Computer Vision

Instance Recognition

Matching local features

Kristen Grauman

Matching local features

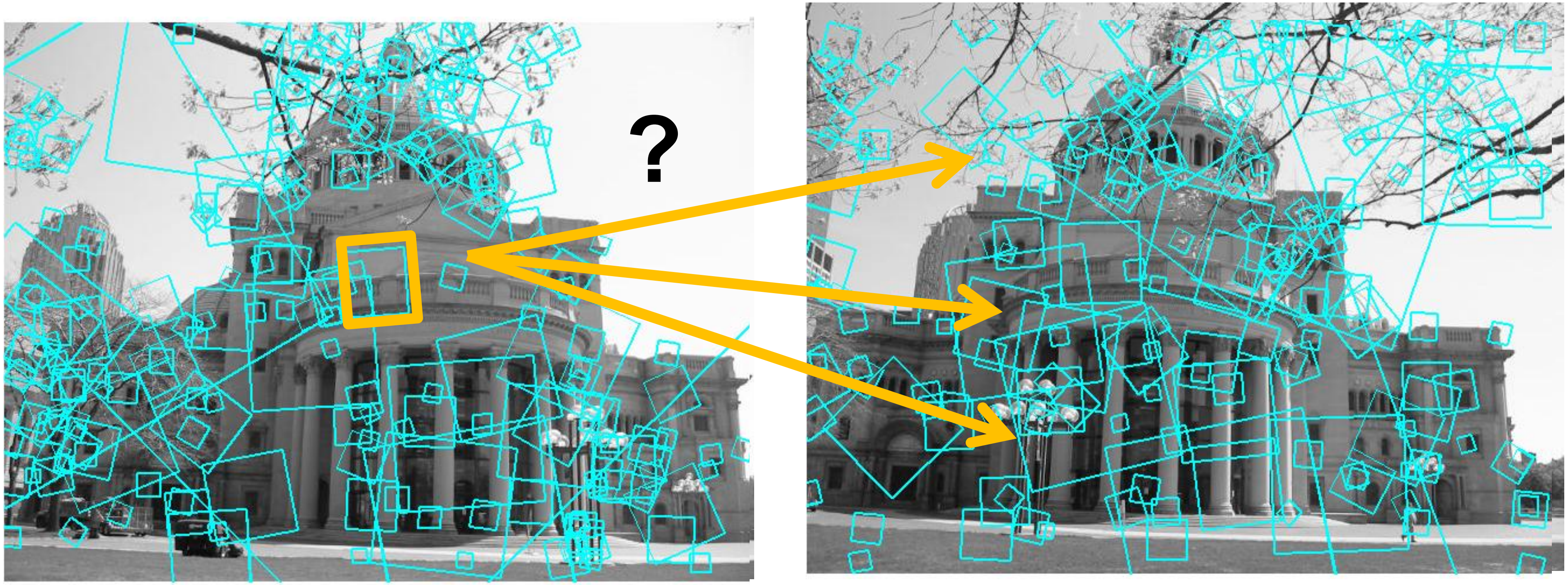


Image 1 Image 2

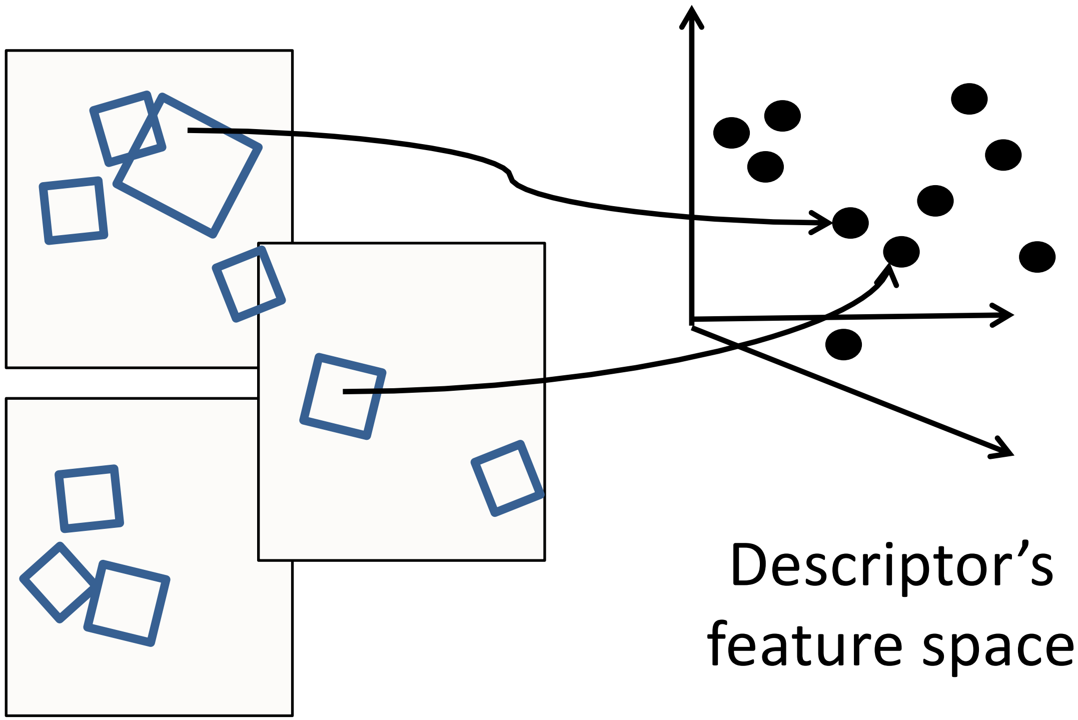
To generate candidate matches, find patches that have the most similar appearance (e.g., lowest SSD)

Simplest approach: compare them all, take the closest (or closest k, or within a thresholded distance)

Kristen Grauman

Indexing local features

- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)

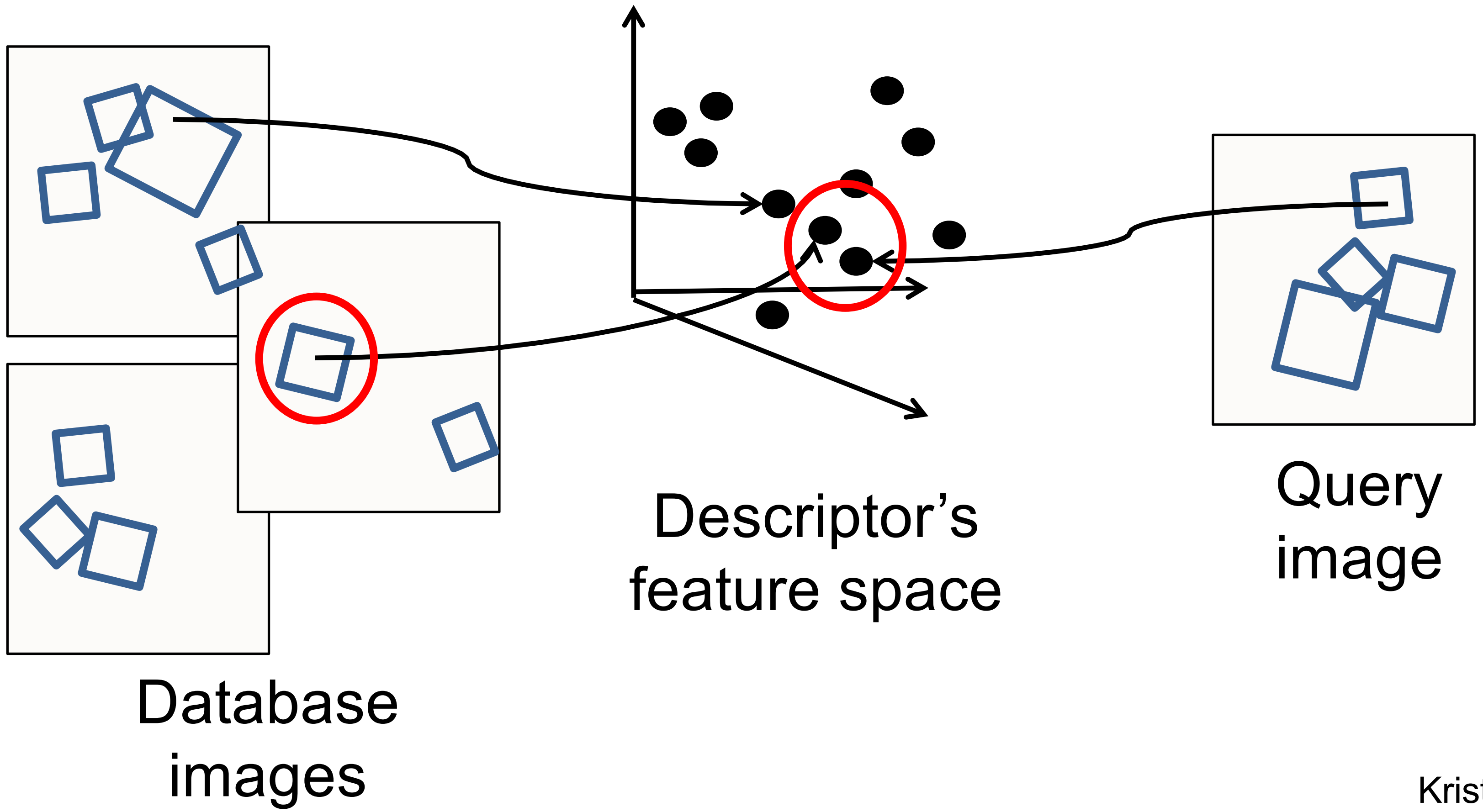


Descriptor's feature space

Kristen Grauman

Indexing local features

- When we see close points in feature space, we have similar descriptors, which indicates similar local content.



