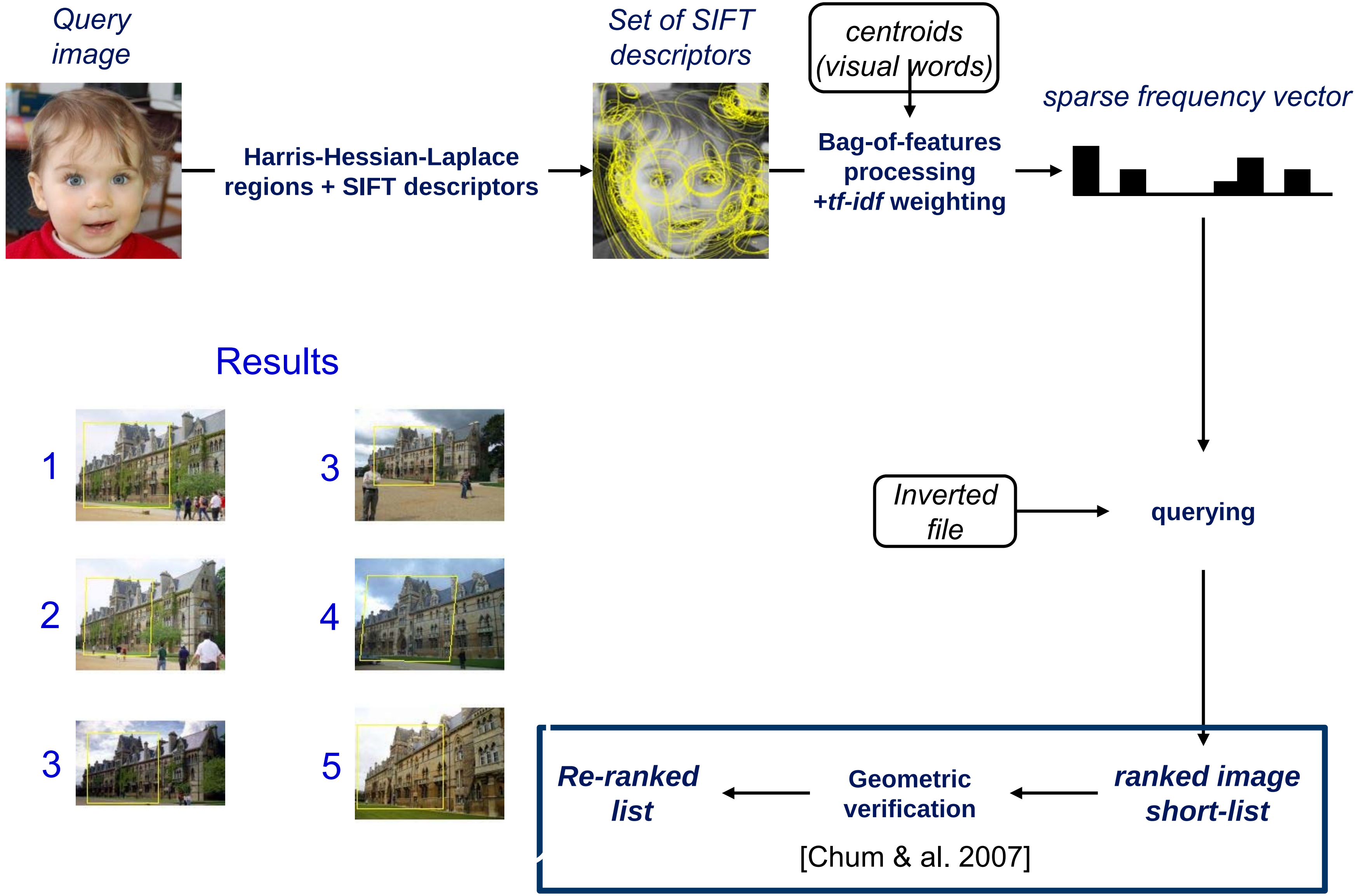
where t_i is the number of occurrences of visual word i

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

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

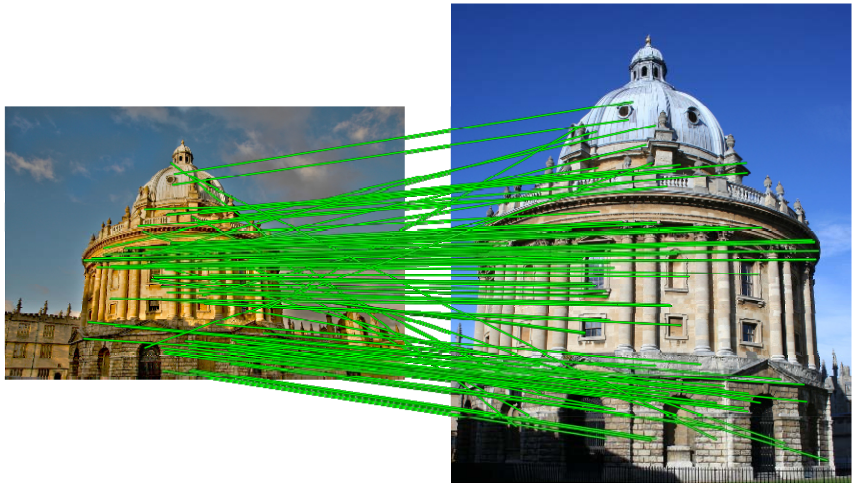
Scalar product can be computed efficiently using inverted file

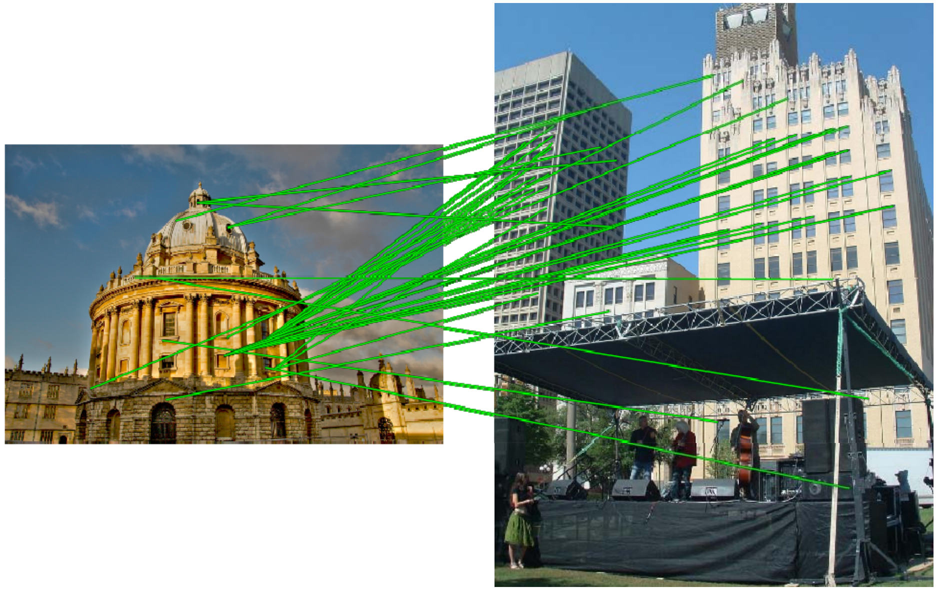
Bag-of-features [Sivic&Zisserman'03]



Geometric verification

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





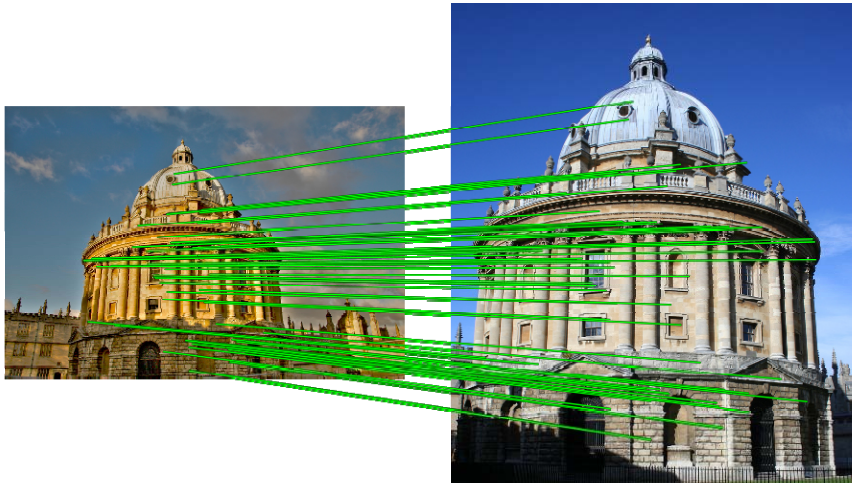
Both images have many matches – which is correct?

