# Efficient visual search of local features

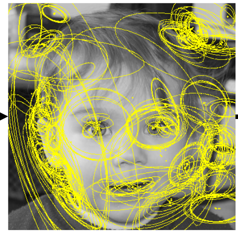
Cordelia Schmid

# 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**

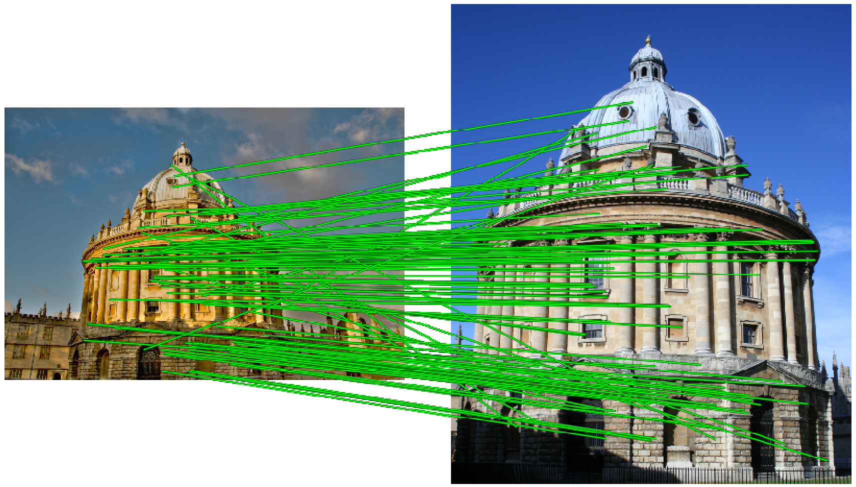
[Chum & al. 2007]

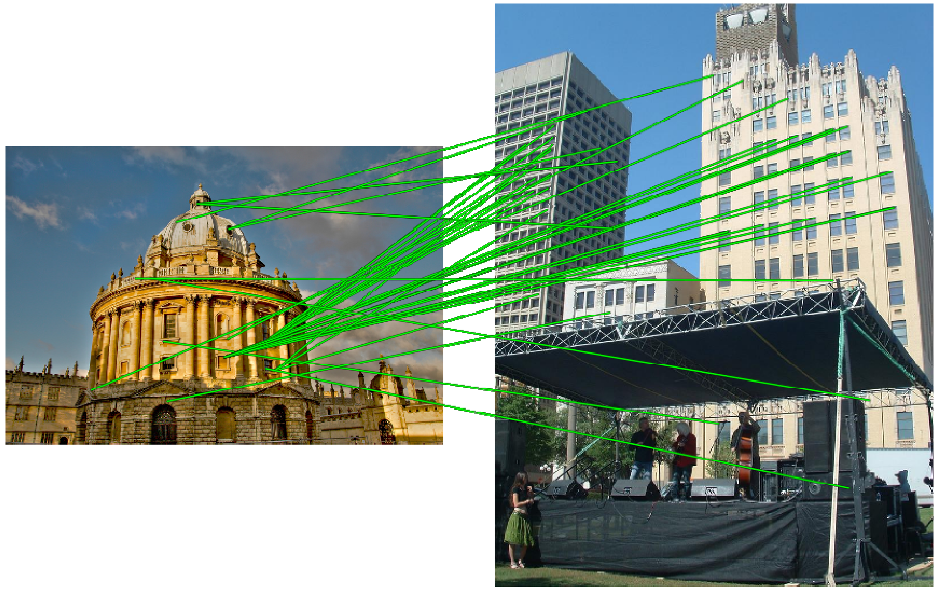
***Re-ranked list***

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

# Geometric verification

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

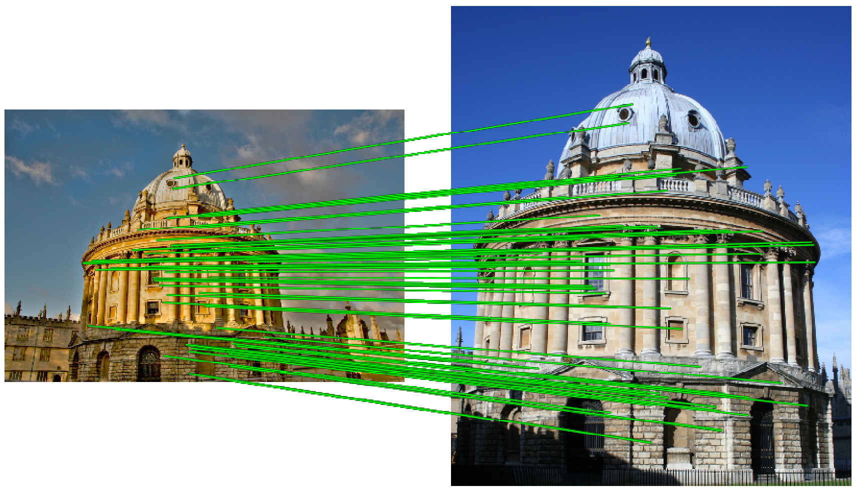




Both images have many matches – which is correct?