Kristen Grauman

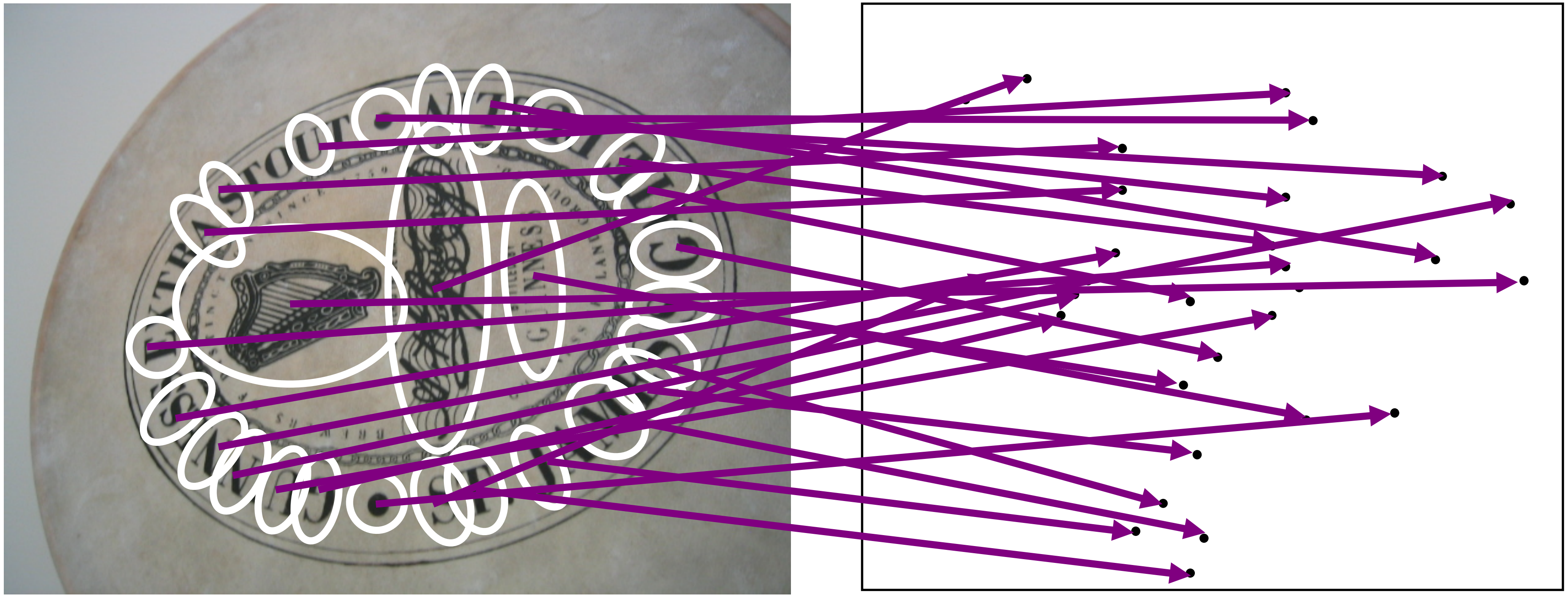
Indexing local features

- With potentially thousands of features per image, and hundreds to millions of images to search, how to efficiently find those that are relevant to a new image?

Kristen Grauman

Visual words: main idea

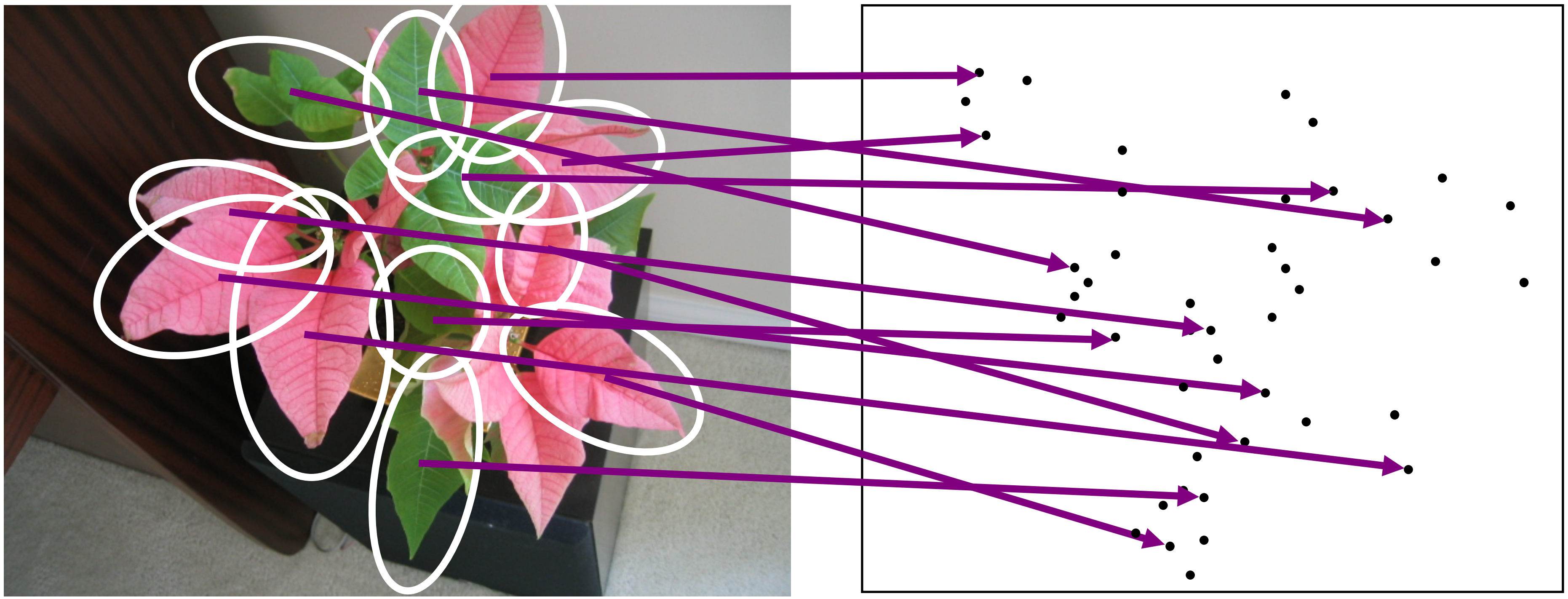
Extract some local features from a number of images …



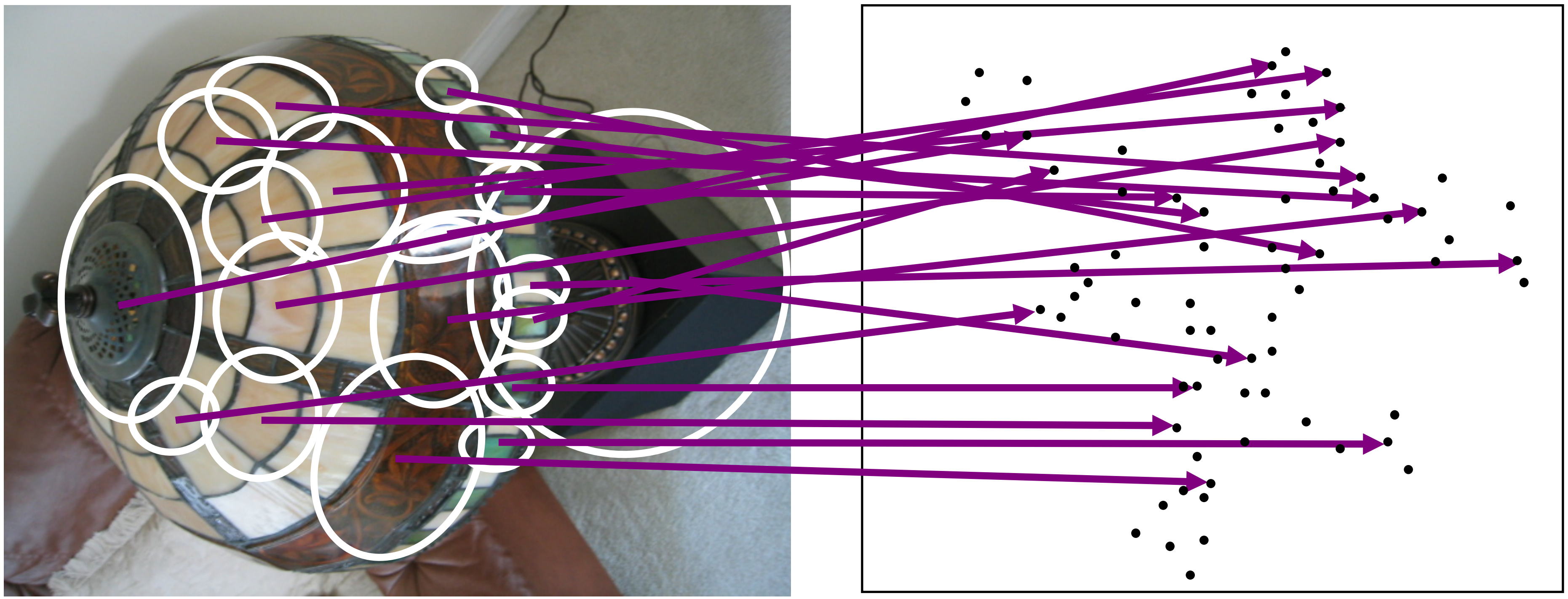
e.g., SIFT descriptor space: each point is 128-dimensional