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

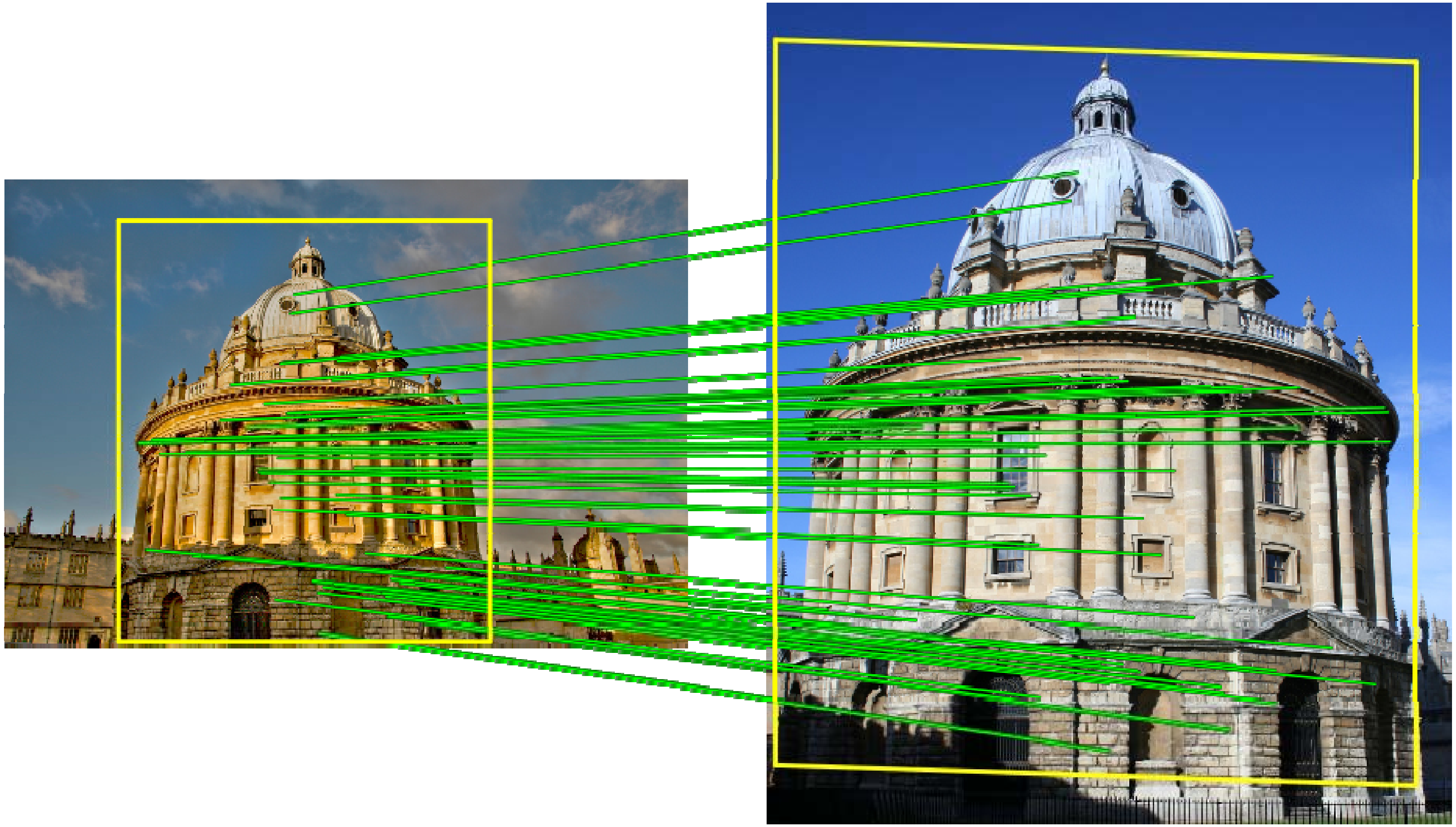


Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**

Geometric verification

Gives **localization** of the object



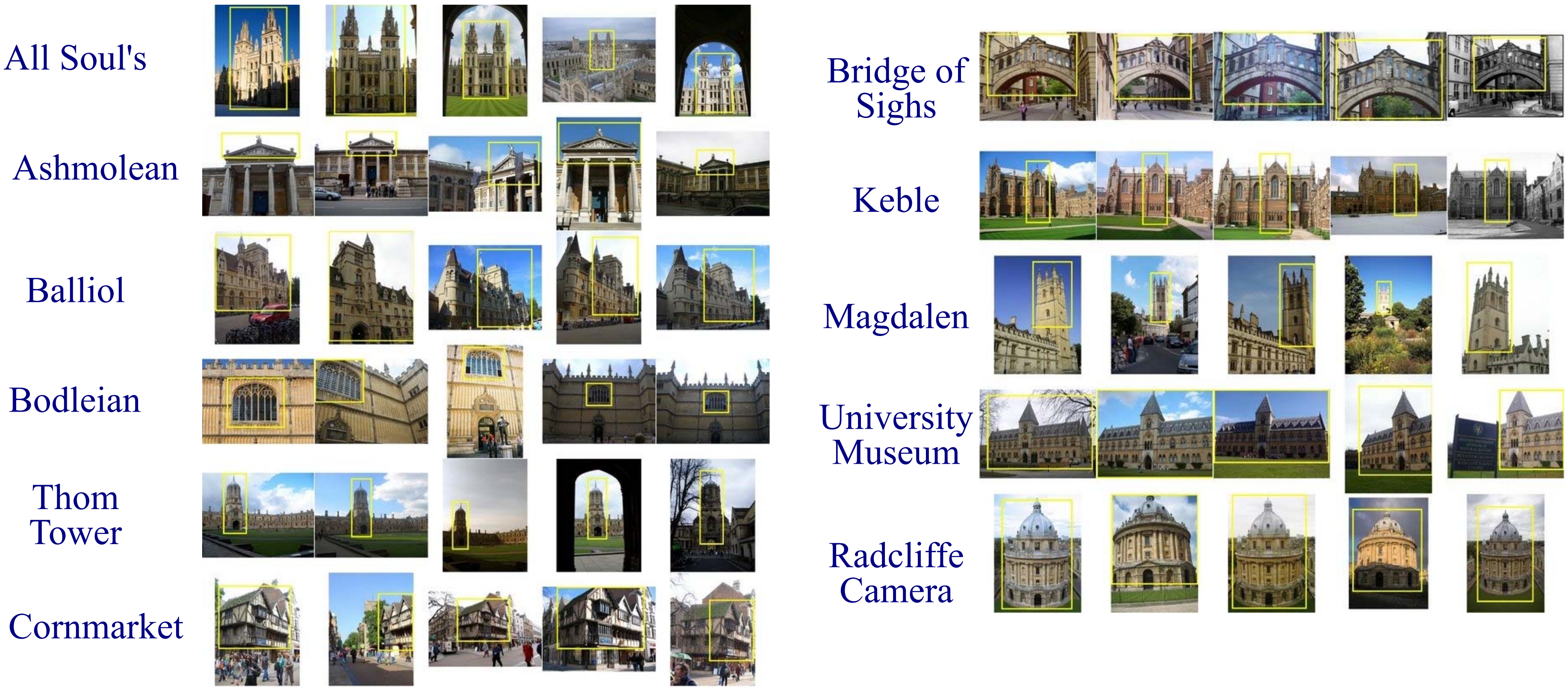
Geometric verification – example

1. Query

2. Initial retrieval set (bag of words model)

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

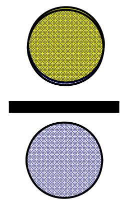
Evaluation dataset: Oxford buildings



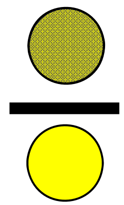
- Ground truth obtained for 11 landmarks
- Evaluate performance by mean Average Precision

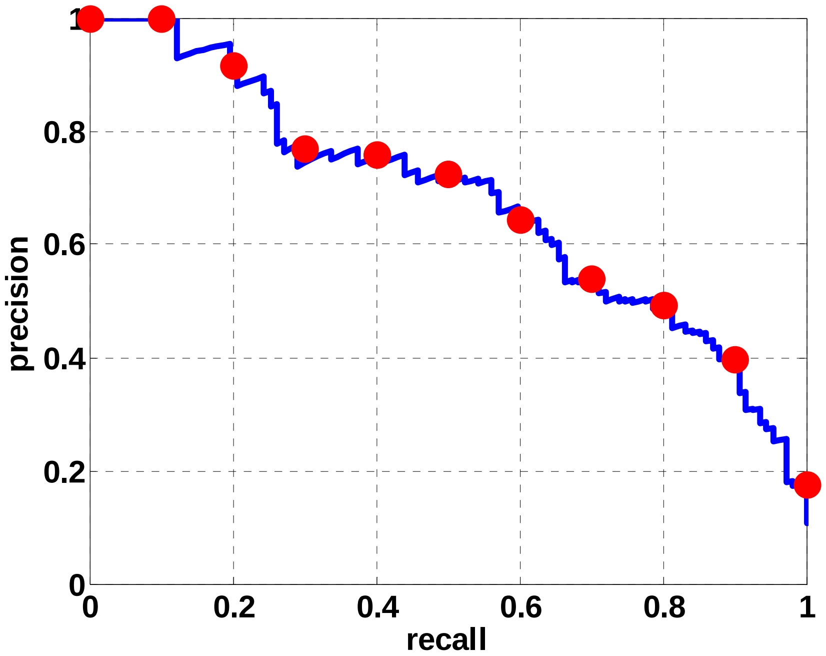
Measuring retrieval performance: Precision - Recall

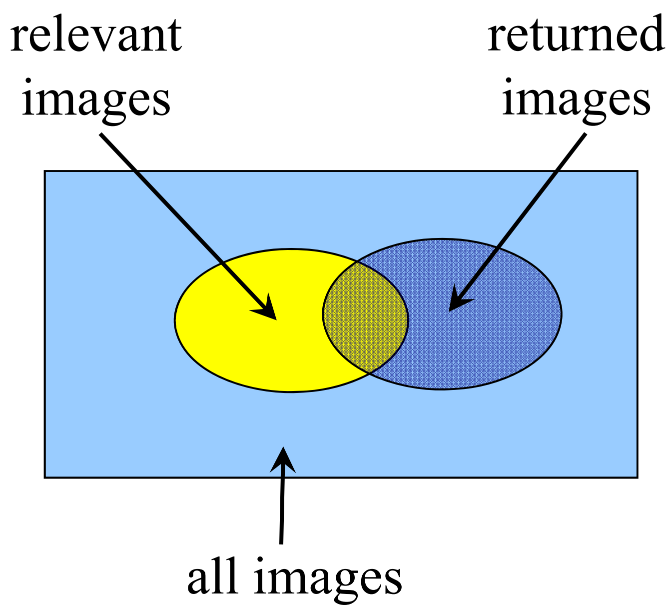
- Precision: % of returned images that are relevant



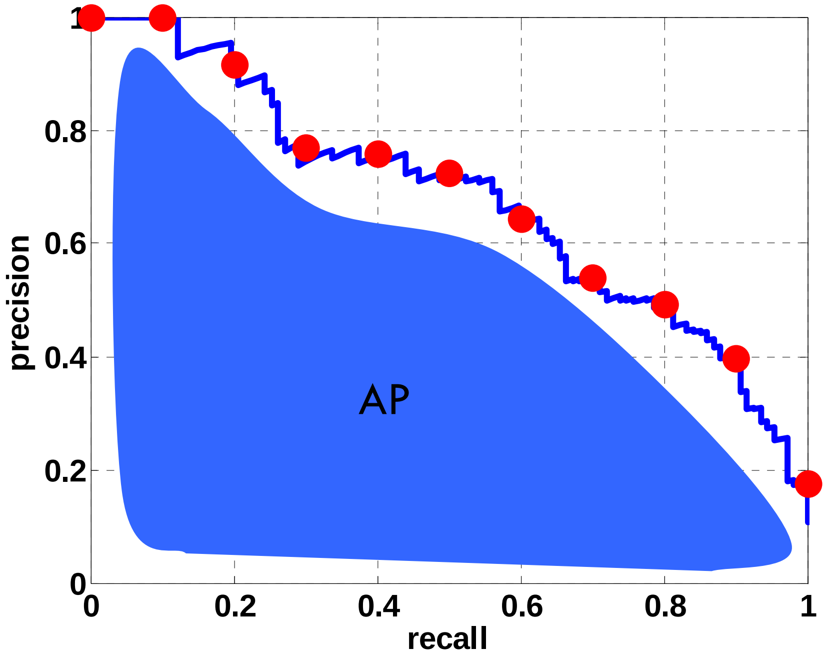
- Recall: % of relevant images that are returned





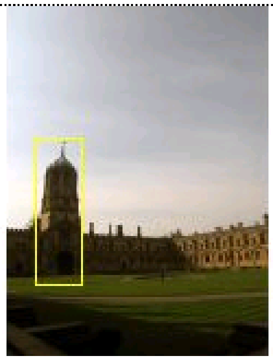


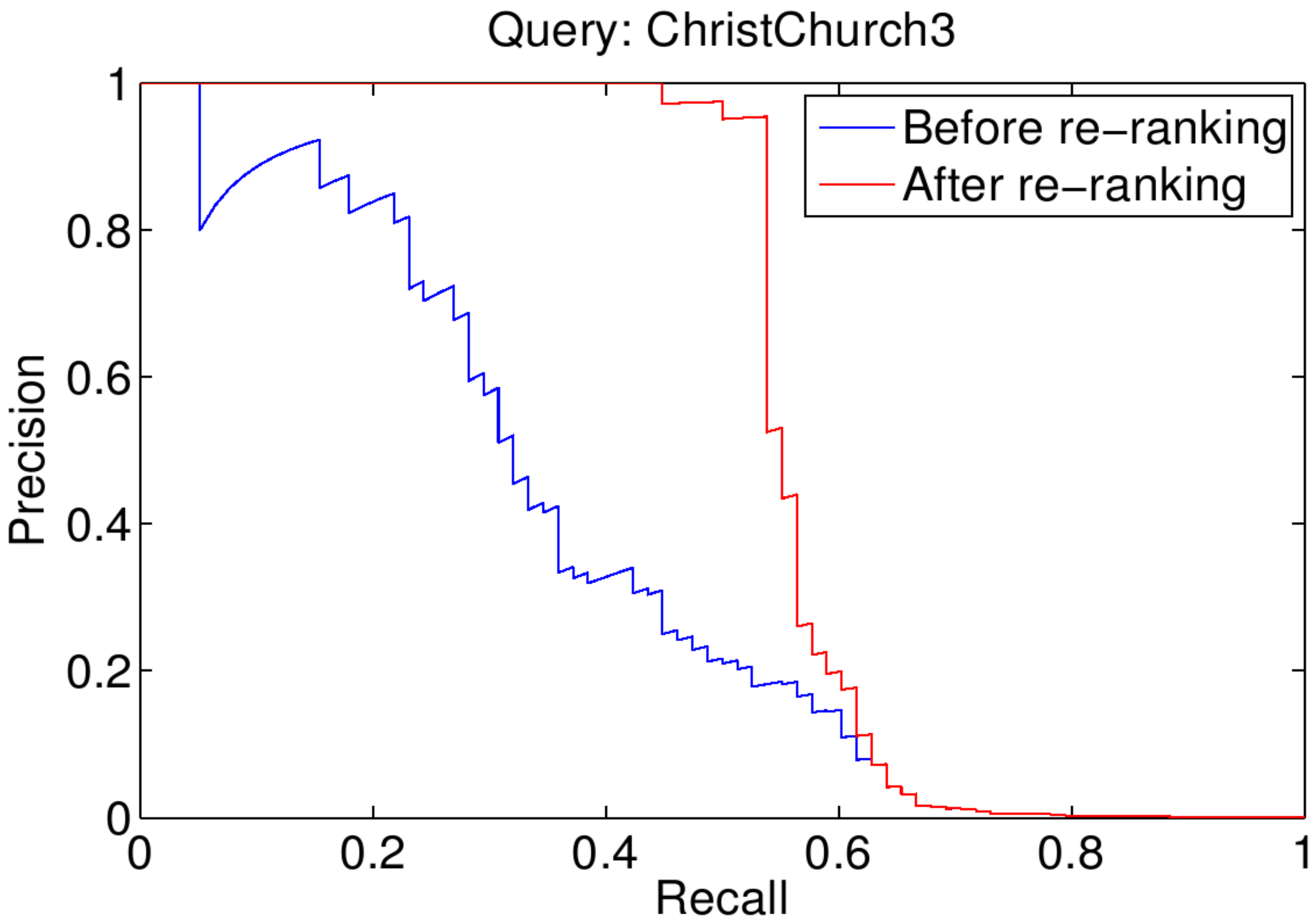
Average Precision



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

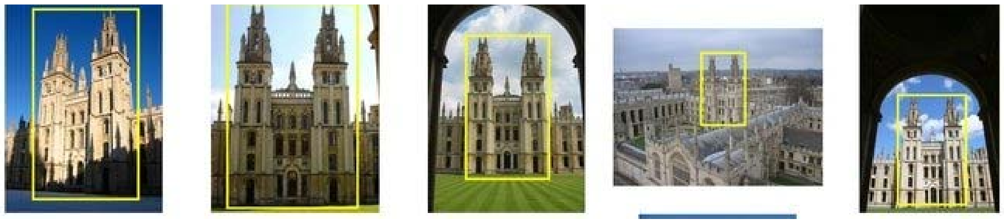
Performance measured by mean Average Precision (mAP) over 55 queries on 100K or 1.1M image datasets