# Geometric verification

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

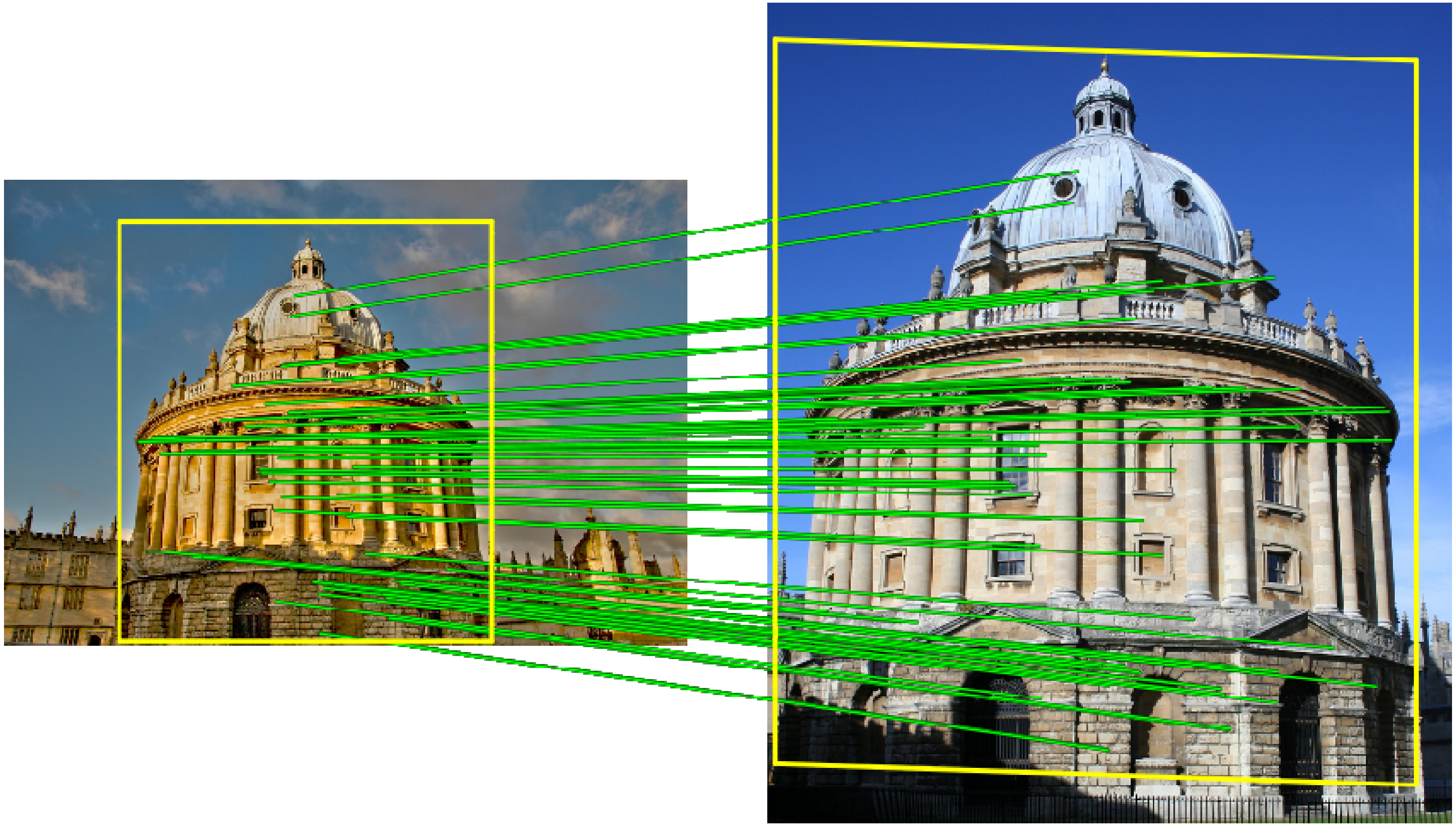


Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**

# Geometric verification

Gives **localization** of the object



# Geometric verification

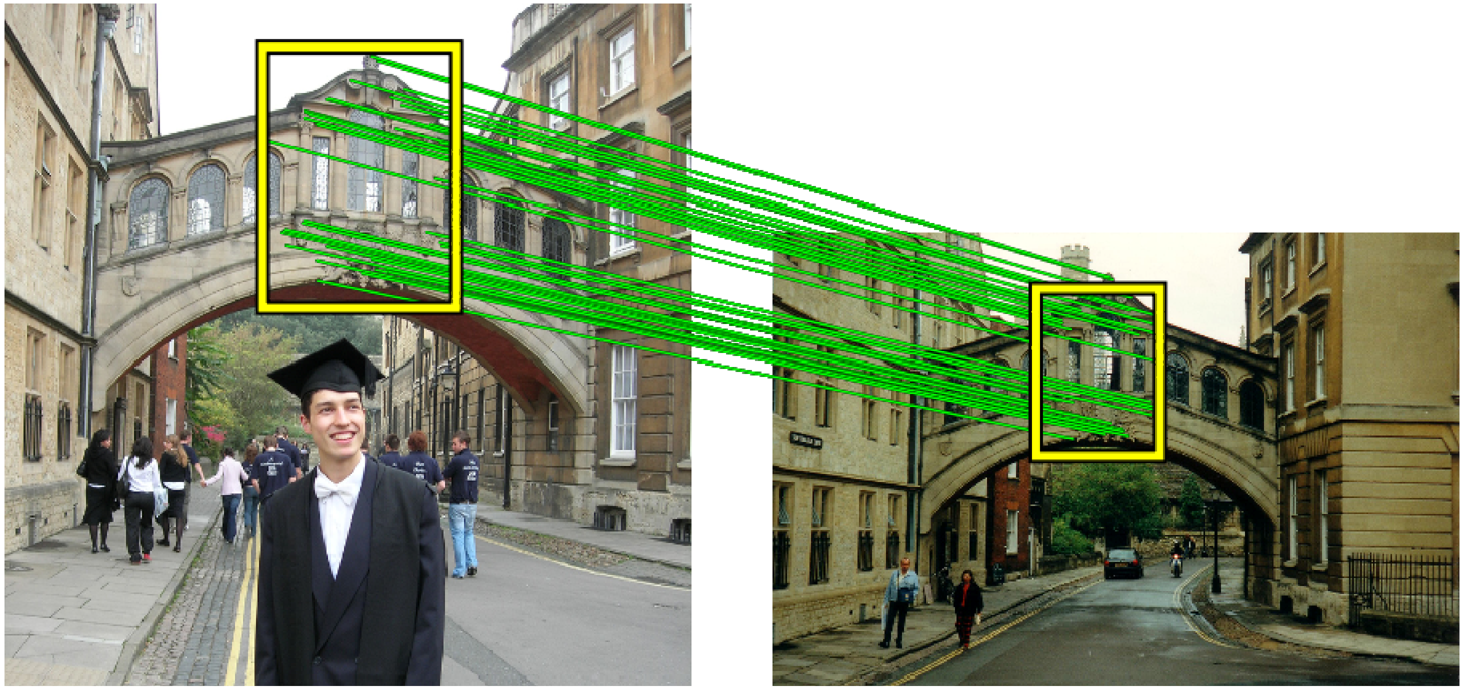
- Remove outliers, matches contain a high number of incorrect ones