Slide credit: D. Nister, CVPR 2006

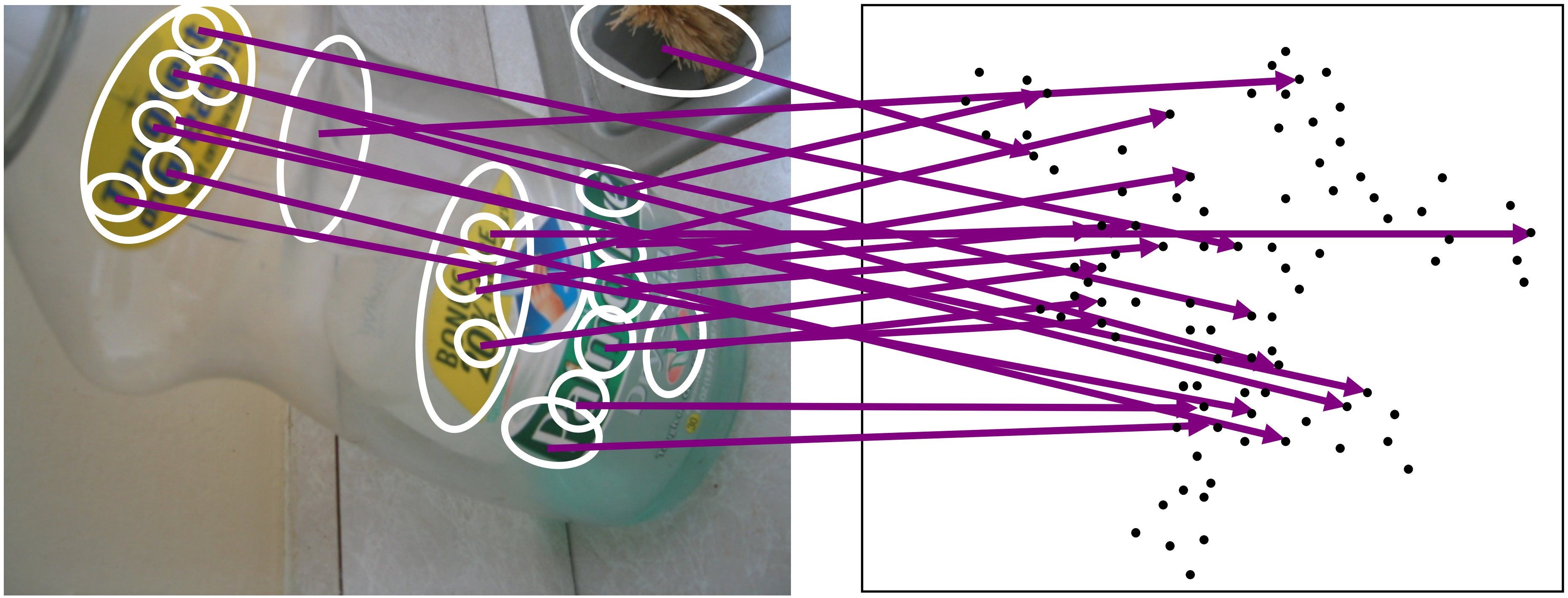
Visual words: main idea

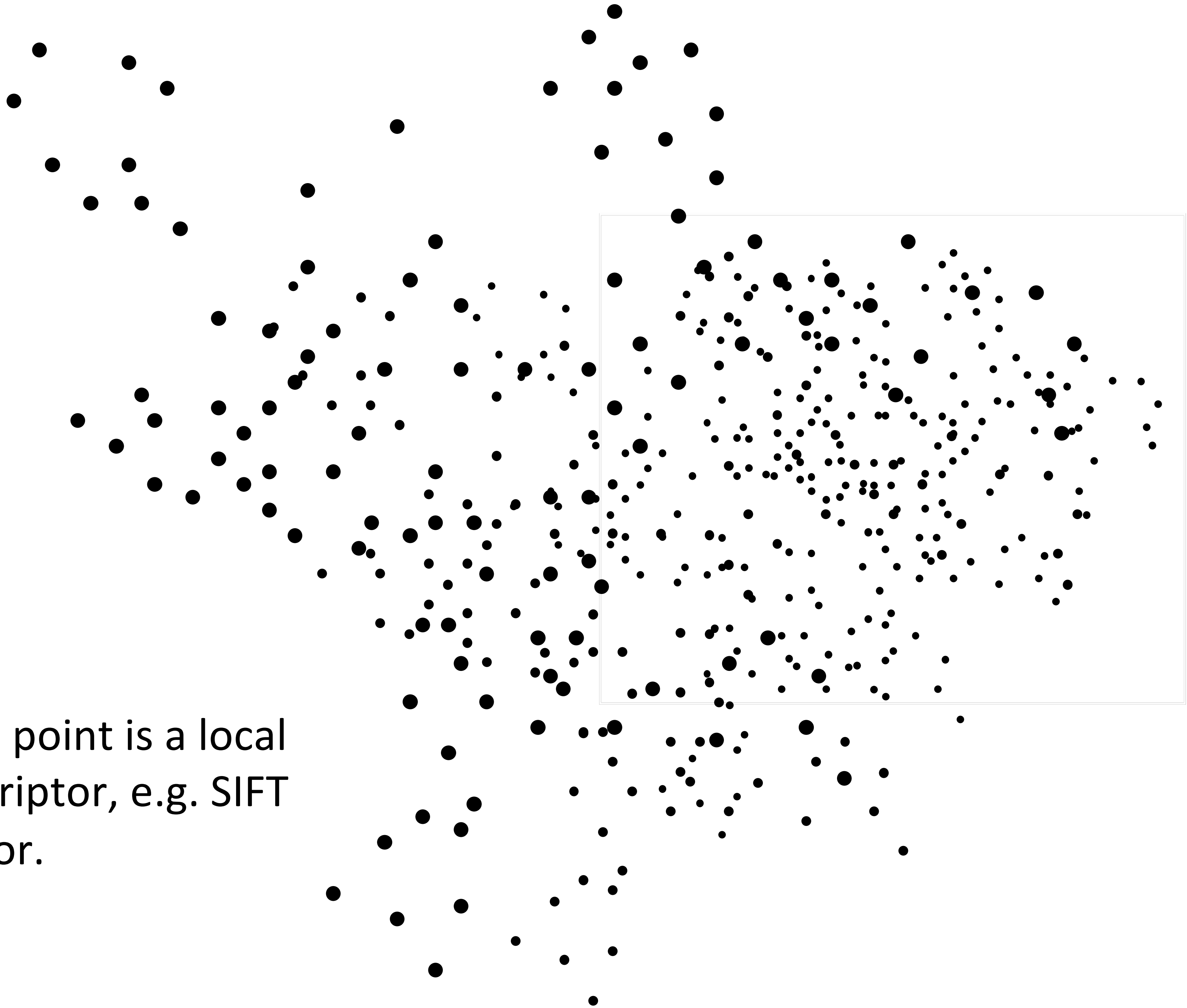


Visual words: main idea

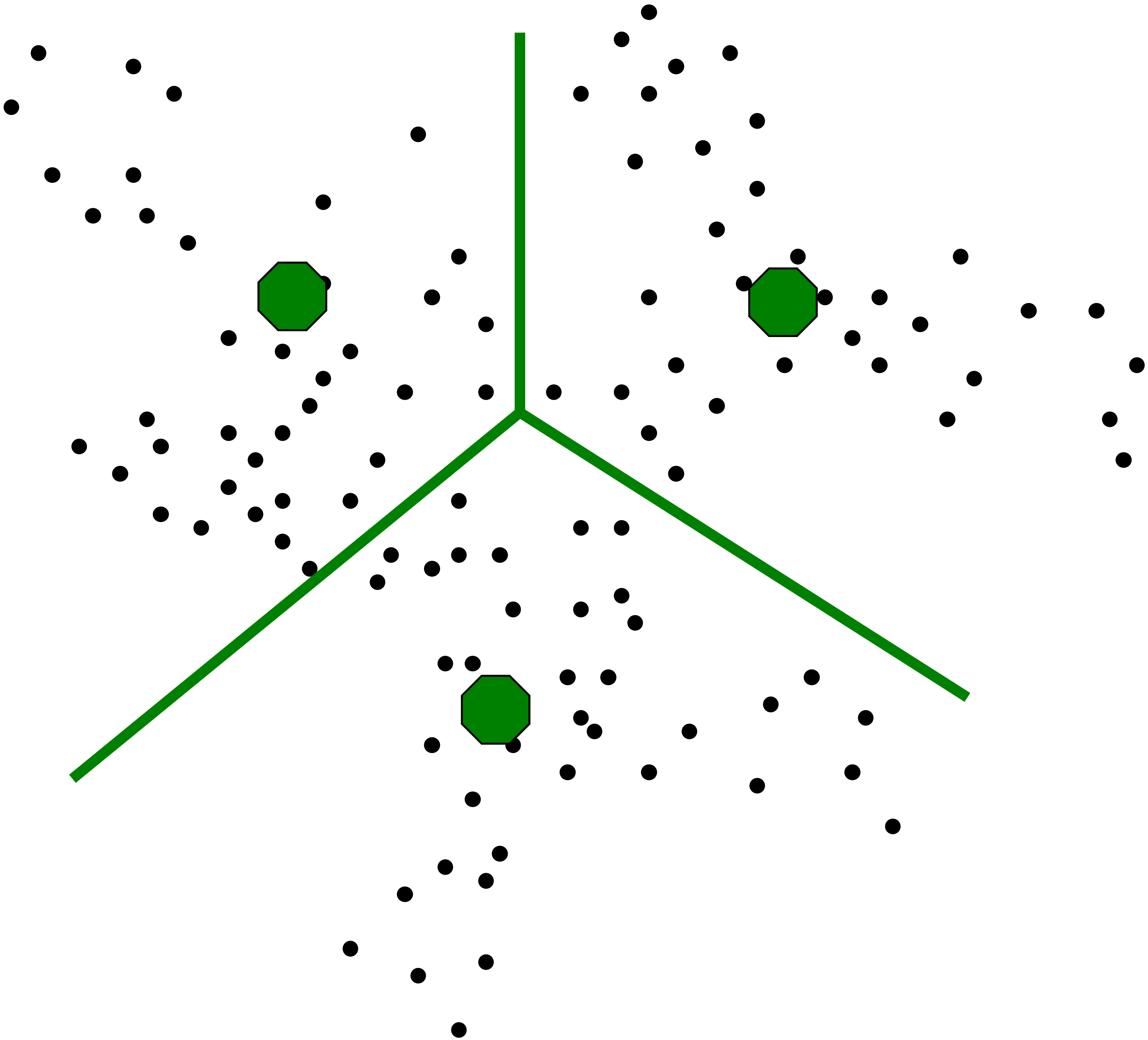


Visual words: main idea



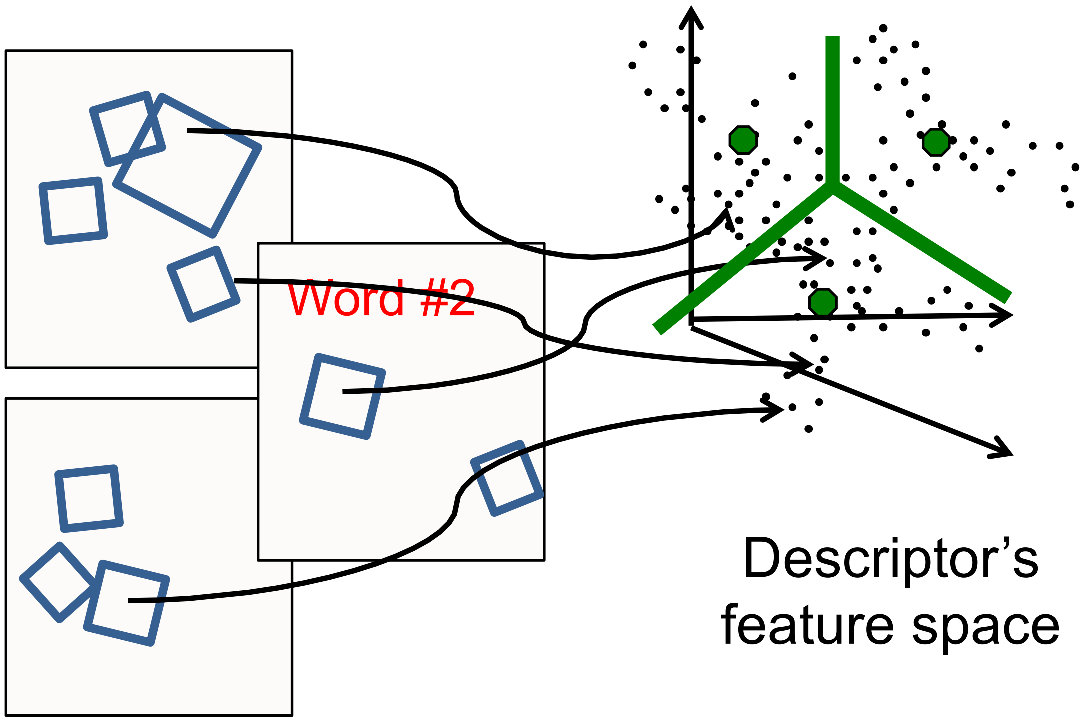


Each point is a local descriptor, e.g. SIFT vector.



Visual words

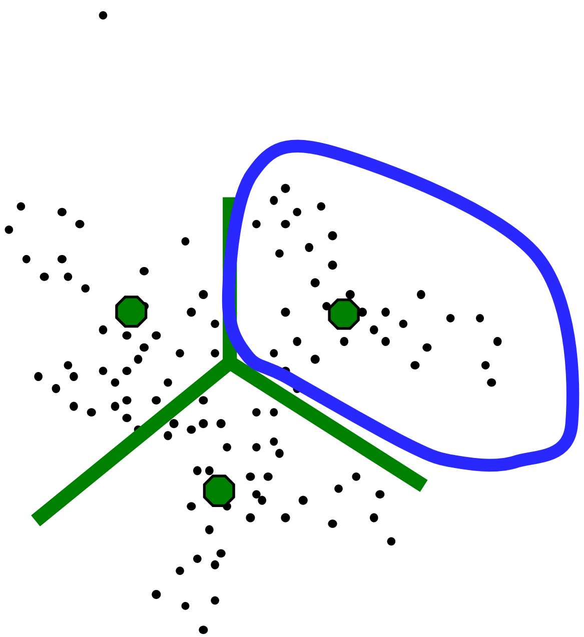
Map high-dimensional descriptors to tokens/words by quantizing the feature space



- Quantize via clustering, let cluster centers be the prototype "words"