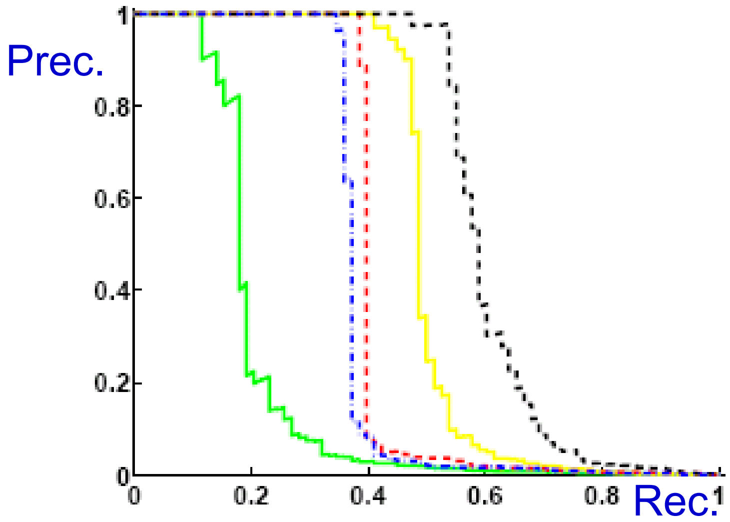




Query: ChristChurch3
Before re−ranking
After re−ranking
Precision
Recall
0
0.2
0.4
0.6
0.8
1

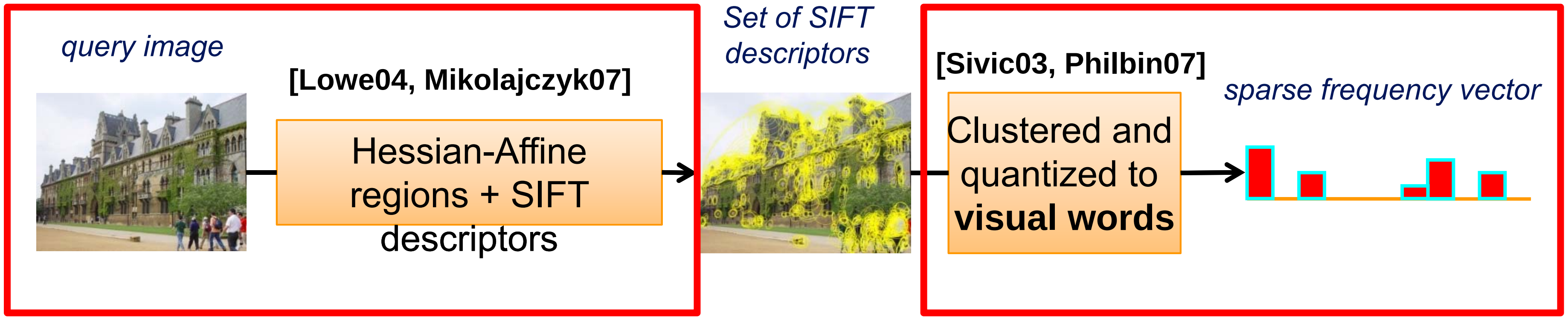
Query images





- high precision at low recall (like google)
- 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