- Estimate geometric transformation

- Robust strategies
  - RANSAC
  - Hough transform

# Example: estimating 2D affine transformation

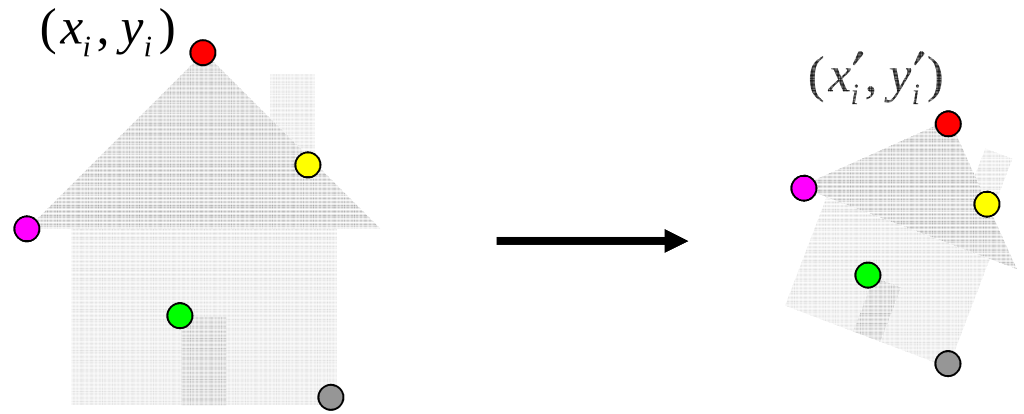
- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models



Matches consistent with an affine transformation

# 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

# Dealing with outliers

The set of putative matches may contain a high percentage (e.g. 90%) of outliers

How do we fit a geometric transformation to a small subset of all possible matches?

Possible strategies:

- RANSAC
- Hough transform

# Strategy 1: RANSAC

RANSAC loop (Fischler & Bolles, 1981):

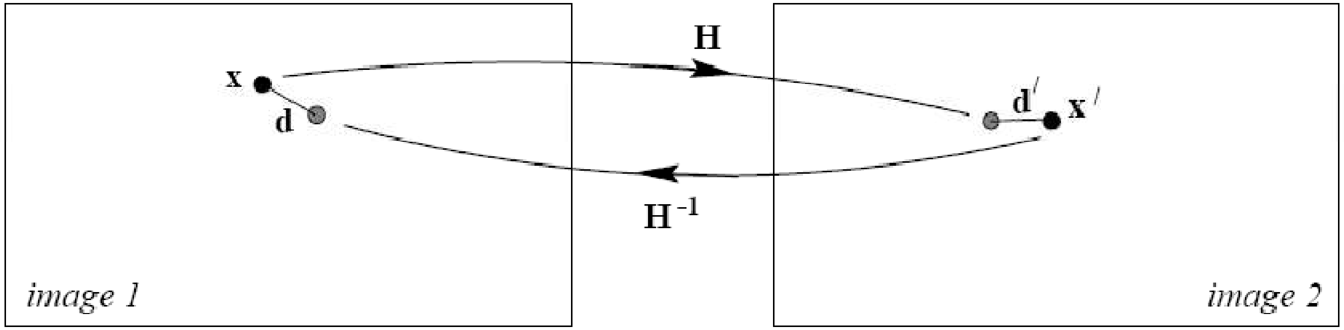
- Randomly select a *seed group* of matches
- Compute transformation from seed group
- Find *inliers* to this transformation
- If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
- Keep the transformation with the largest number of inliers

# Algorithm summary – RANSAC robust estimation of 2D affine transformation

Repeat

1. Select 3 point to point correspondences
2. Compute H (2x2 matrix) + t (2x1) vector for translation
3. Measure support (number of inliers within threshold distance, 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 the (H,t) with the largest number of inliers

(Re-estimate (H,t) from all inliers)

# Strategy 2: Hough Transform

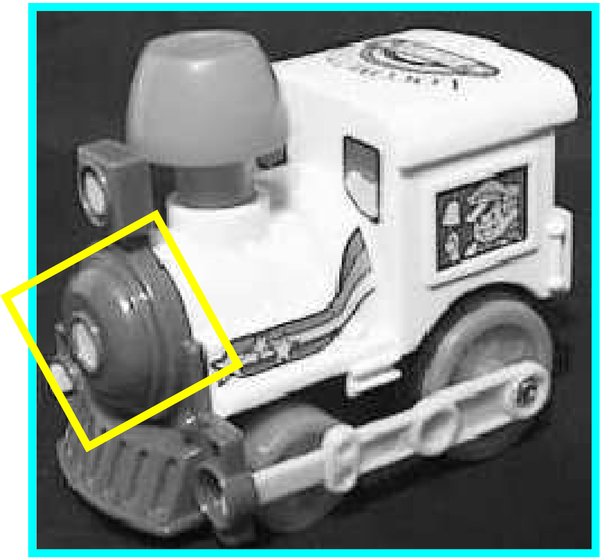
- Origin: Detection of straight lines in cluttered images
- Can be generalized to arbitrary shapes
- Can extract feature groupings from cluttered images in linear time.
- Illustrate on extracting sets of local features consistent with a similarity transformation