- Determine which word to assign to each new image region by finding the closest cluster center.

Visual words

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



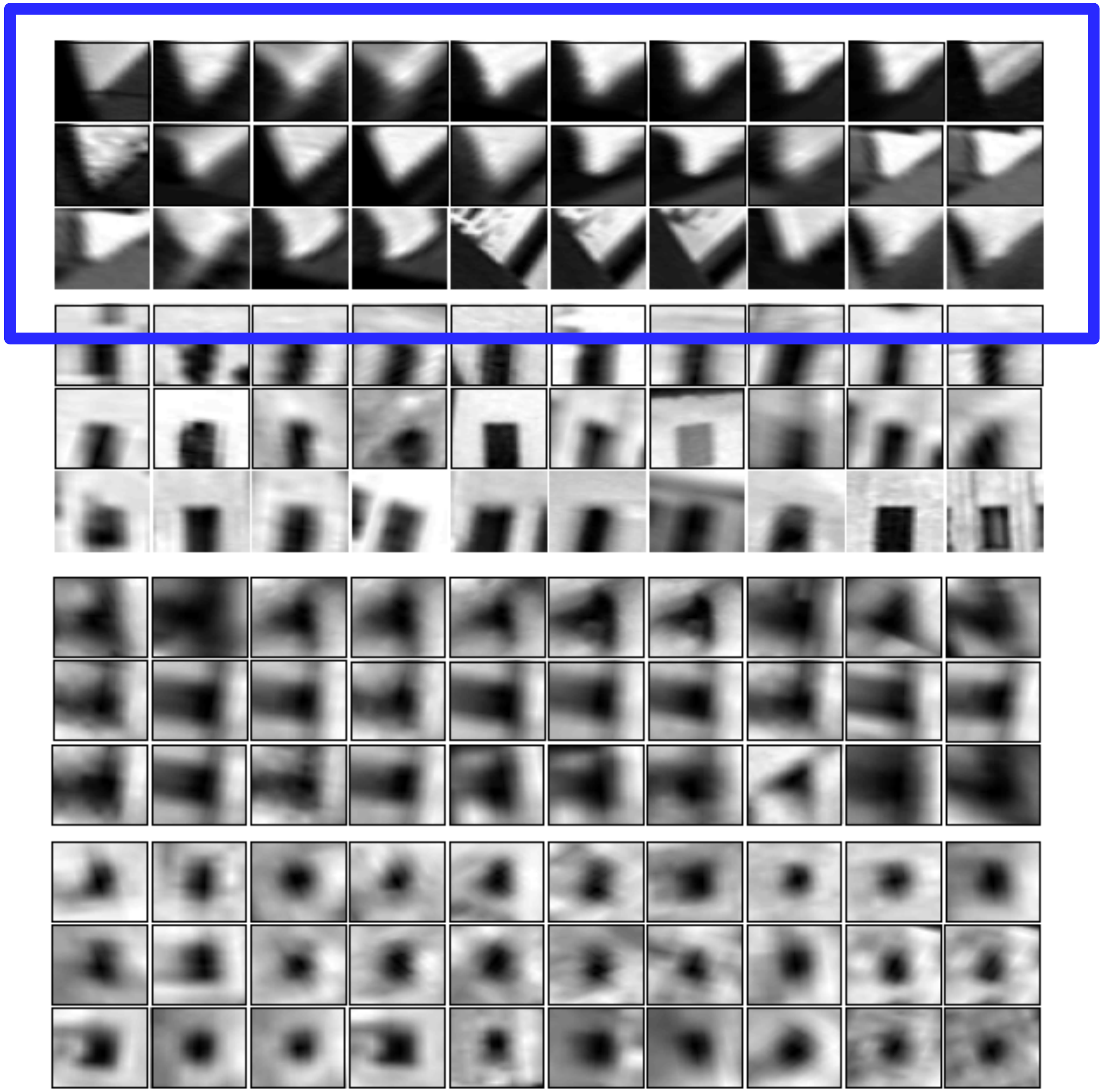


Figure from Sivic & Zisserman, ICCV 2003

Indexing local features

Index

- **Inverted file index**
- For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…
- We want to find all *images* in which a *feature* occurs.
- To use this idea, we'll need to map our features to "visual words".