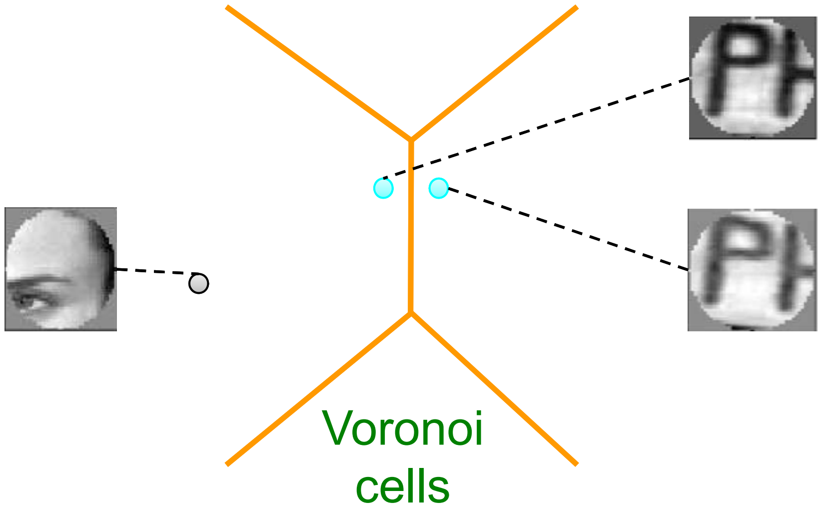
Better quantization

Consequence: Visual word in query is missing

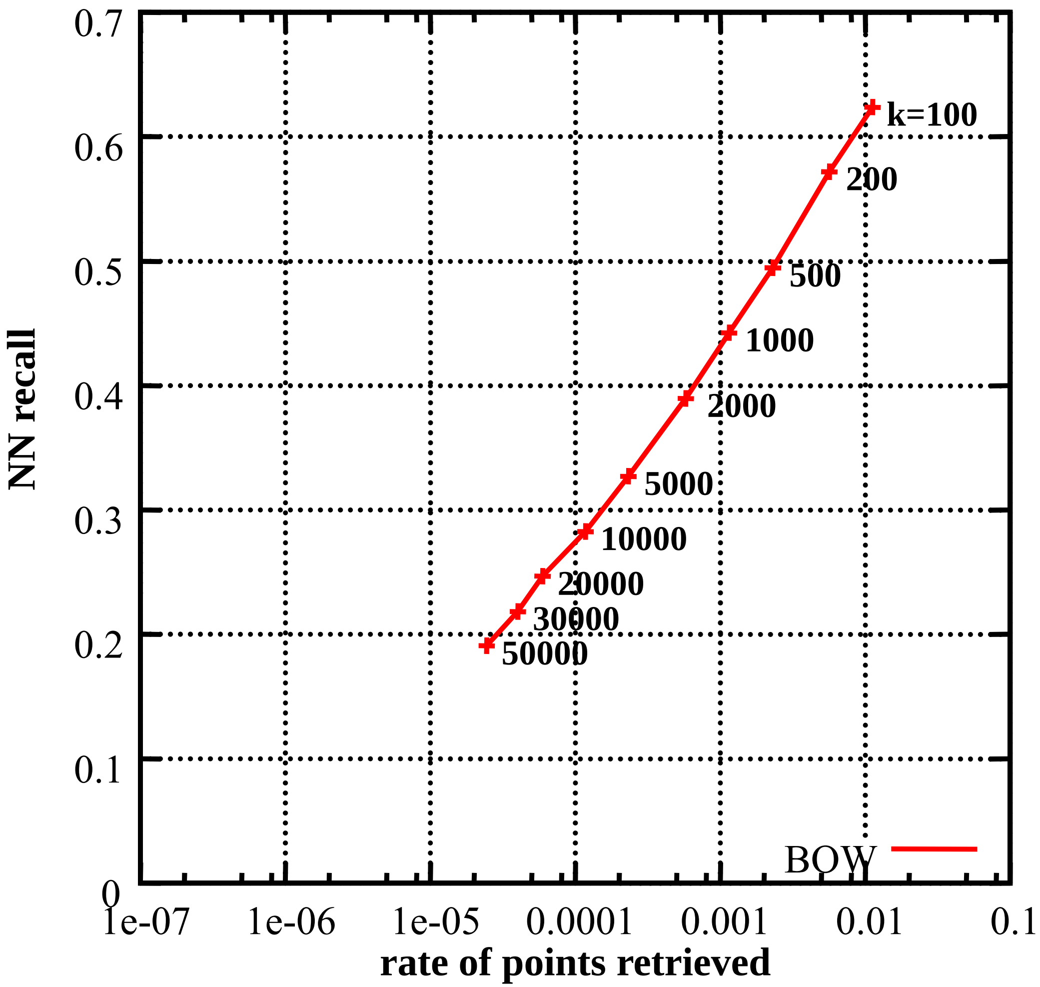
Quantization errors

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

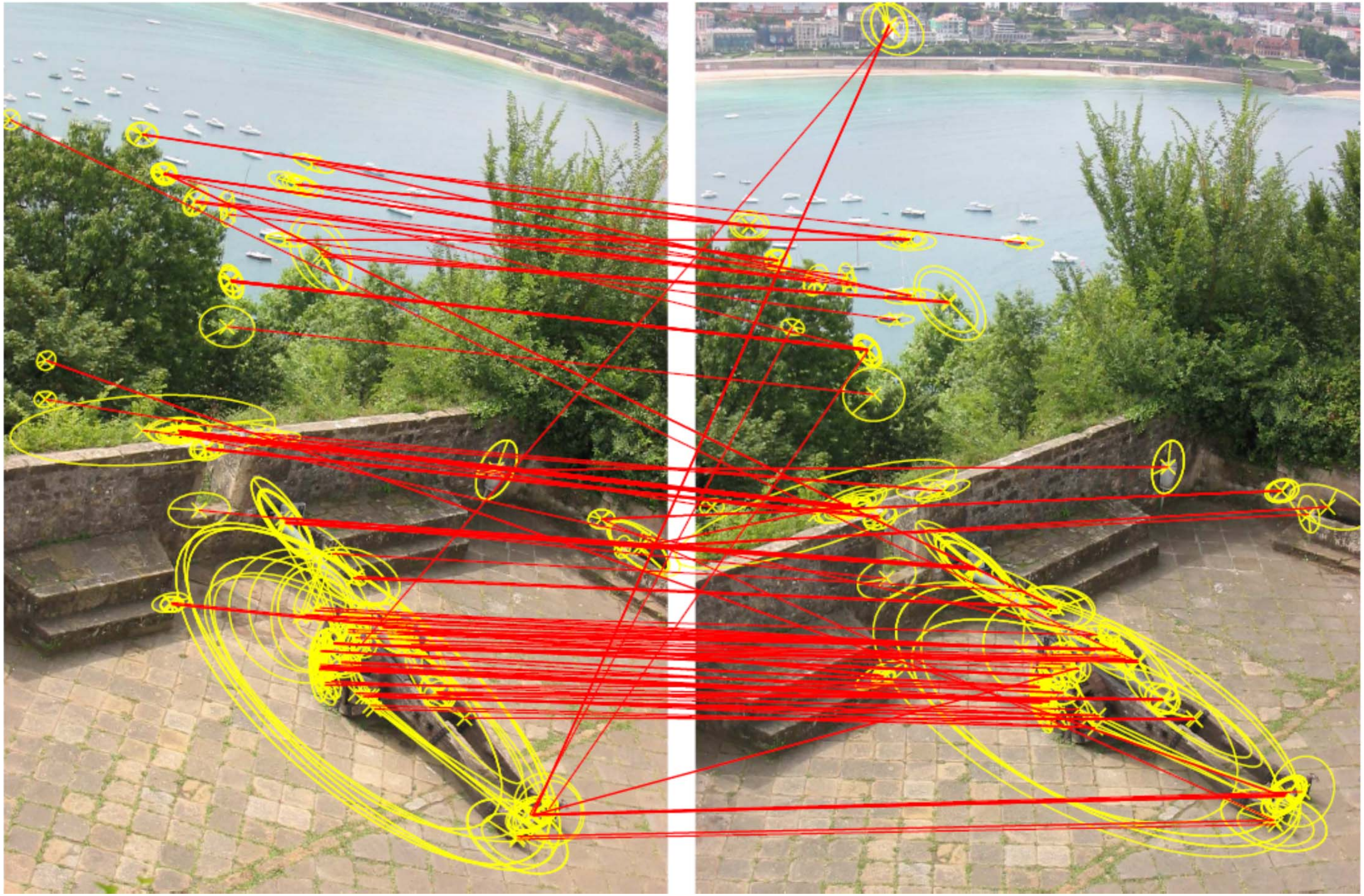


ANN evaluation of bag-of-features

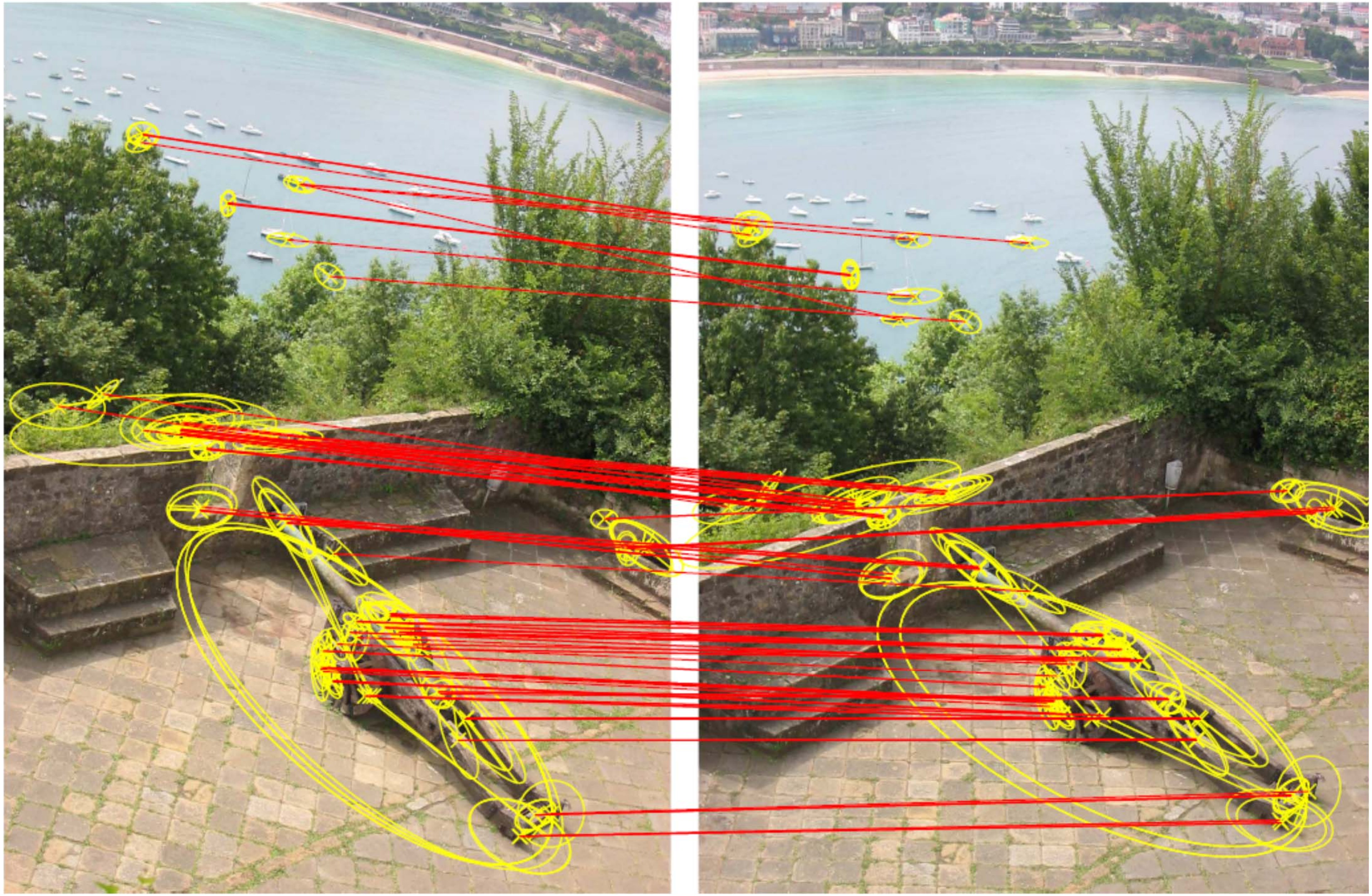


- ANN algorithms returns a list of potential neighbors
- **NN recall** = probability that *the* NN is in this list
- **NN precision**: = proportion of vectors in the short-list
- In BOF, this trade-off is managed by the number of clusters *k*

20K visual word: false matches



200K visual word: good matches missed



Problem with bag-of-features

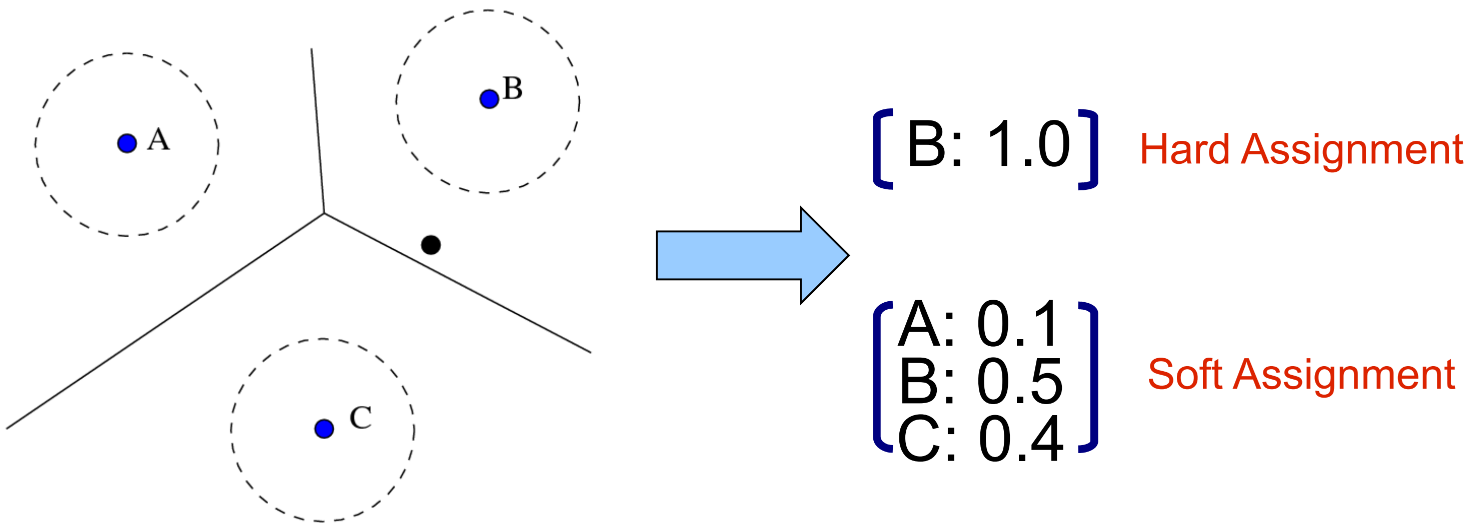
- The matching performed by BOF is weak
  - for a “small” visual dictionary: too many false matches
  - for a “large” visual dictionary: many true matches are missed

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

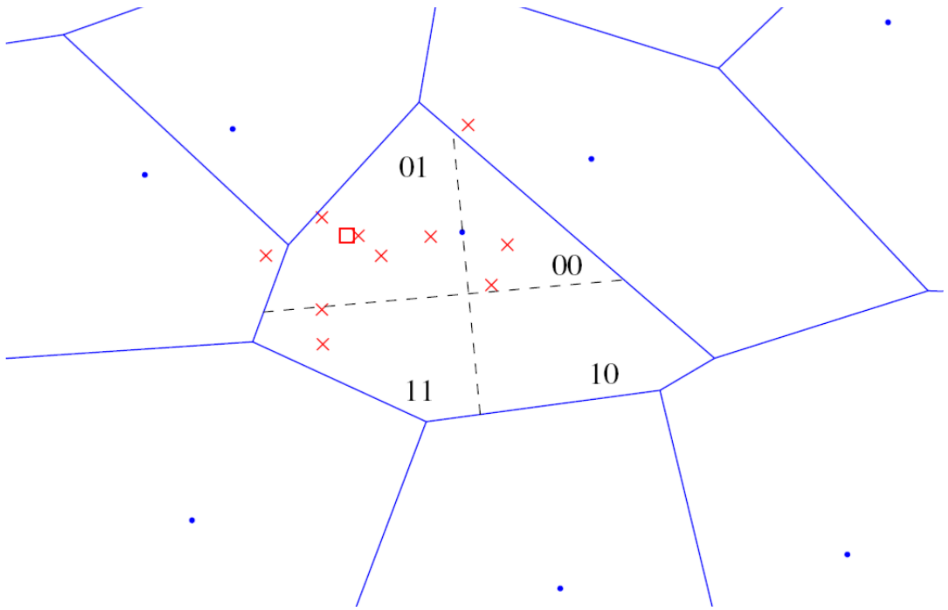
[Philbin et al. 2008, Van Gemert et al. 2008]



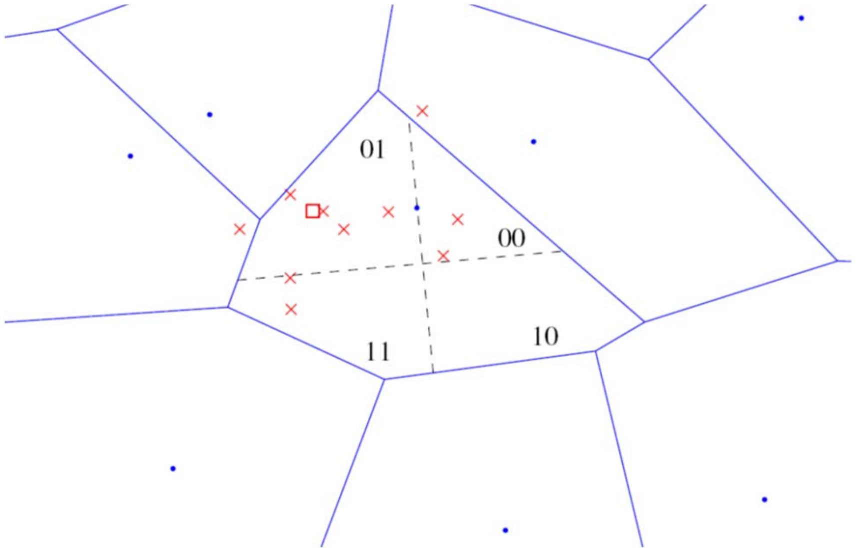
Beyond bag-of-visual-words

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

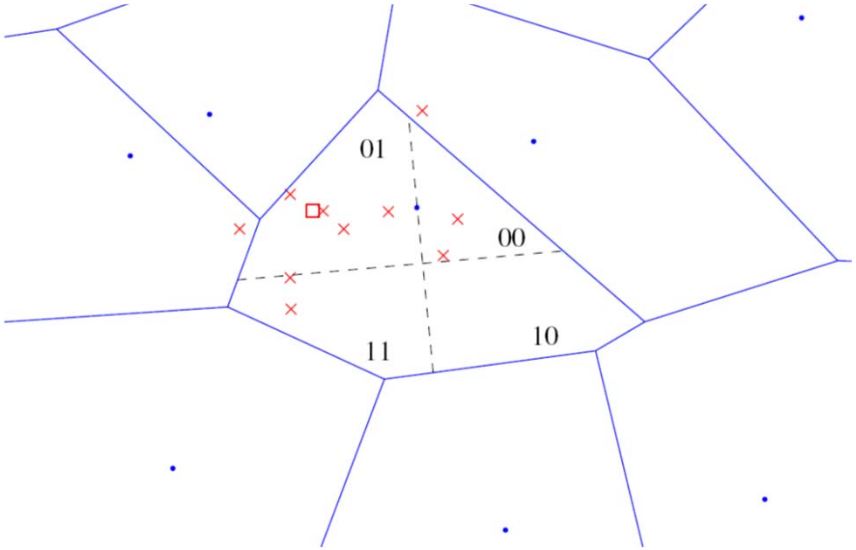
+ short binary vector *b(x)* for an additional localization in the Voronoi cell