# 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

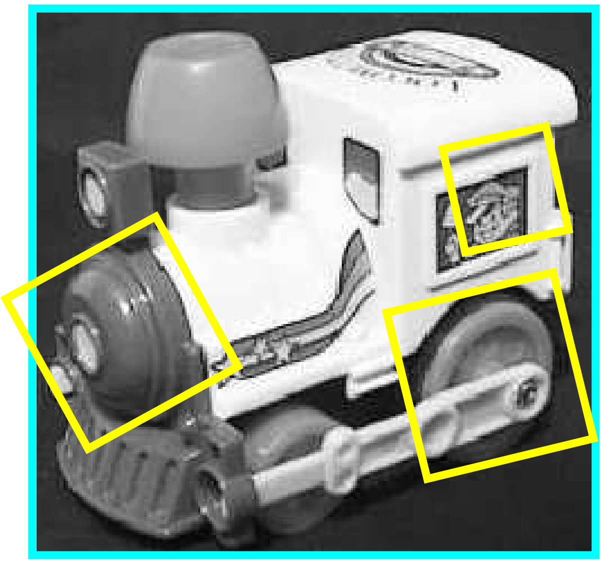

David G. Lowe. **"Distinctive image features from scale-invariant keypoints",** *IJCV* 60 (2), pp. 91-110, 2004.

# 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

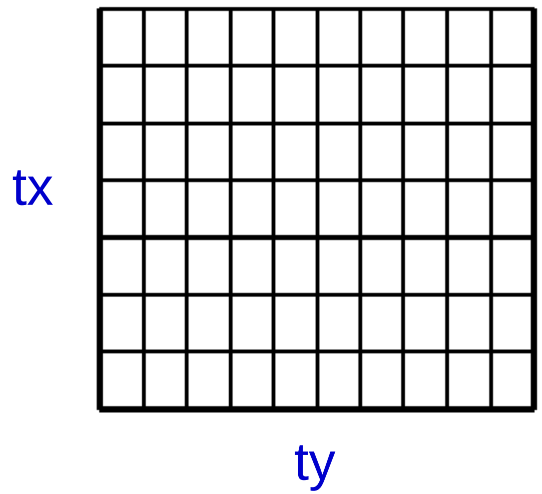



David G. Lowe. **“Distinctive image features from scale-invariant keypoints”,** *IJCV* 60 (2), pp. 91-110, 2004.

# 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

H: 4D-accumulator array
(only 2-d shown here)




- Correct matches will consistently vote for the same transformation while mismatches will spread votes

# Hough transform details (D. Lowe's system)

**Training phase:** For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)

**Test phase:** Let each match between a test and a model feature vote in a 4D Hough space

- Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
- Vote for two closest bins in each dimension

Find all bins with at least three votes and perform geometric verification

- Estimate least squares *affine* transformation
- Use stricter thresholds on transformation residual
- Search for additional features that agree with the alignment