Kristen Grauman

Inverted file index

Database images

Image #1

Image #2

Image #3

⋮

Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2

…

⋮ ⋮

- Database images are loaded into the index mapping words to image numbers

Kristen Grauman

Inverted file index

New query image

Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2
⋮	⋮

…

Image #1

Image #2

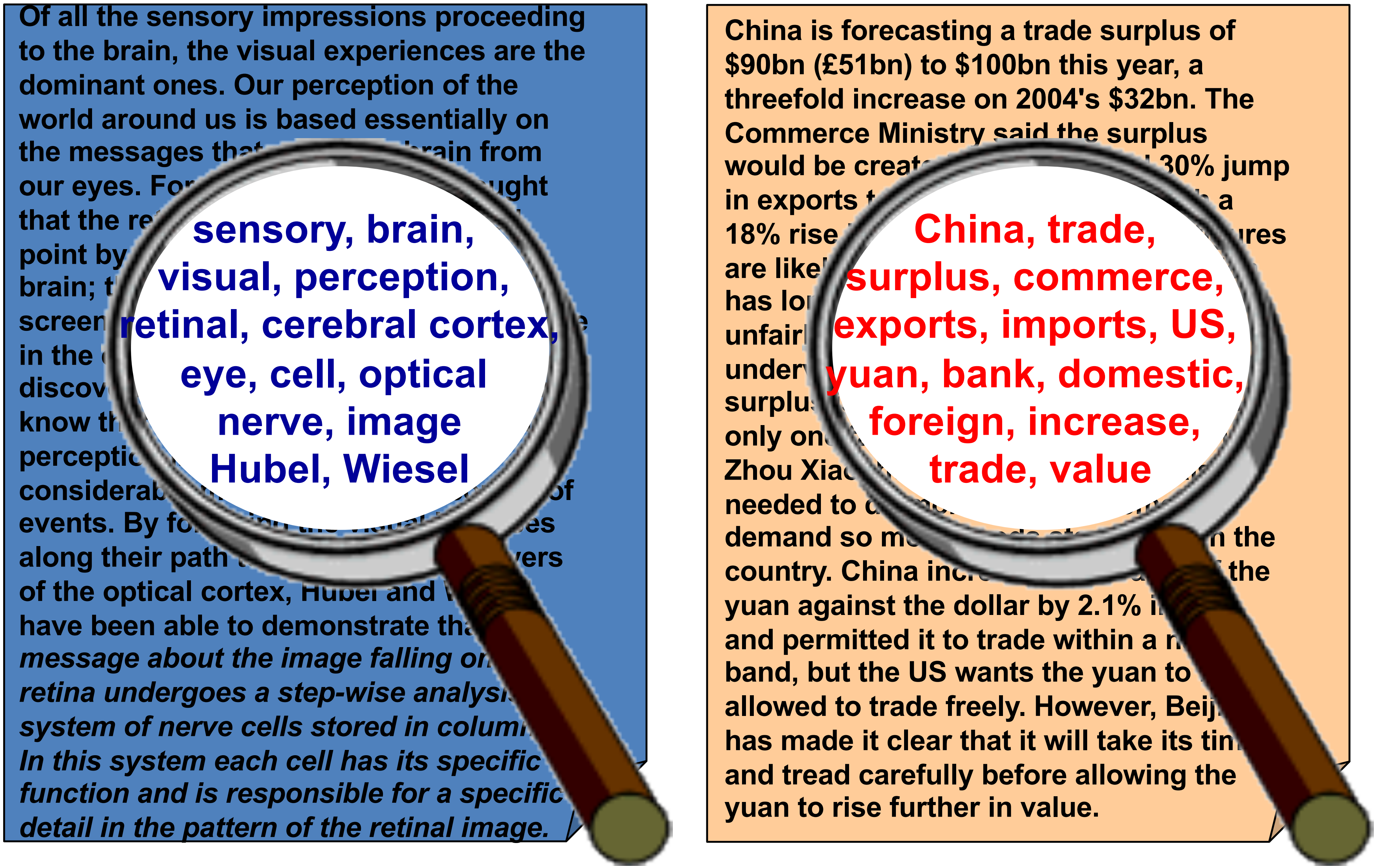
New query image is mapped to indices of database images that share a word.

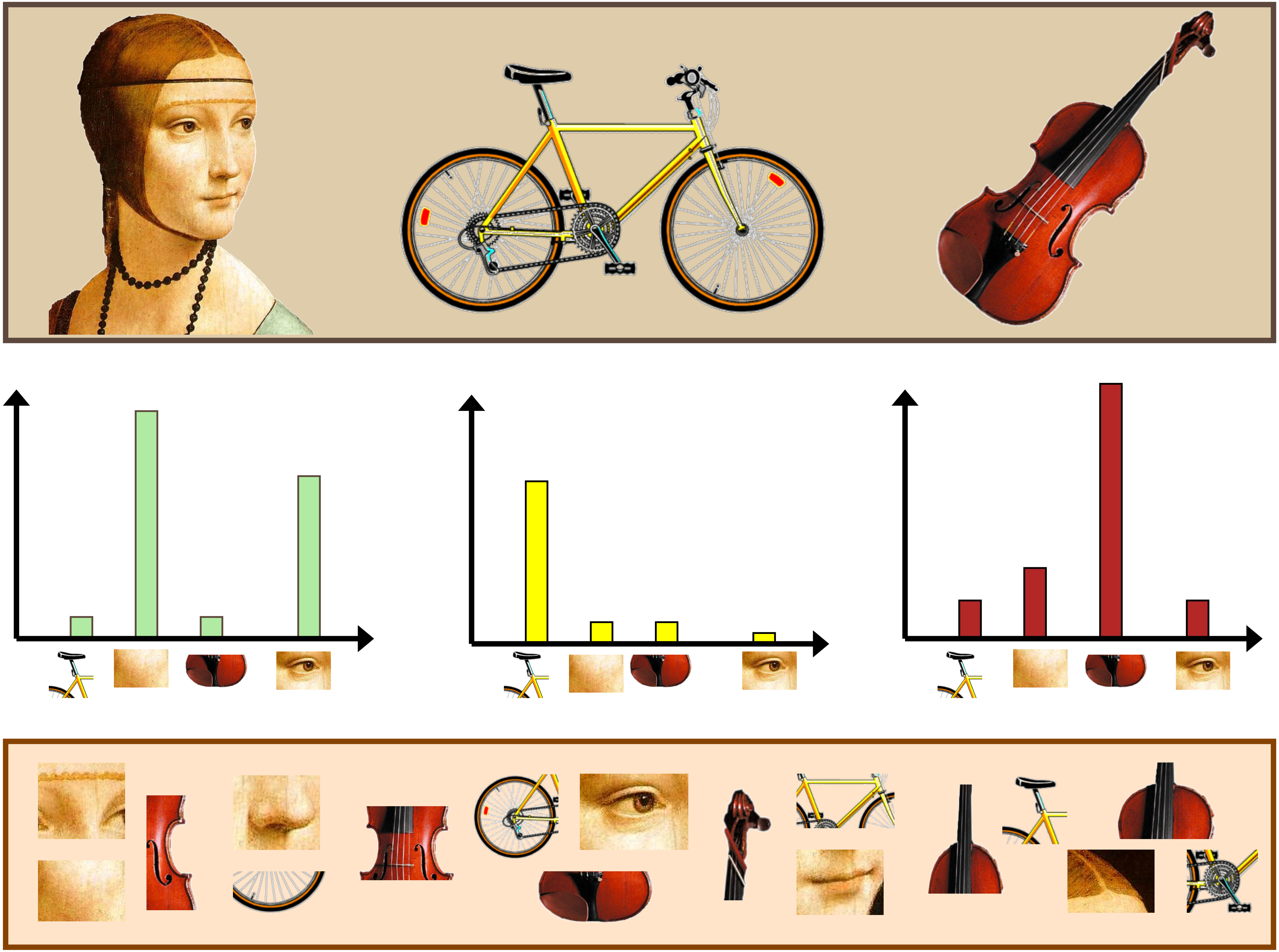
Kristen Grauman

Instance recognition: remaining issues

- **How to summarize the content of an entire image? And gauge overall similarity?**
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Analogy to documents





Bags of visual words

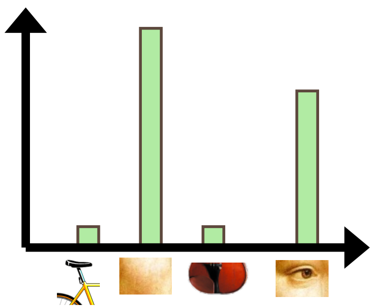
- Summarize entire image based on its distribution (histogram) of word occurrences.

- Analogous to bag of words representation commonly used for documents.

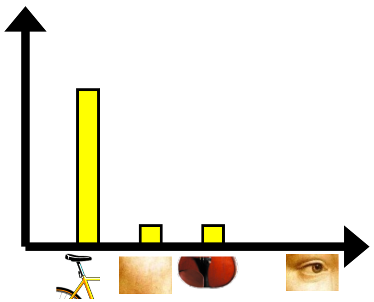
Bags of visual words

- Rank frames by normalized scalar product between their (possibly weighted) occurrence counts--*nearest neighbor* search for similar images.

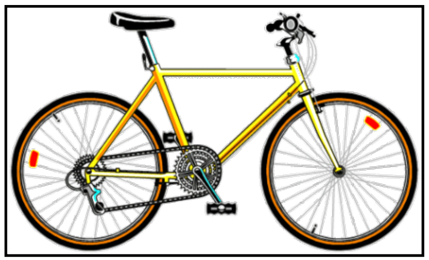
[1 8 1 4]



[5 1 1 0]