Two descriptors x and y match iif

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

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

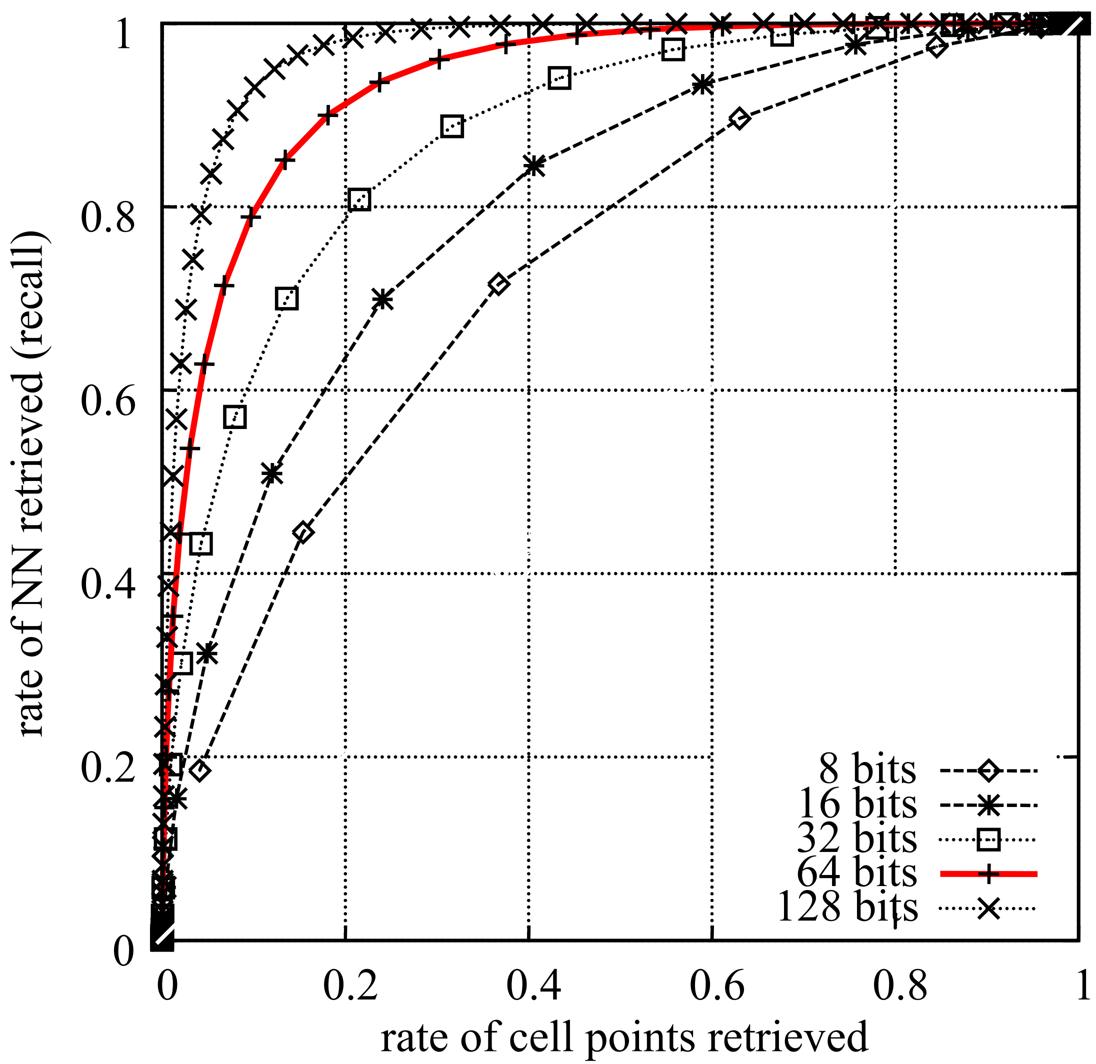
- **Off-line** (given a quantizer)
  - draw an orthogonal projection matrix *P* of size $d_b \times d$

  $\rightarrow$ 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

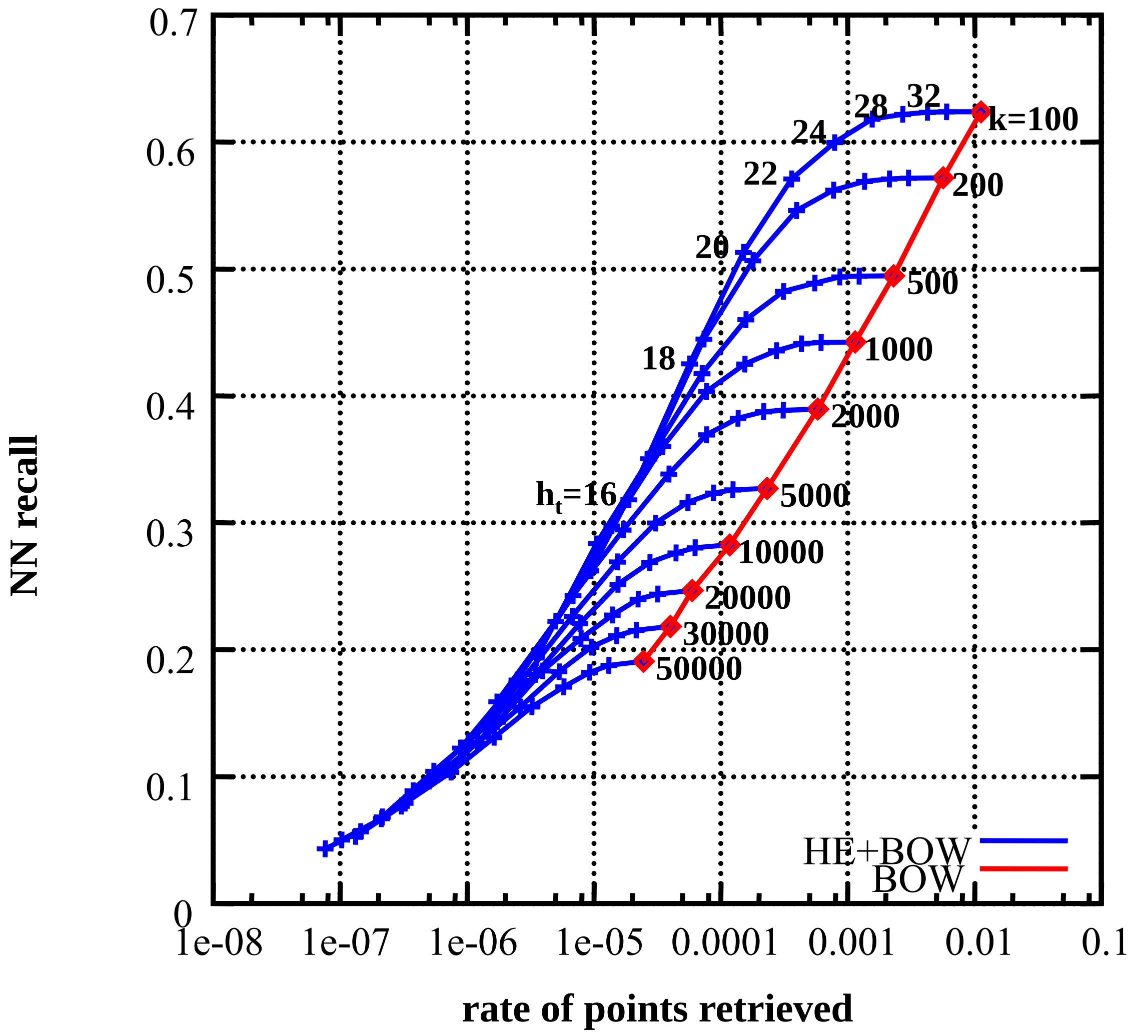


Trade-off between memory usage and accuracy

→More bits yield higher accuracy

In practice, 64 bits (8 byte)

ANN evaluation of Hamming Embedding



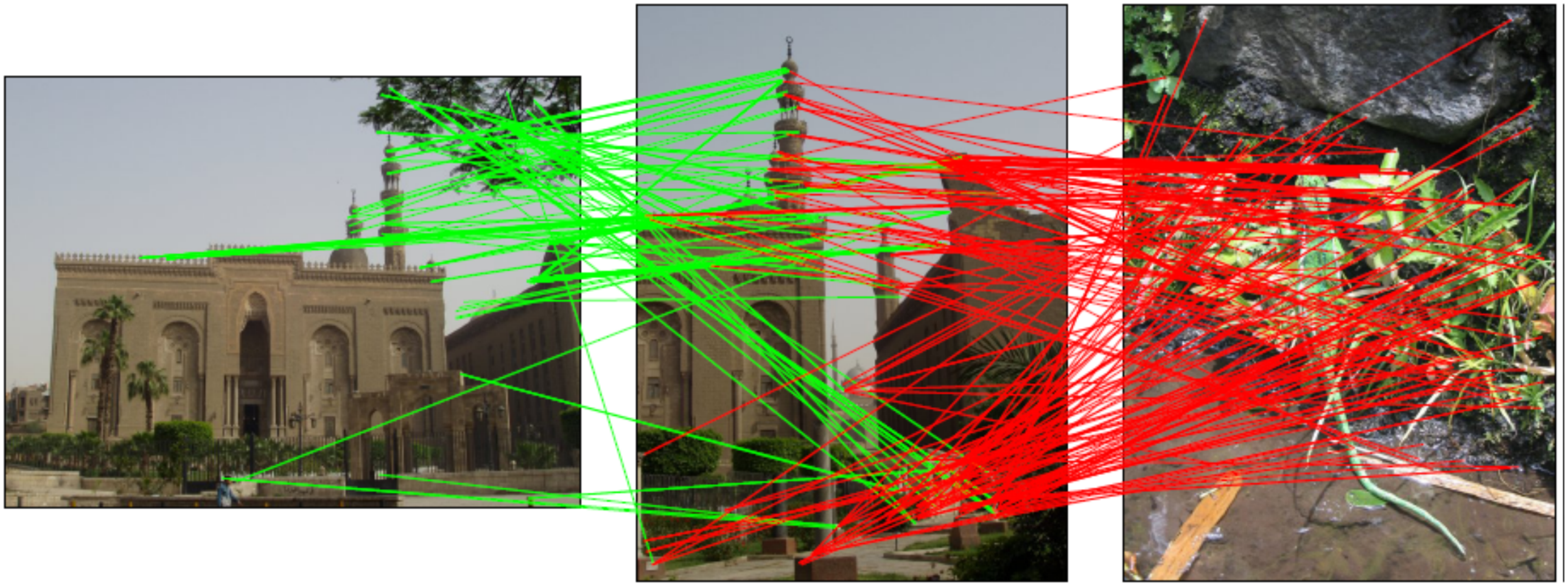
compared to BOW: at least 10 times less points in the short-list for the same level of NN recall

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

Matching points - 20k word vocabulary

201 matches

240 matches

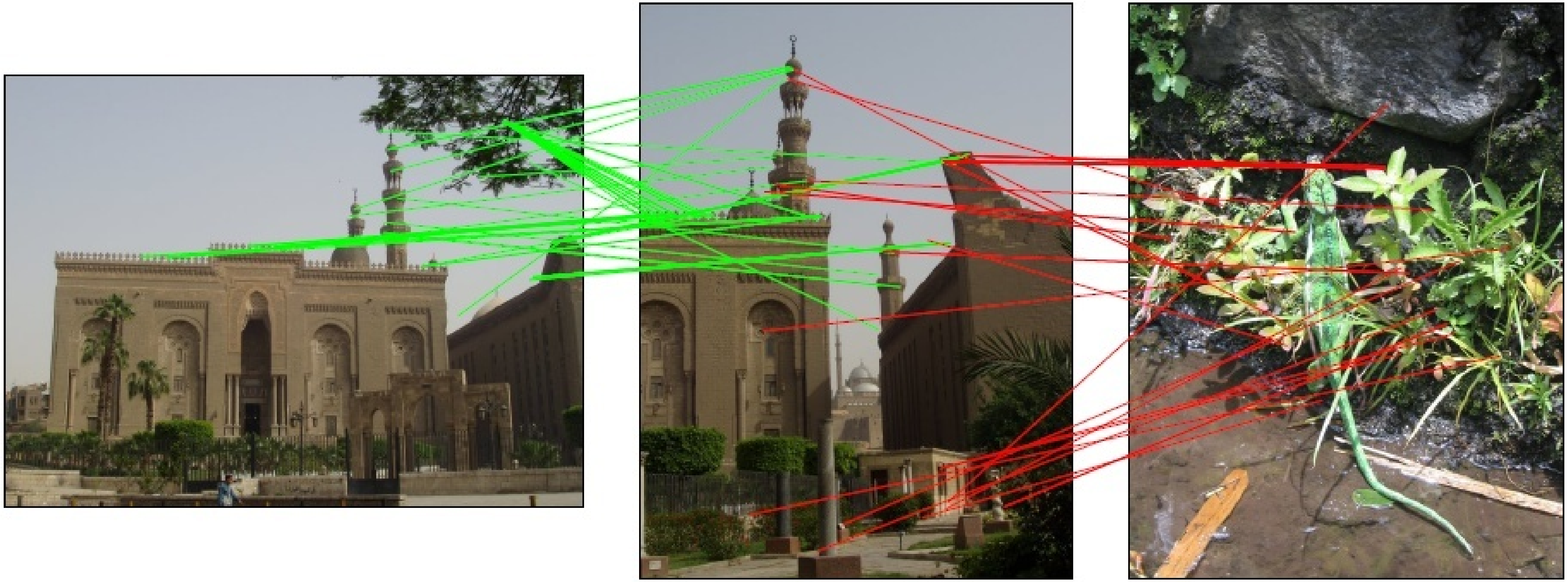


Many matches with the non-corresponding image!