# 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]

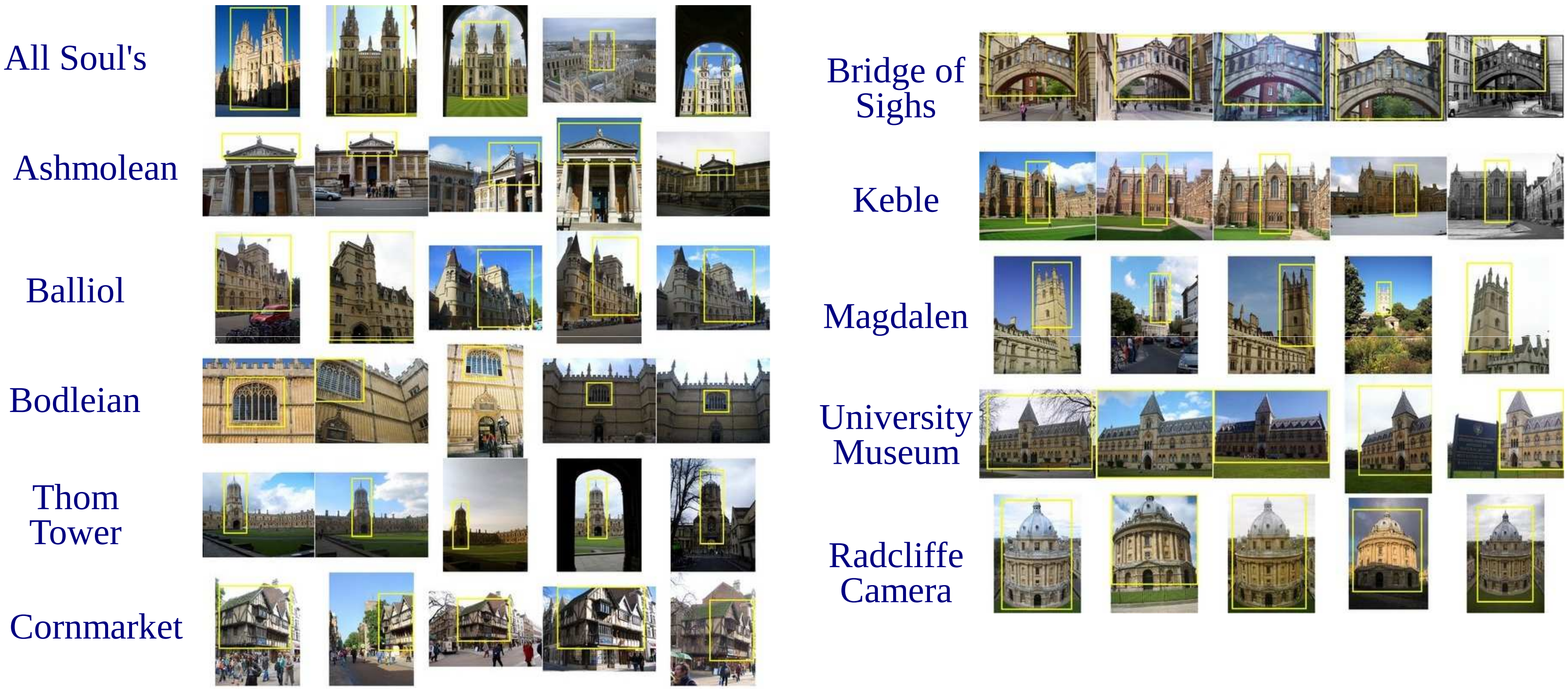
# Geometric verification – example

1. Query

2. Initial retrieval set (bag of words model)

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

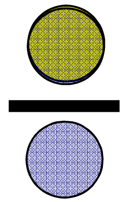
# Evaluation dataset: Oxford buildings



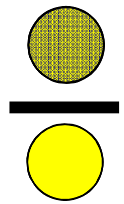

- Ground truth obtained for 11 landmarks
- Evaluate performance by mean Average Precision

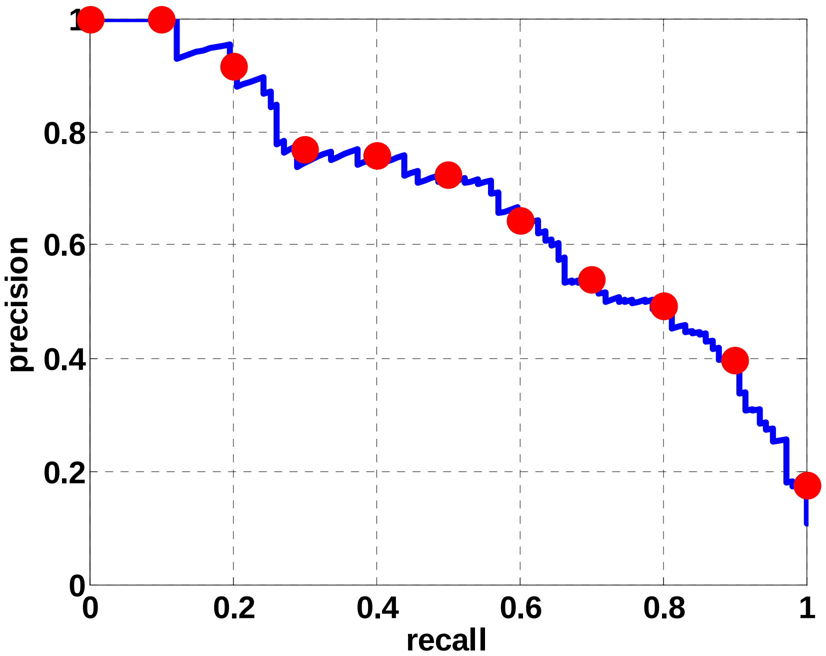
# Measuring retrieval performance: Precision - Recall

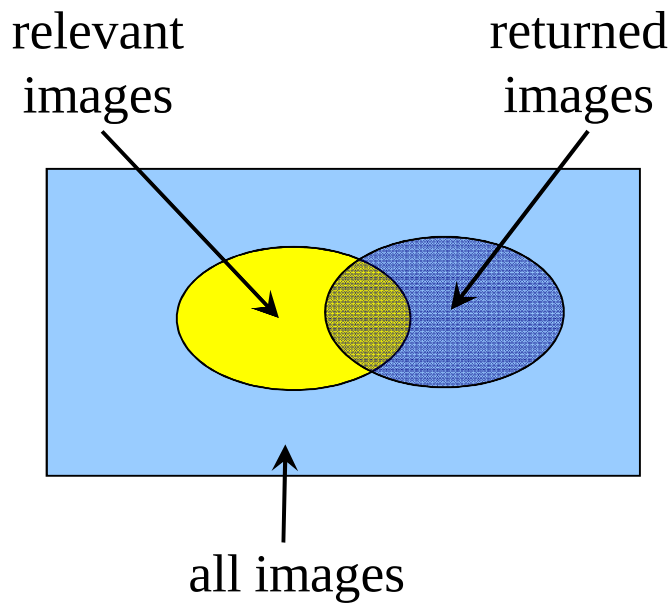
- Precision: % of returned images that are relevant