$\vec{d}_j$



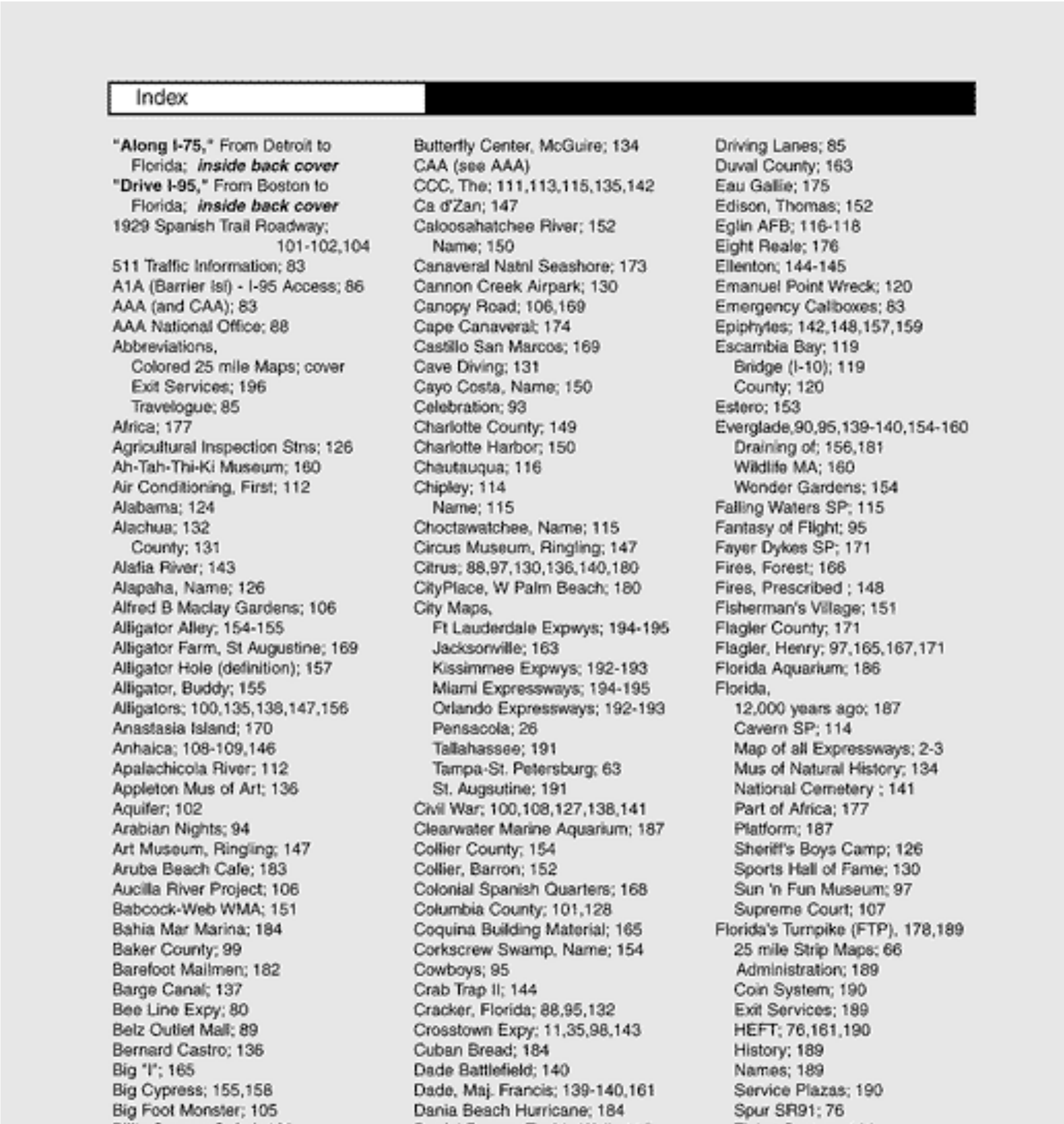
$\vec{q}$

$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

$$= \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}}$$

for vocabulary of *V* words

What else can we borrow from text retrieval?



Index

Weighting the words

- Just as with text, some visual words are more discriminative than others

the, and, or **vs.** ***cow, AT&T, Cher***

- the bigger fraction of the documents a word appears in, the less useful it is for matching
  - e.g., a word that appears in *all* documents is not helping us

tf-idf weighting

- **T**erm **f**requency – **i**nverse **d**ocument **f**requency
- Describe frame by frequency of each word within it, downweight words that appear often in the database
- (Standard weighting for text retrieval)

Number of occurrences of word i in document d

Total number of documents in database

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Number of words in document d

Number of documents word i occurs in

Large-scale image search

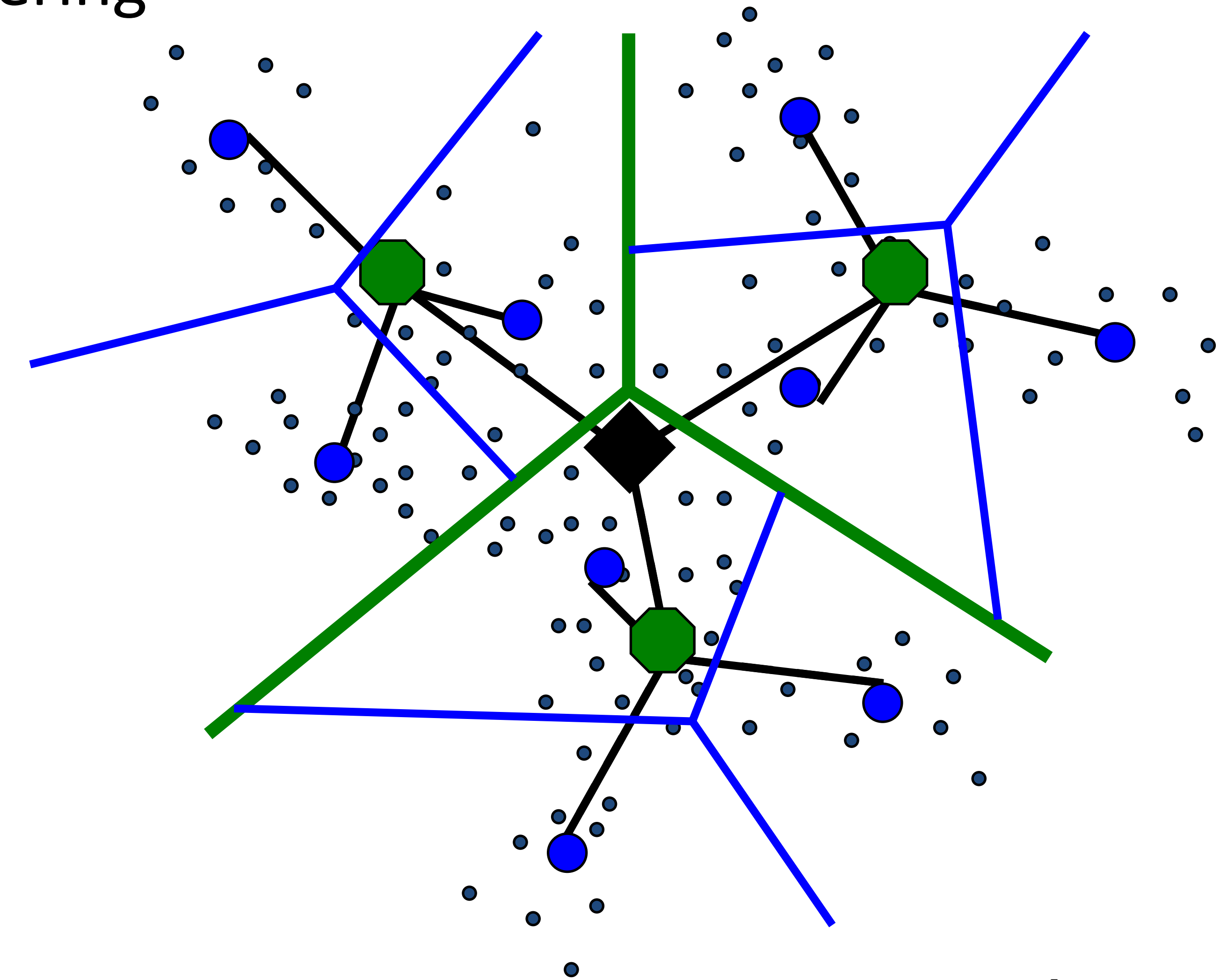
- Build the database:
  - Extract features from the database images
  - Learn a vocabulary using k-means (typical k: 100,000)
  - Compute *weights* for each word
  - Create an inverted file mapping words → images

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- **How large should the vocabulary be? How to perform quantization efficiently?**
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Vocabulary Tree

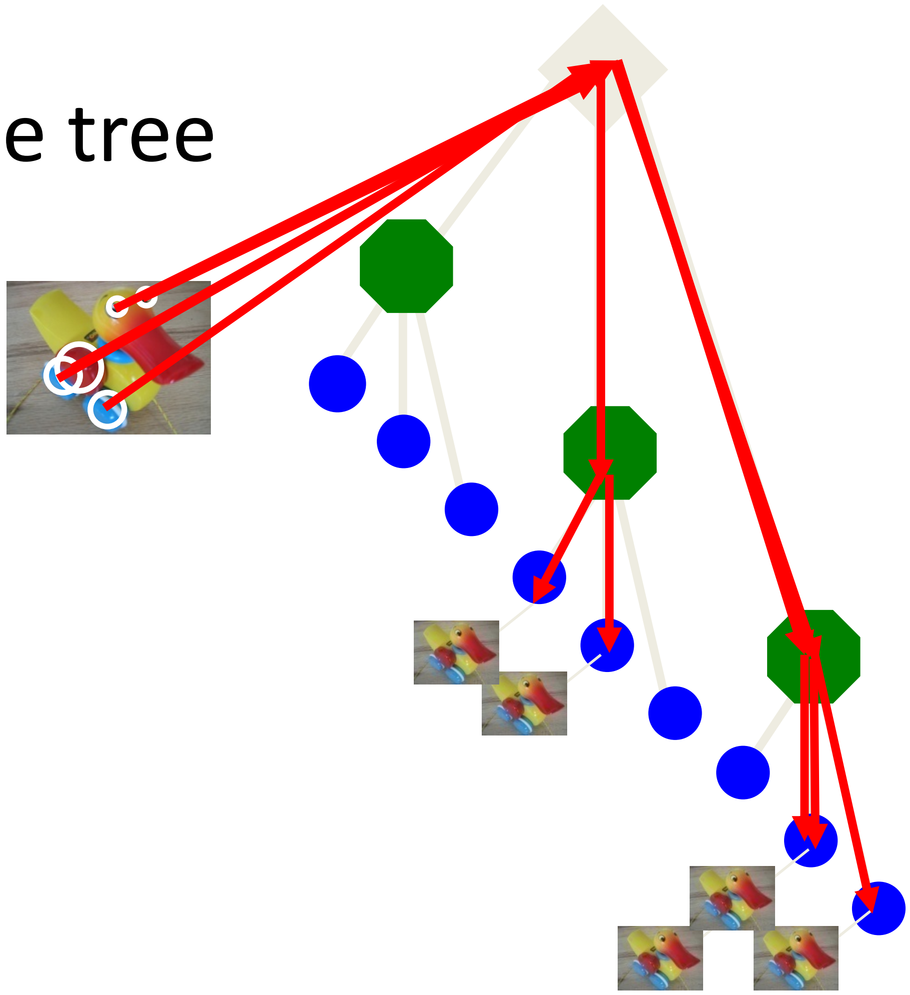
- Large vocabularies can be improved with hierarchical clustering