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

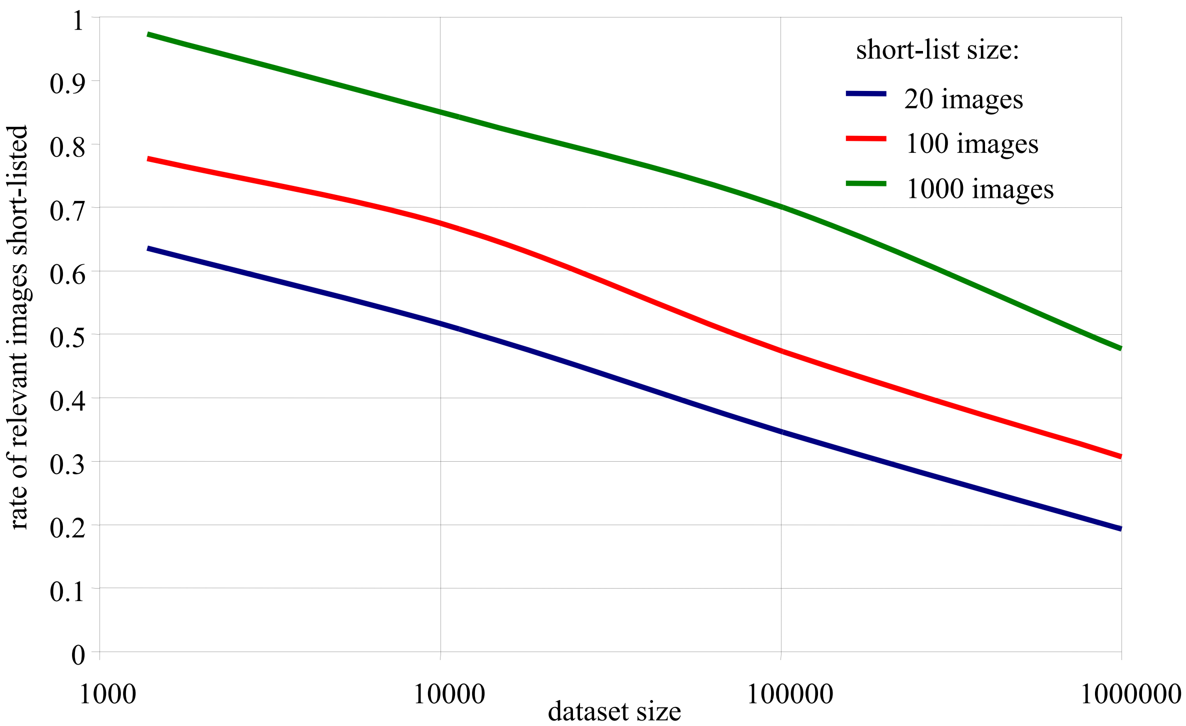
10x more matches with the corresponding image!

Indexing geometry of local features

- Re-ranking with geometric verification works very well
- but performed on a short-list only (typically, 1000 images)

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

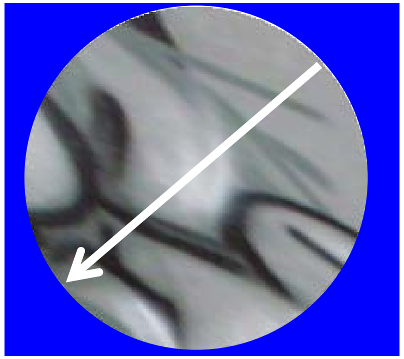
  → weak geometry in the image index



Weak geometry consistency

- Weak geometric information used for **all** images (not only the short-list)

- Each invariant interest region detection has a scale and rotation angle associated, here characteristic scale and dominant gradient orientation

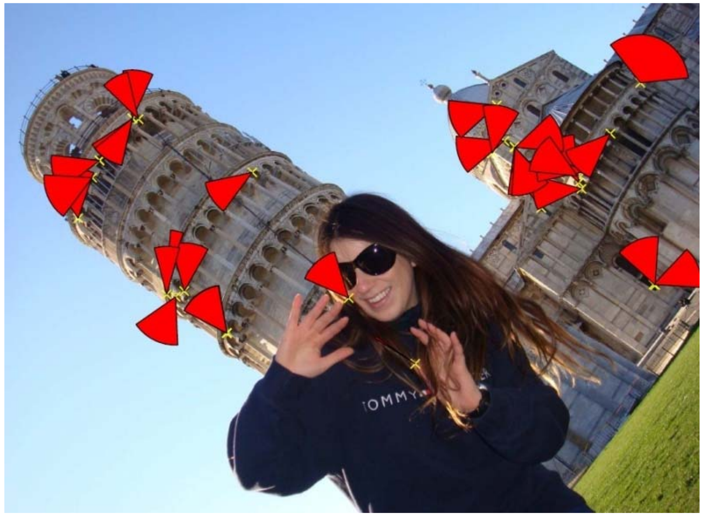


Scale change 2
Rotation angle ca. 20 degrees

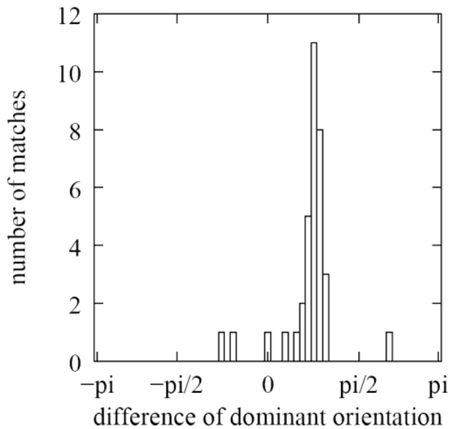
- Each matching pair results in a scale and angle difference

- For the global image scale and rotation changes are roughly consistent

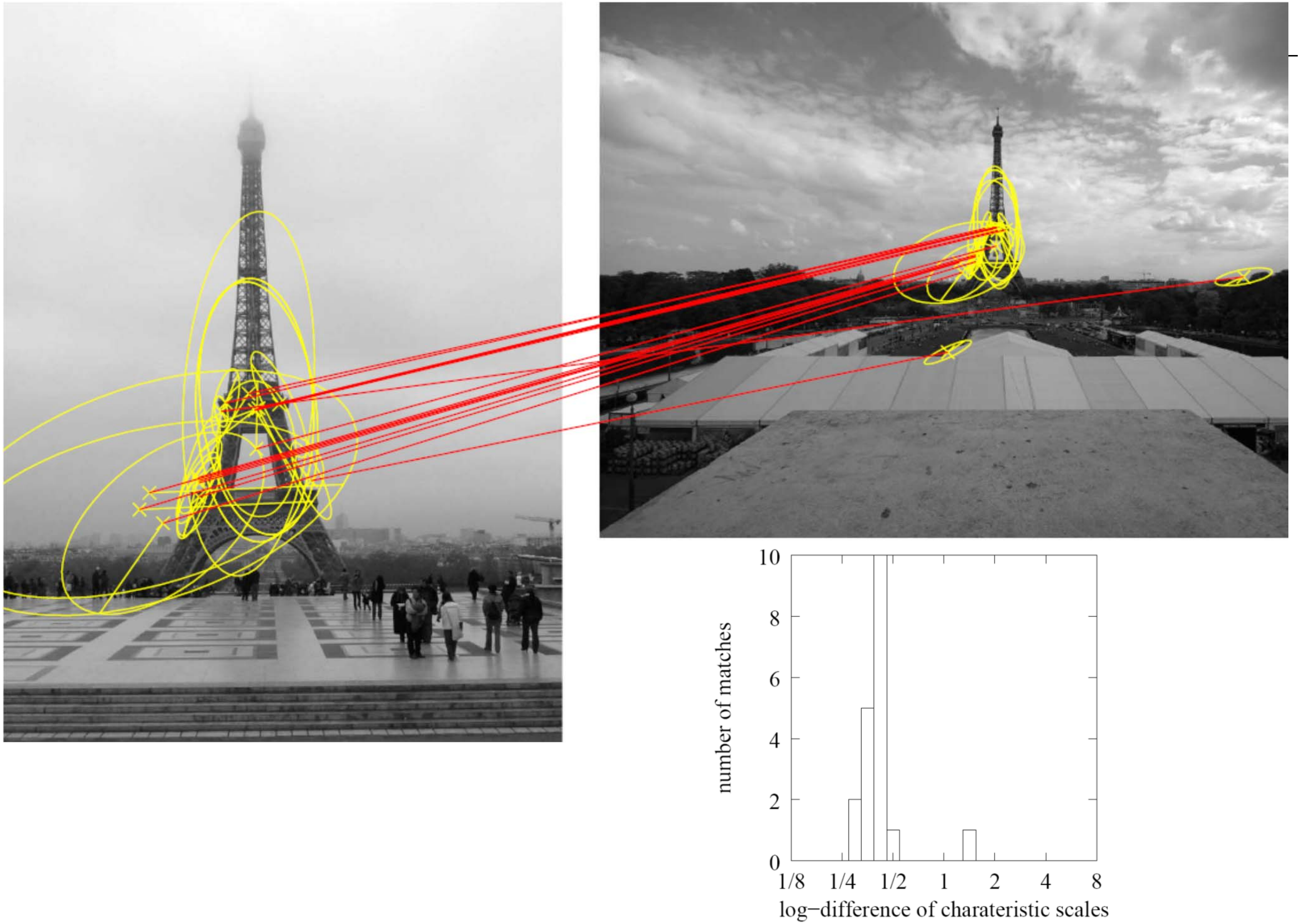
WGC: orientation consistency



Max = rotation angle between images



WGC: scale consistency



Weak geometry consistency

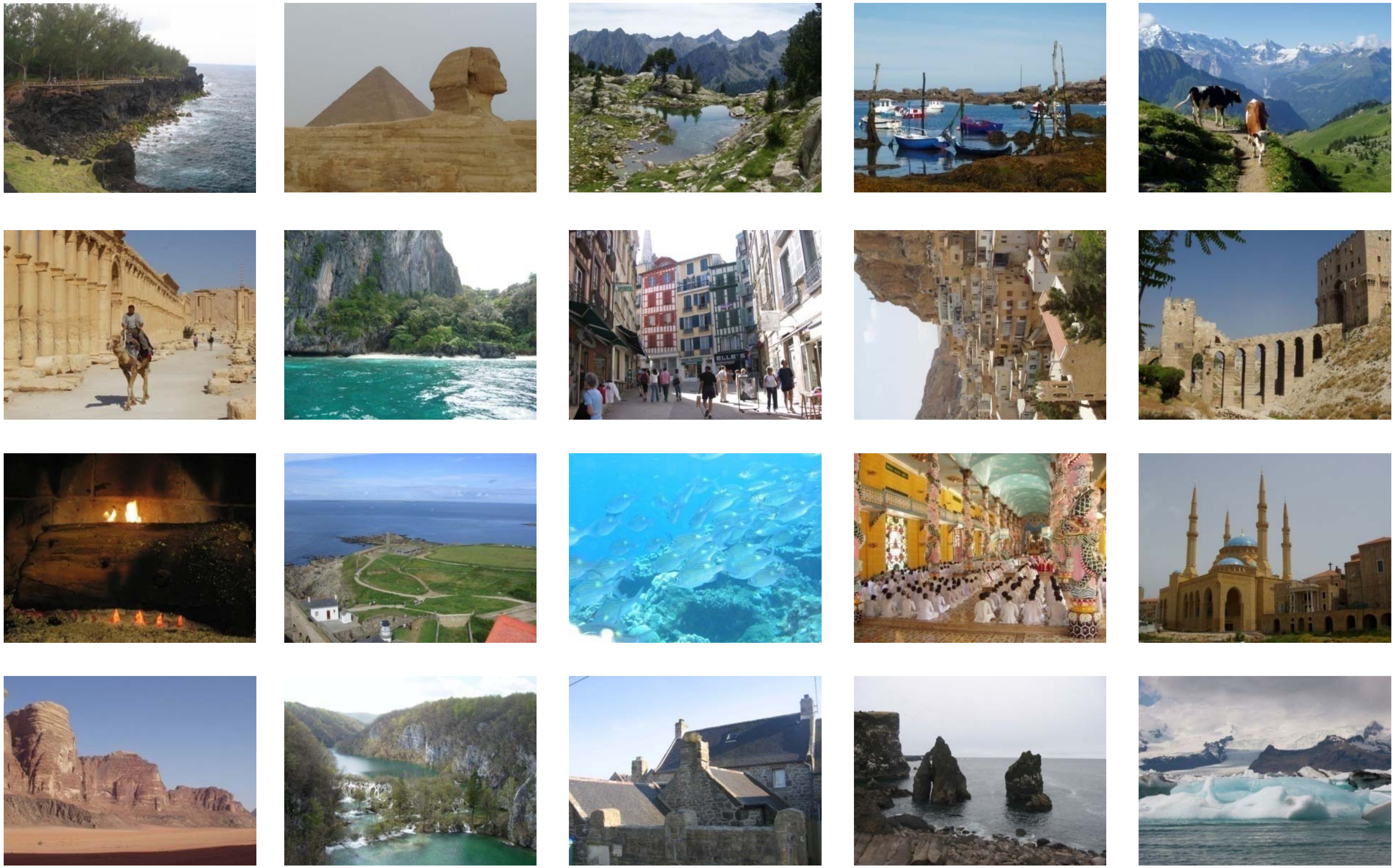
- Integration of the geometric verification into the BOF
  - votes for an image in two quantized subspaces, i.e. for angle & scale
  - these subspace are shown to be roughly independent
  - final score: filtering for each parameter (angle and scale)
- Only matches that do agree with the main difference of orientation and scale will be taken into account in the final score
- Re-ranking using full geometric transformation still adds information in a final stage