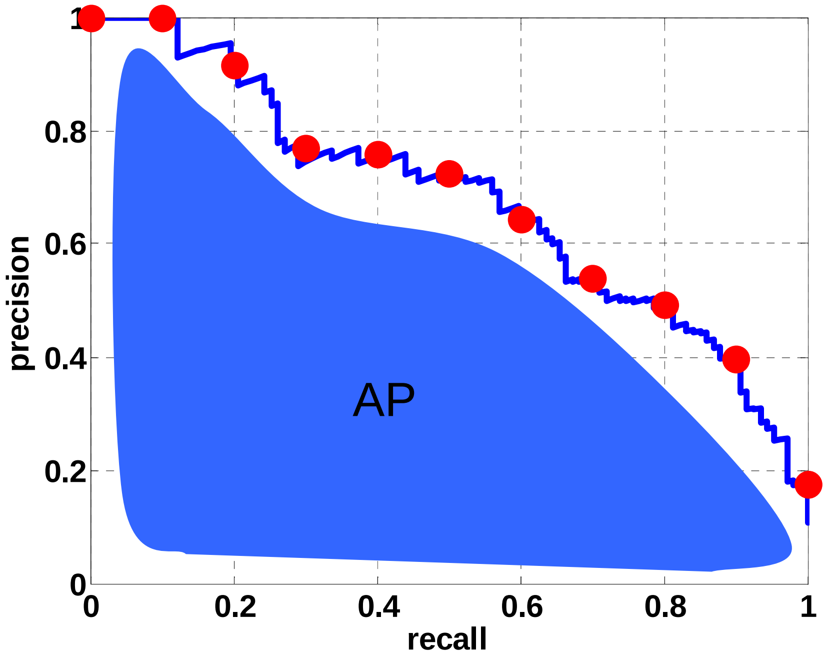
- Recall: % of relevant images that are returned