[Nister & Stewenius, CVPR'06]

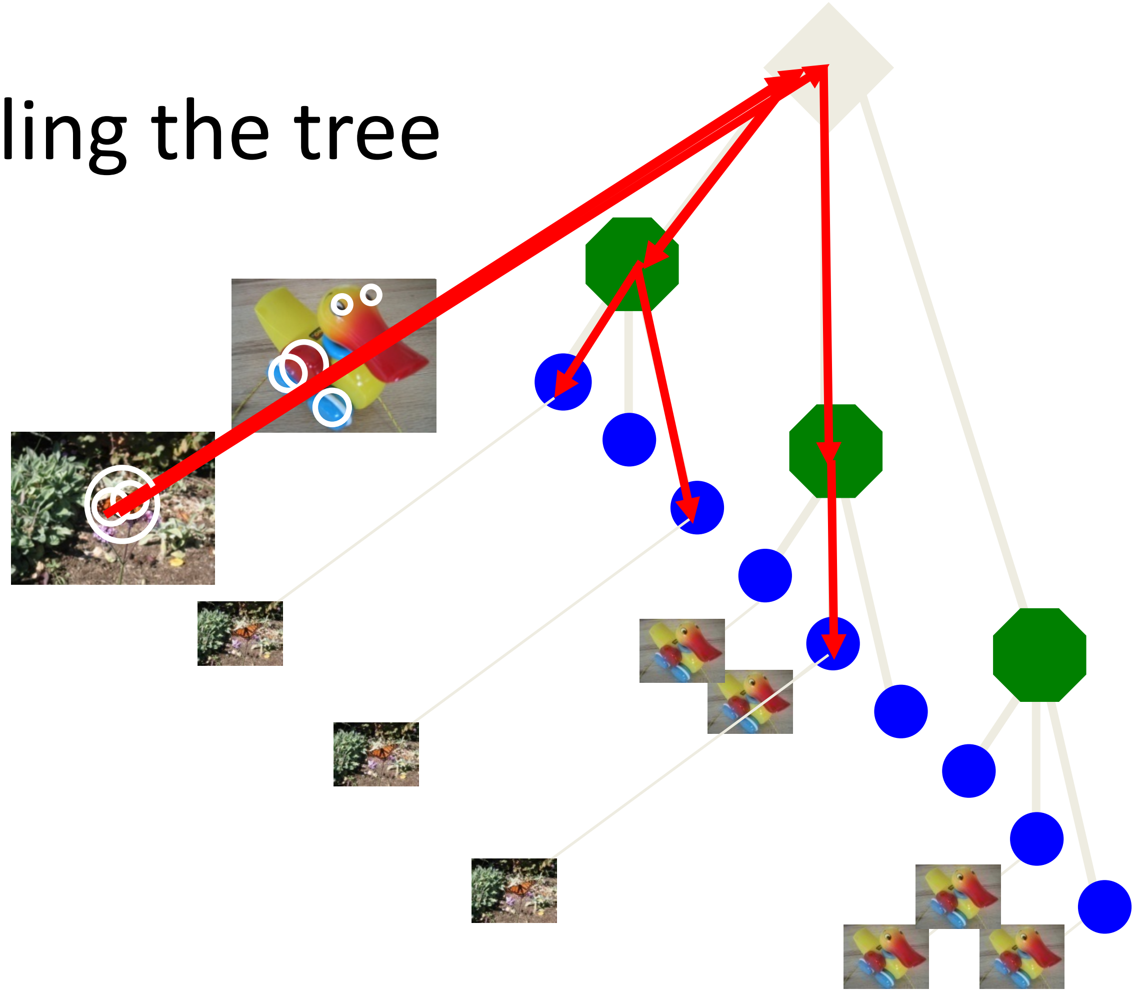
Vocabulary Tree

- Training: Filling the tree



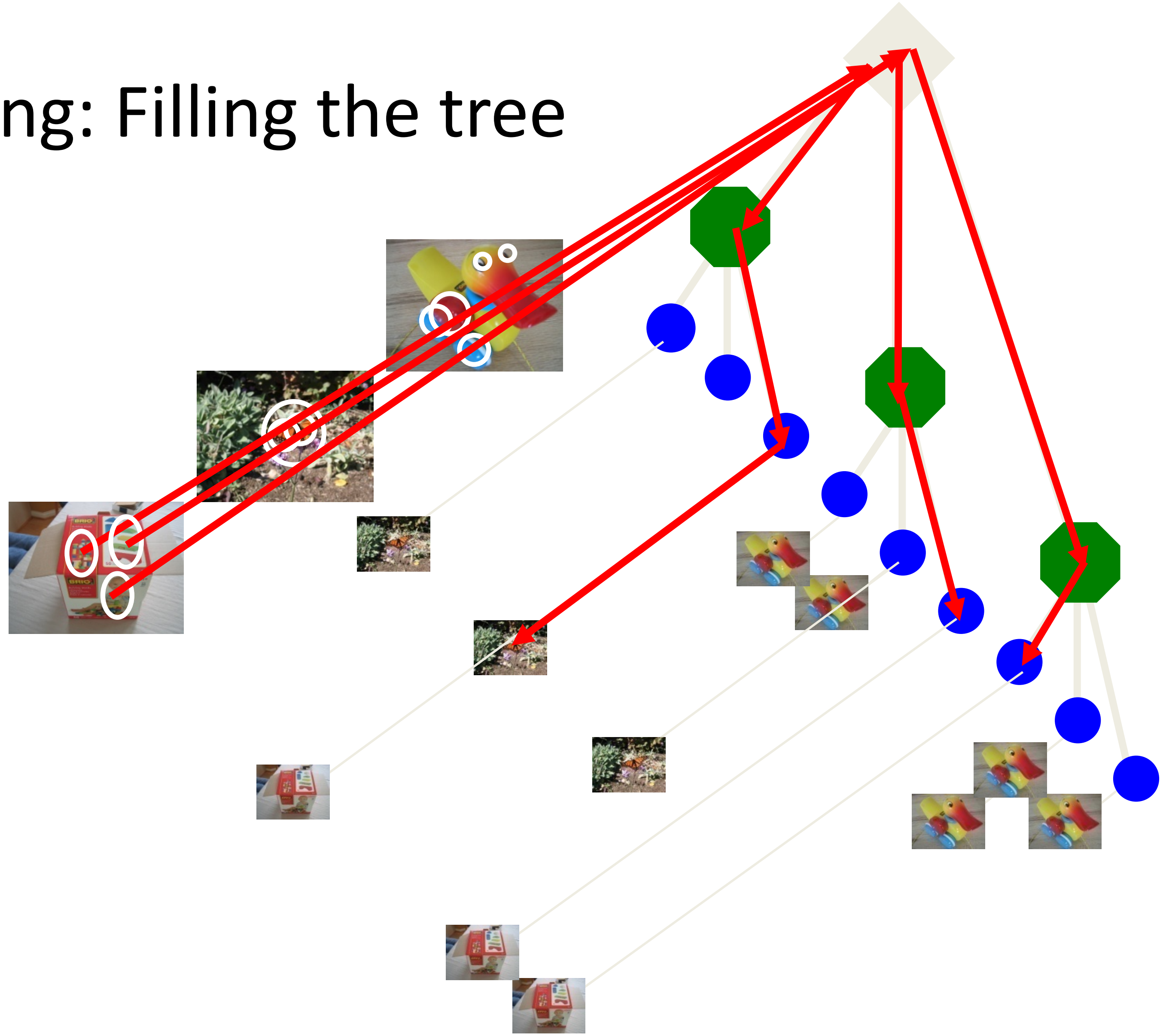
Vocabulary Tree

- Training: Filling the tree



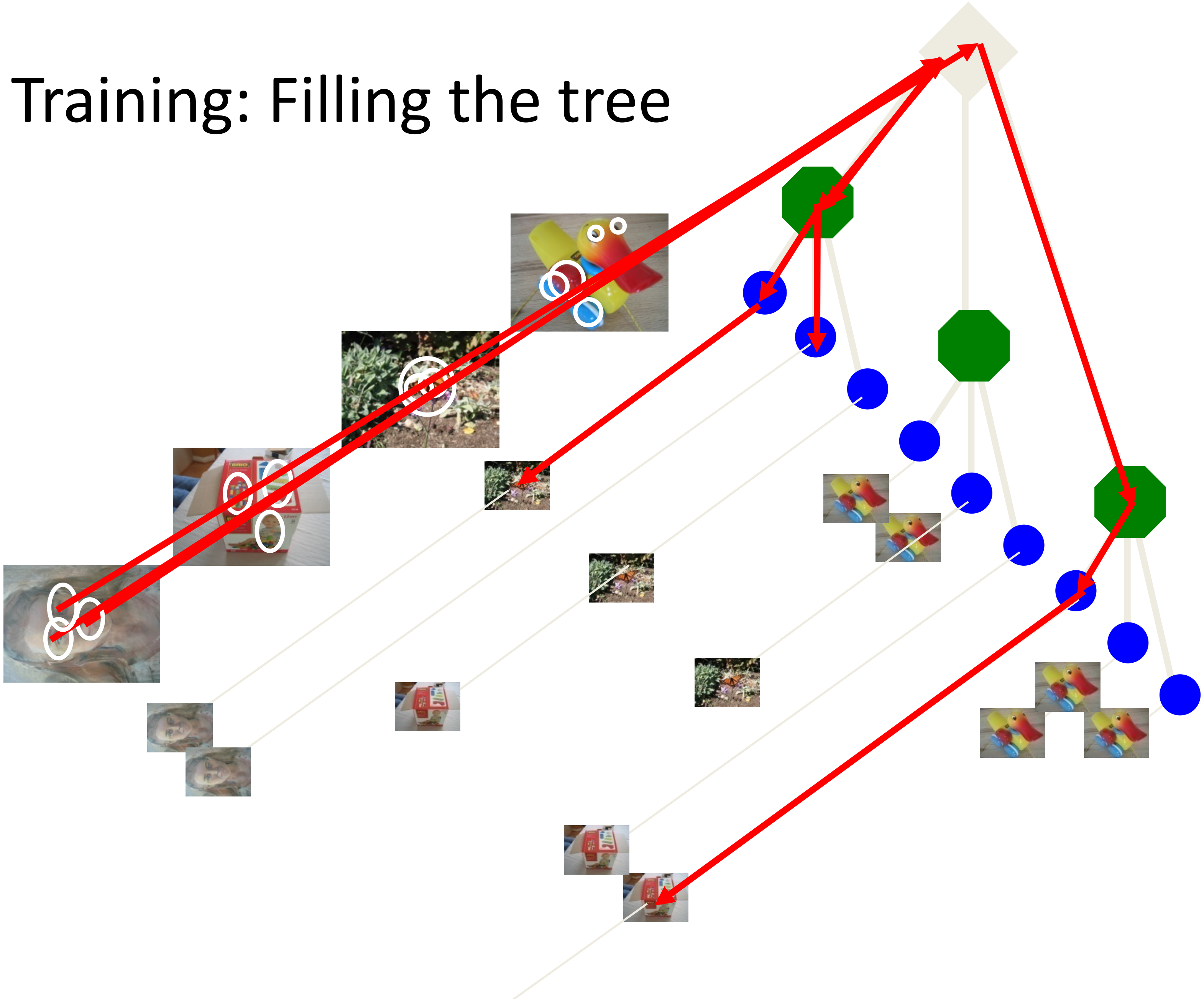