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


Query


Base 1


Base 2


Base 3


Base 4

Dataset : San Marco square


Query


Base 1


Base 2


Base 3


Base 4


Base 5

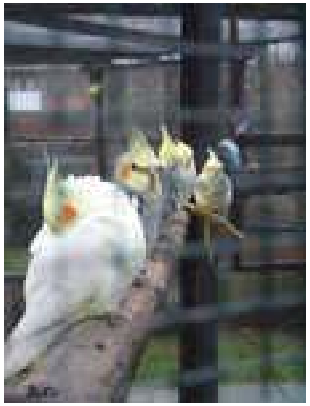

Base 6


Base 7


Base 8


Base 9

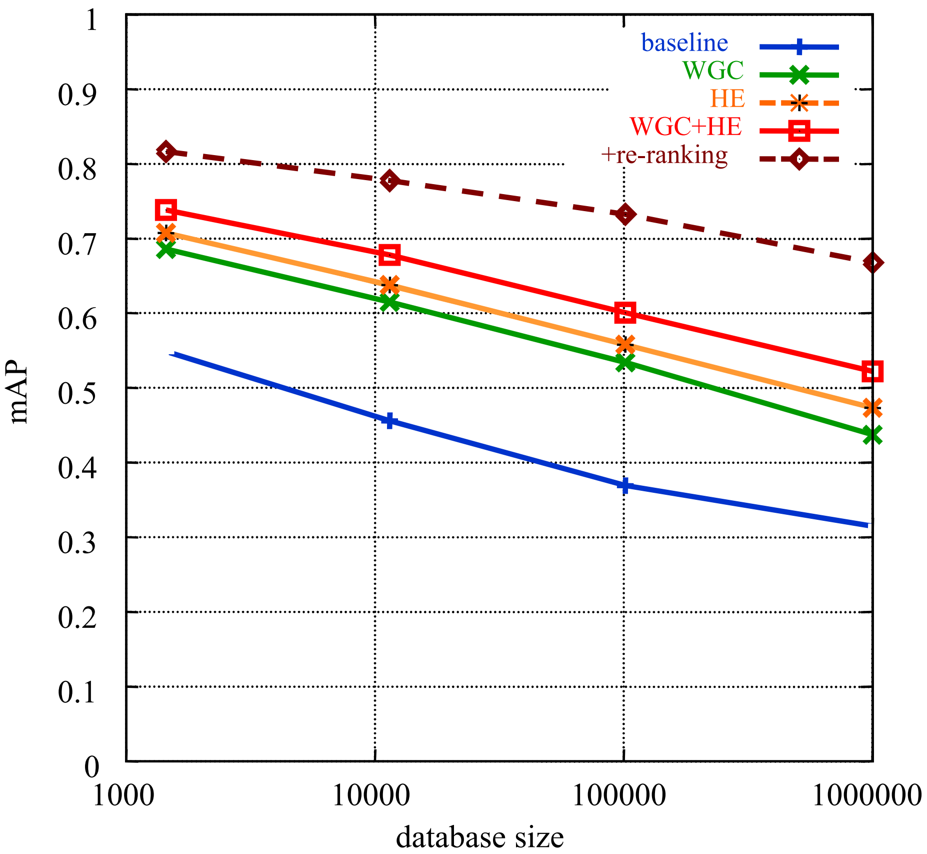
Example distractors - Flickr




Experimental evaluation

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



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

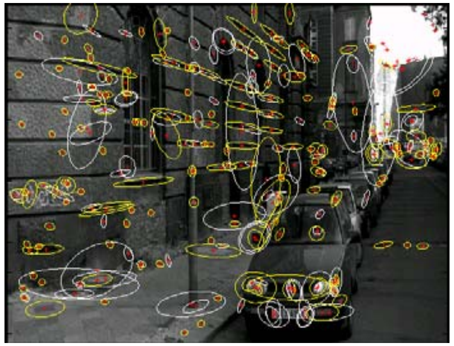
Towards large-scale image search

- 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

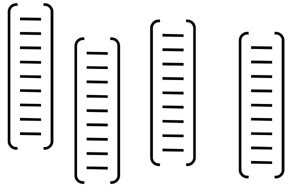
Strategy I: Efficient approximate NN search