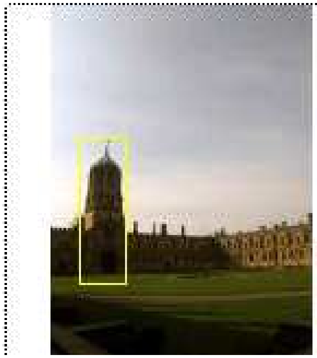



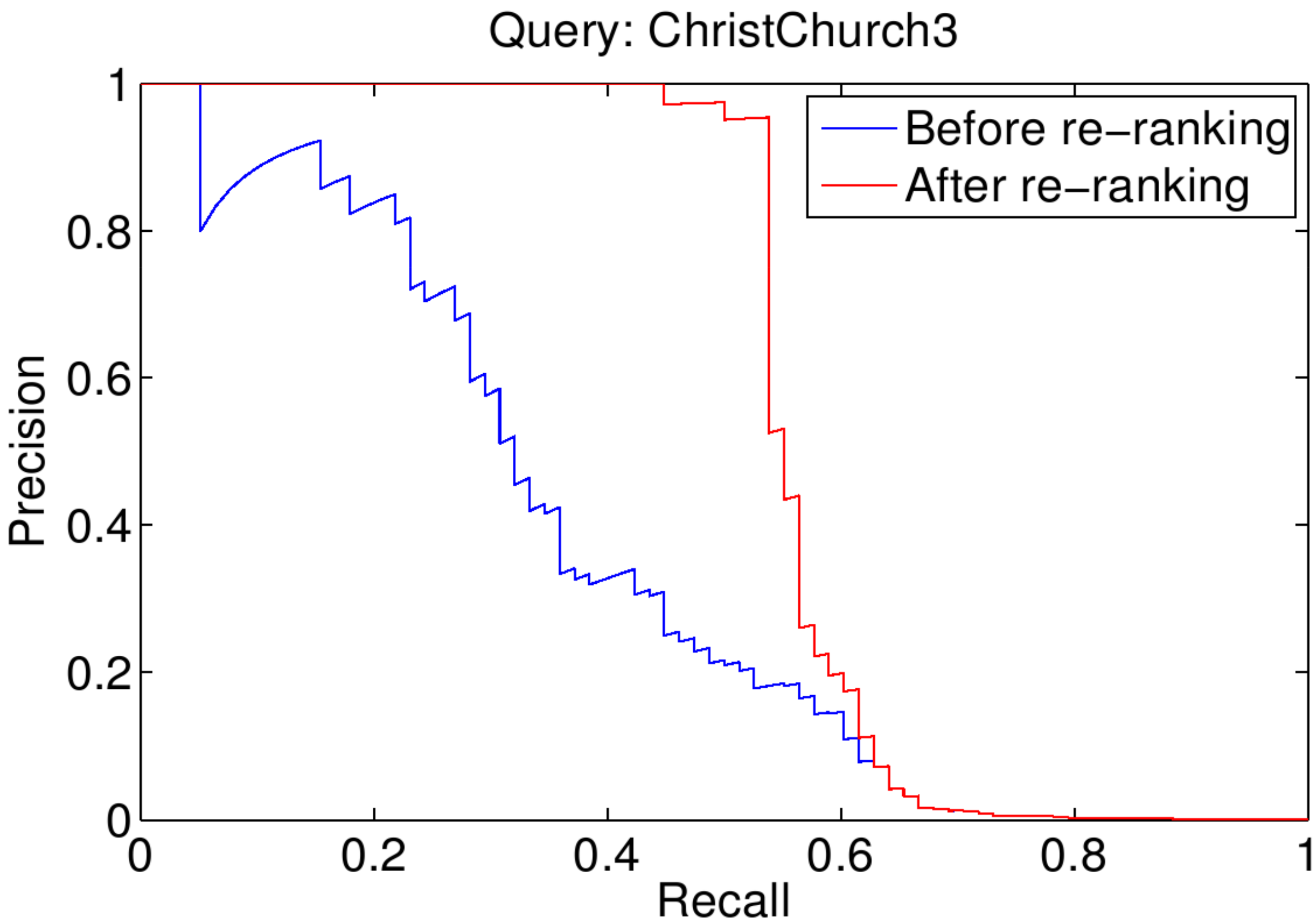
# Average Precision




- 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



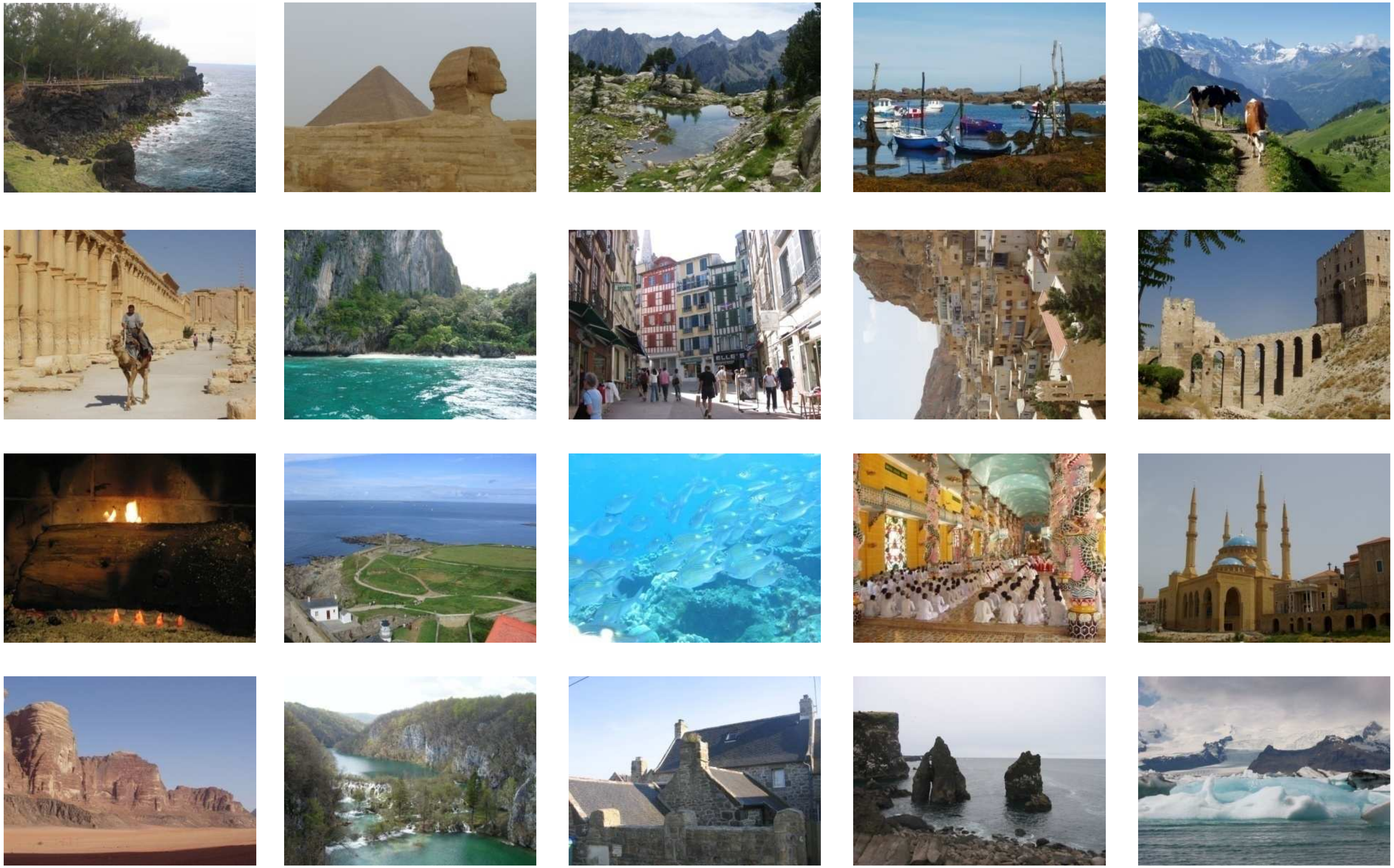

Query: ChristChurch3
Before re−ranking
After re−ranking
Precision
Recall
0
0.2
0.4
0.6
0.8
1

# 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

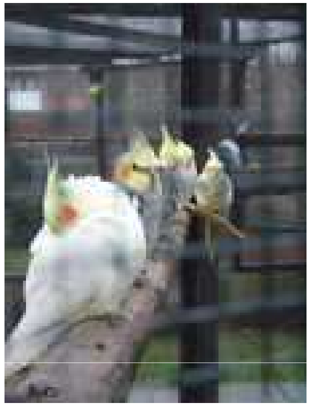








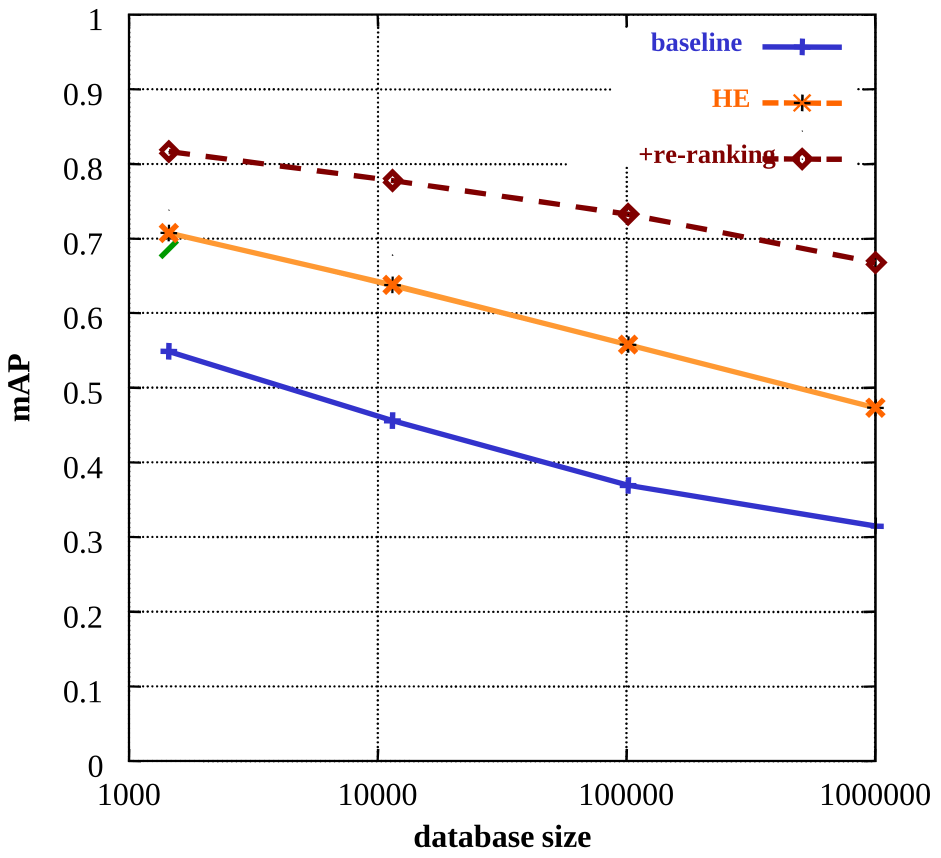

# Example distractors - Flickr




# Experimental evaluation

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



# Results – Venice Channel






Demo at http://bigimbaz.inrialpes.fr

# Towards larger databases?

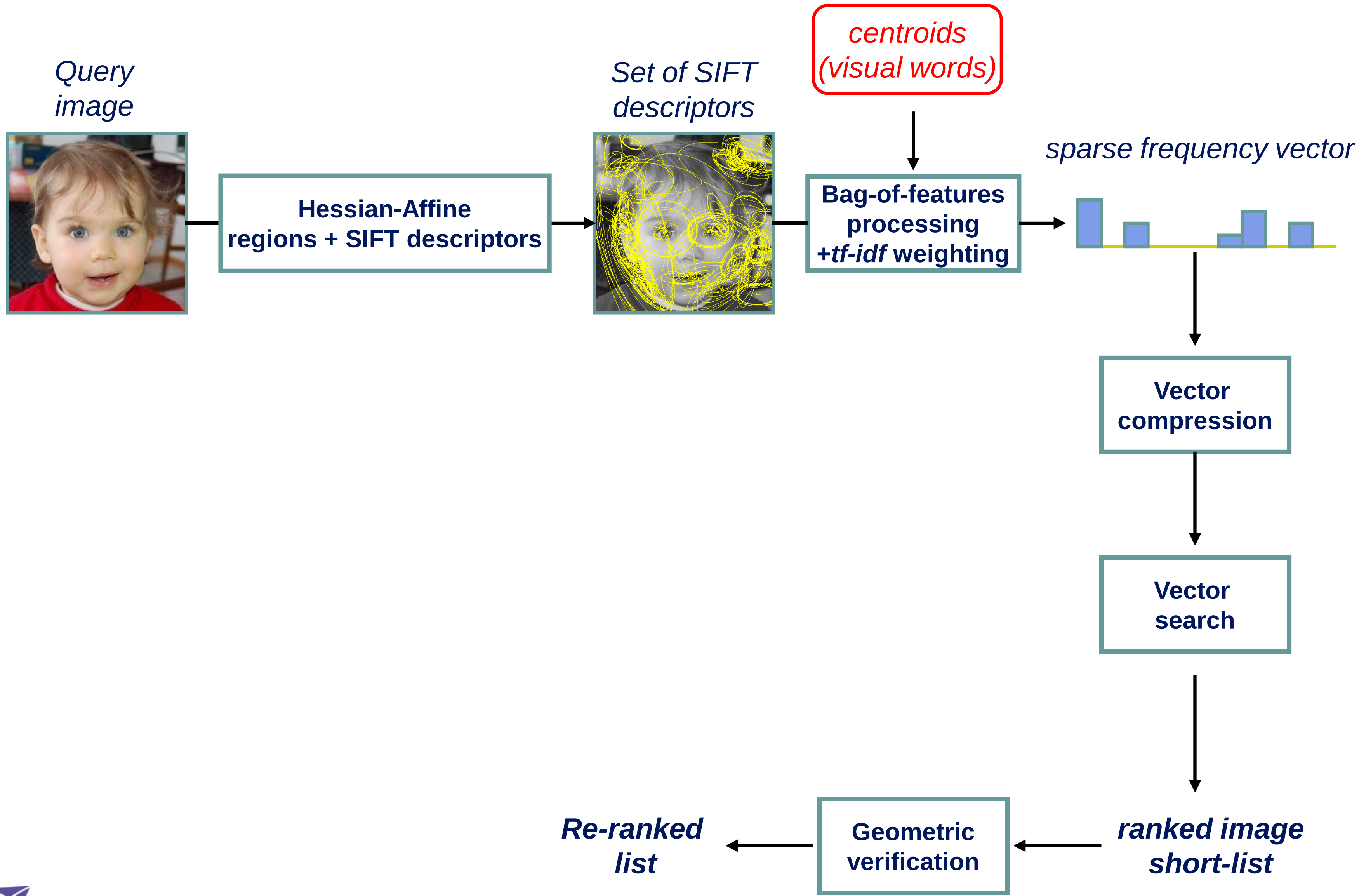
- BOF can handle up to ~10 M d'images
  - with a limited number of descriptors per image
  - 40 GB of RAM
  - search = 2 s

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

    $\longrightarrow$ search = 20 s, RAM = 400 GB

    $\longrightarrow$ not tractable!

# Recent approaches for very large scale indexing



## Related work on very large scale image search

- GIST descriptors with Spectral Hashing [Torralba et al. ‘08]
- Compressing the BoF representation (miniBof) [Jegou et al. ‘09]
- Aggregating local desc into a compact image representation [Jegou et al. ‘10]
- Efficient object category recognition using classemes [Torresani et al.’10]

INRIA