Images

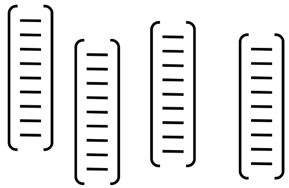
Local features



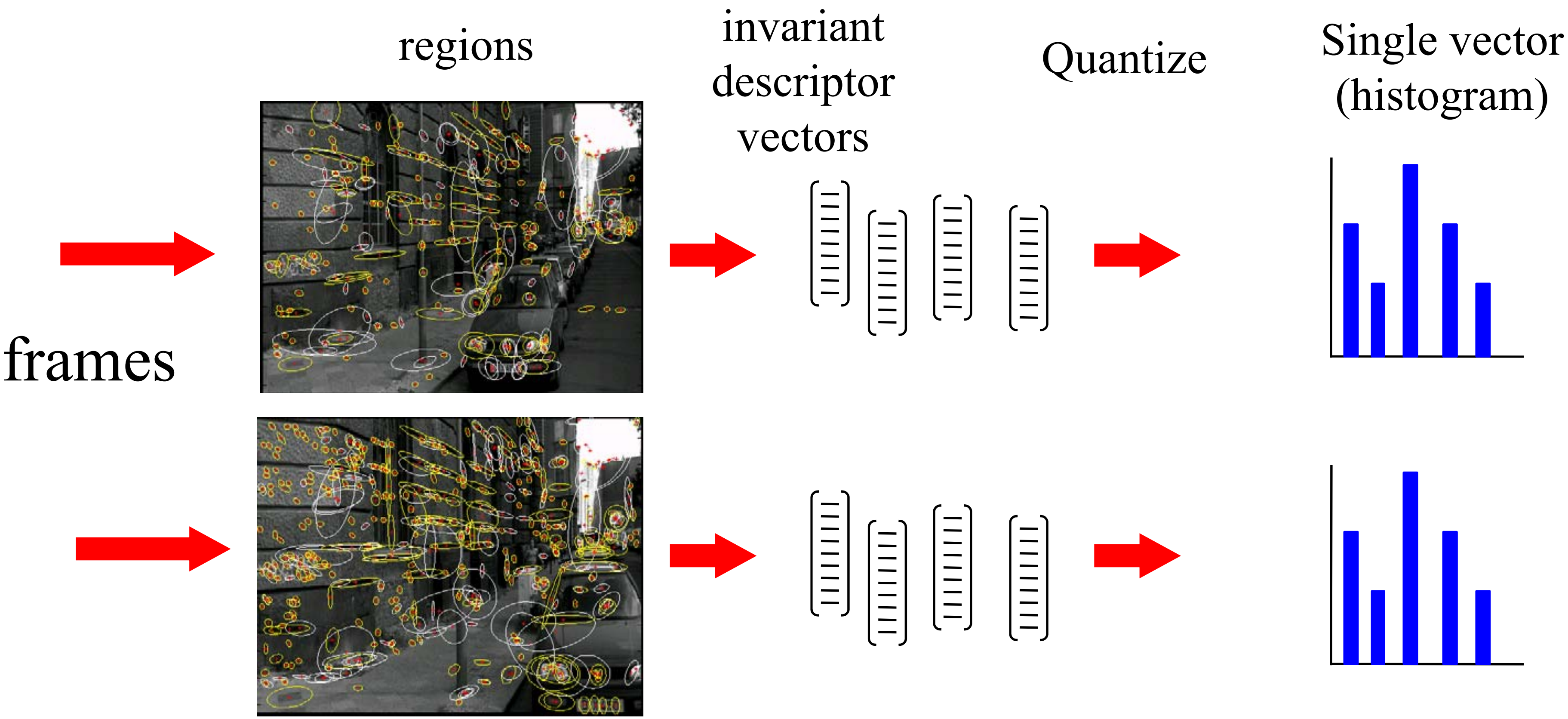
invariant descriptor vectors



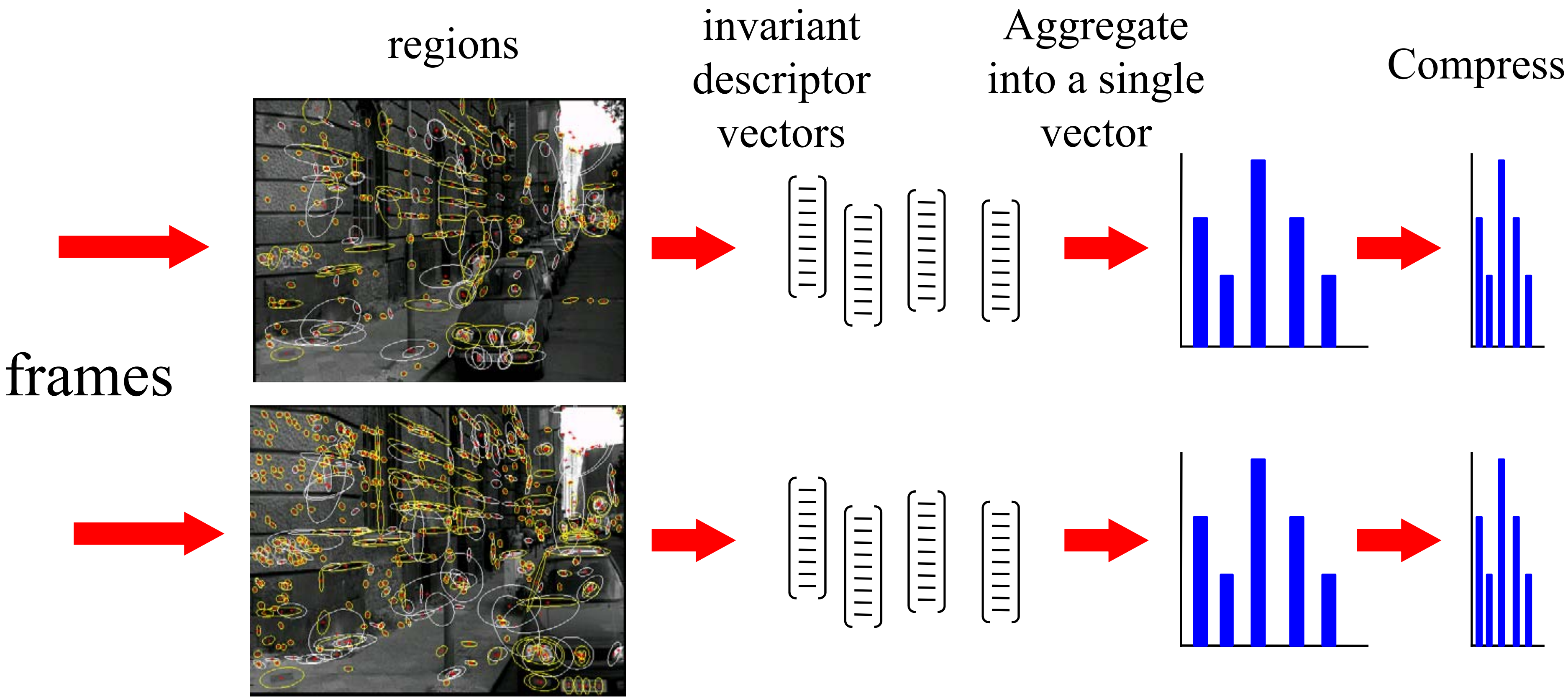
invariant descriptor vectors



Strategy II: Match histograms of visual words



Strategy II+: Match compressed vectors



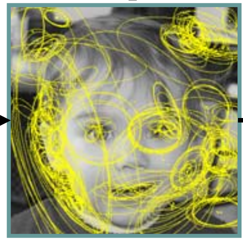
Very large scale image search

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

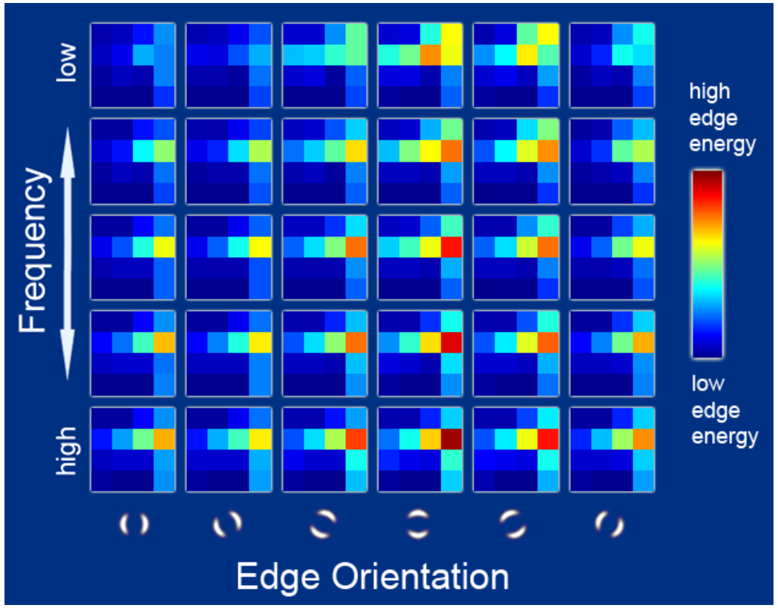
- Each image is represented by one vector (Bag-of-features, VLAD, Fisher, GIST)

- Vector compression to reduce storage requirements and search time

Global image descriptor with encoding

- GIST descriptors with Spectral Hashing [Weiss et al.'08]

- 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

GIST descriptor + spectral hashing

- The position of the descriptor in the image is encoded in the representation
  - very limited invariance to scale/rotation/crop

Torralba et al. (2003)

- Spectral hashing produces binary codes similar to spectral clusters

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*

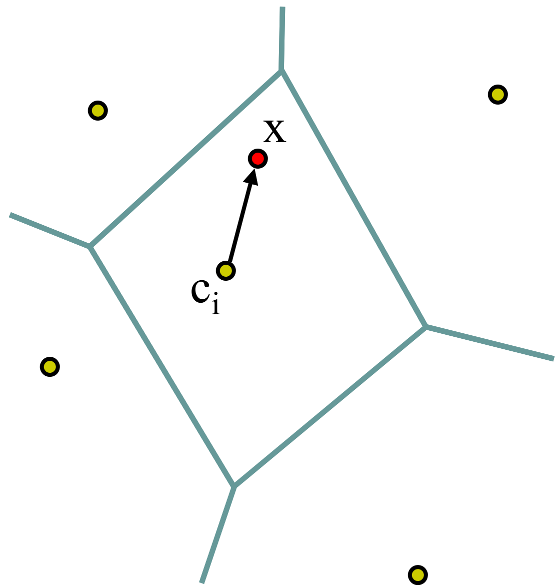
- For a given image
  - assign each descriptor to closest center c_i
  - accumulate (sum) descriptors per cell

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

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

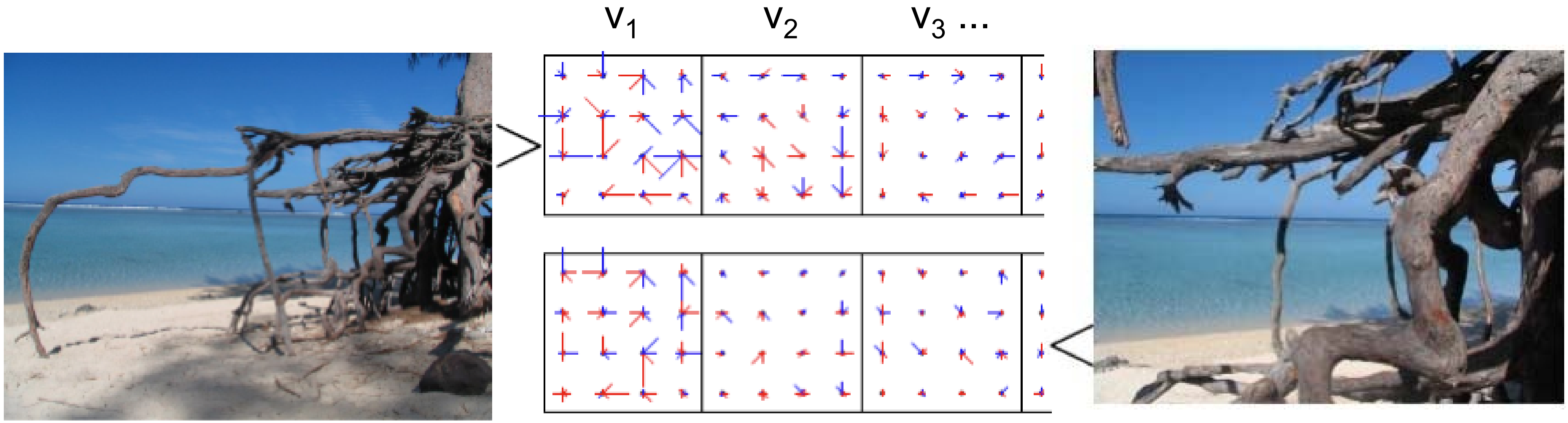
- The vector is square-root + L2-normalized

- Alternative: Fisher vector



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

VLADs for corresponding images

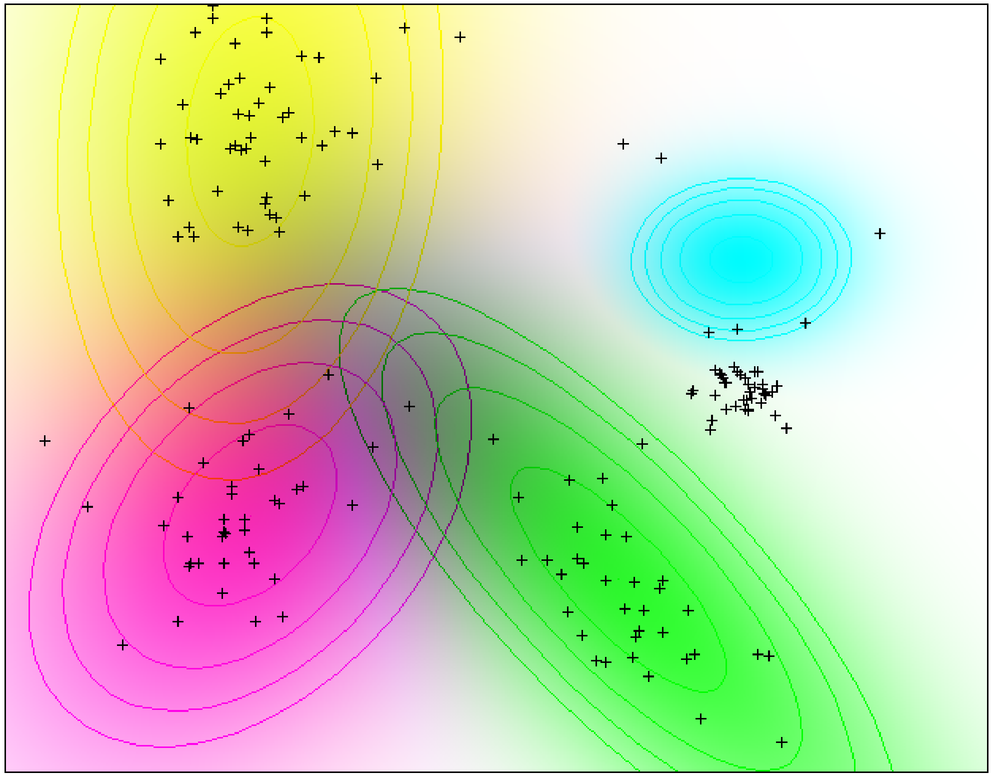


SIFT-like representation per centroid (+ components: blue, - components: red)

- 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

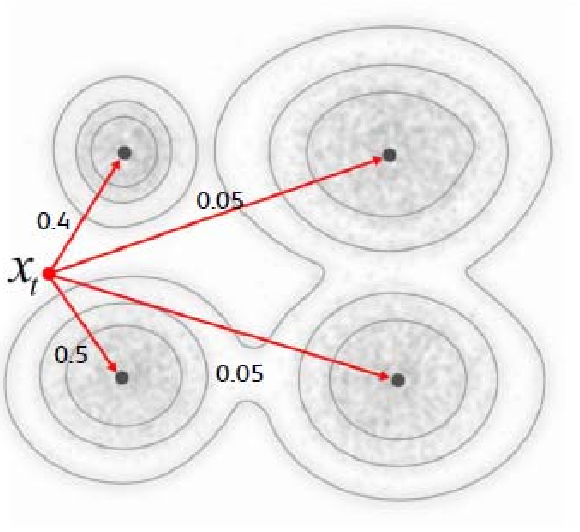
[Perronnin & Dance CVPR'07]

Fisher vector

FV formulas:

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

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



$\gamma_t(i)$ = 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)					
			$D' = D$	$\to D'$=2048	$\to D'$=512	$\to D'$=128	$\to D'$=64	$\to D'$=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

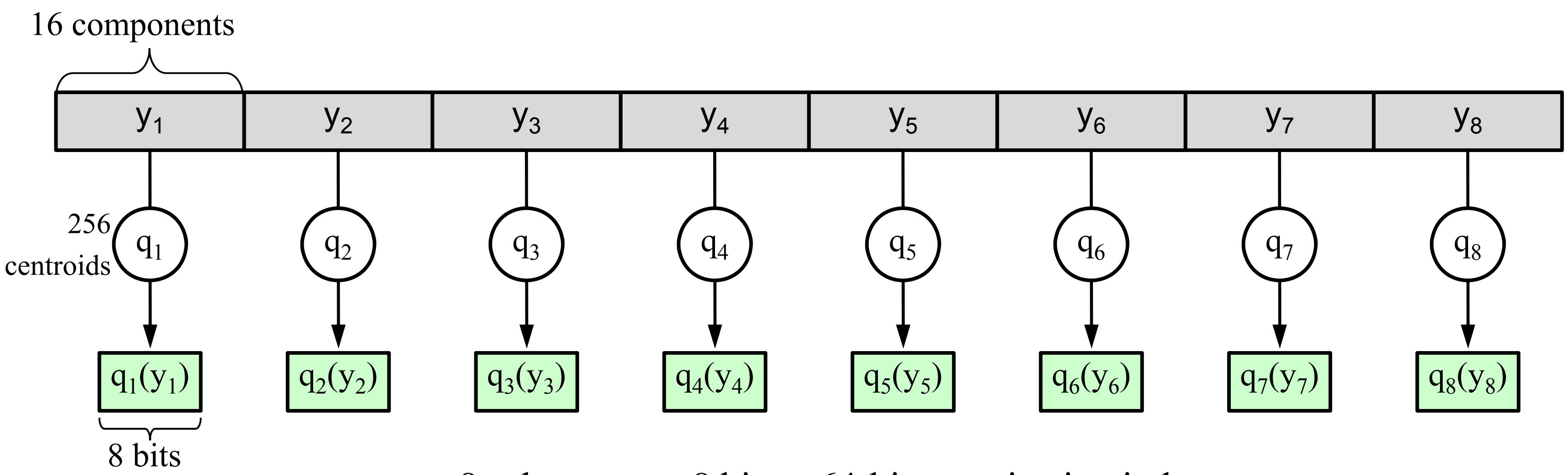
Image representation VLAD / Fisher	→	PCA + PQ codes	→	(Non) – exhaustive search

Product quantization for nearest neighbor search

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

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

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



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

[Jegou, Douze, Schmid, PAMI'11]

Conclusion

- Excellent search accuracy and speed in 10 million of images and more
- Each image is represented by very few bytes (20 – 40 bytes)
- Tested on up to 220 million video frames
  - extrapolation for 1 billion images: 20GB RAM, query time < 1s on 8 cores
- On-line available: Matlab source code for product quantizer
- Extension to video & more “semantic” search