Vocabulary Tree

- Training: Filling the tree



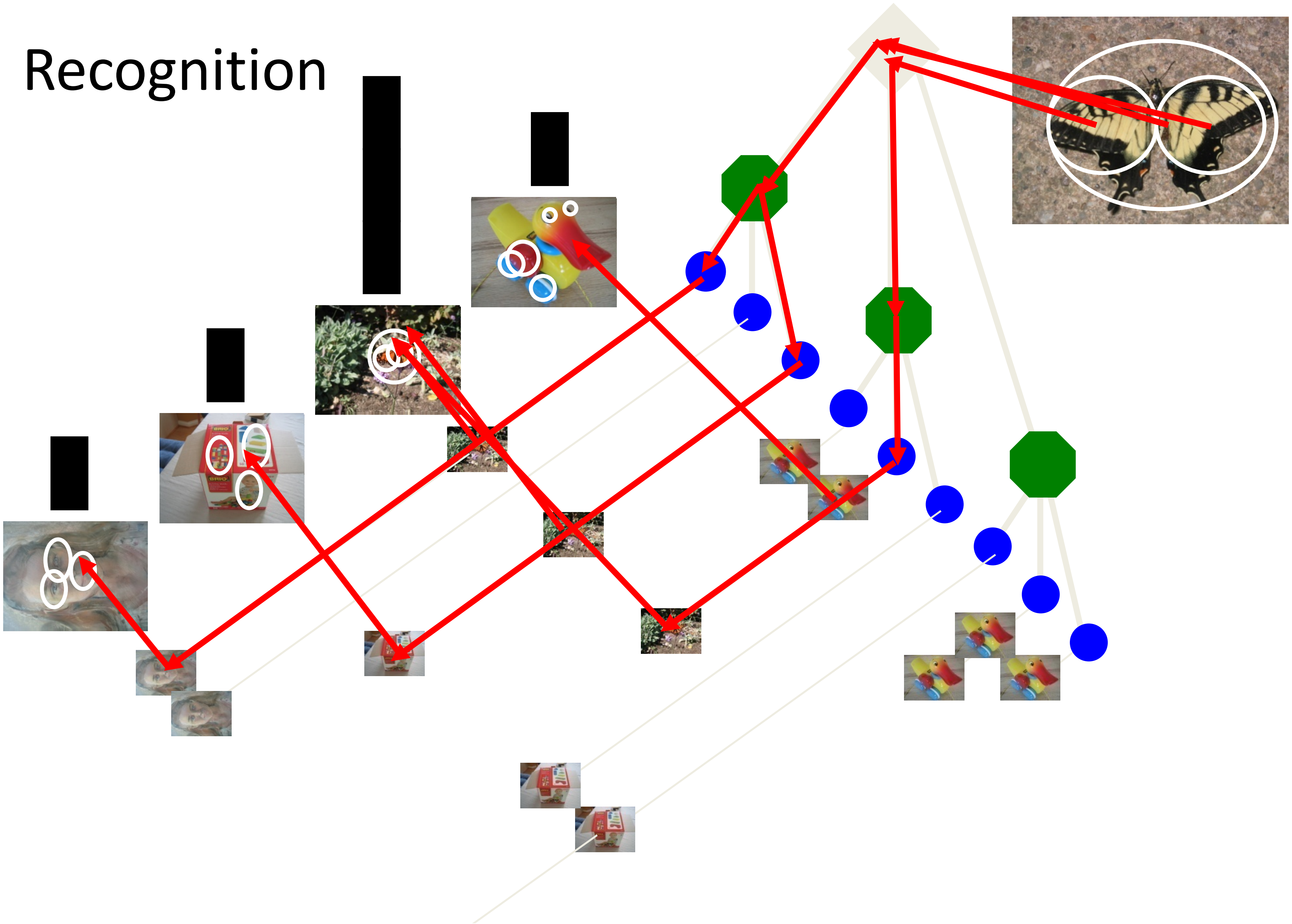
Vocabulary Tree

- Training: Filling the tree

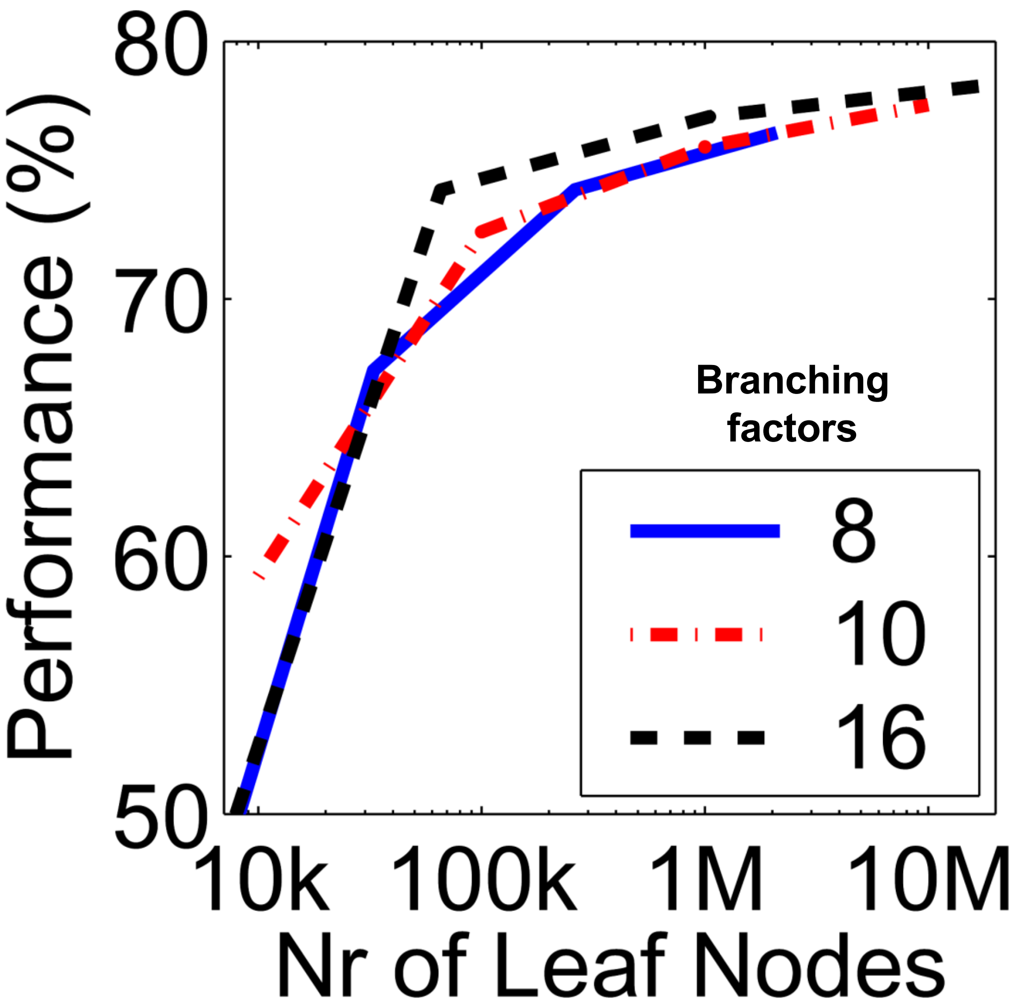


Vocabulary Tree

- Recognition

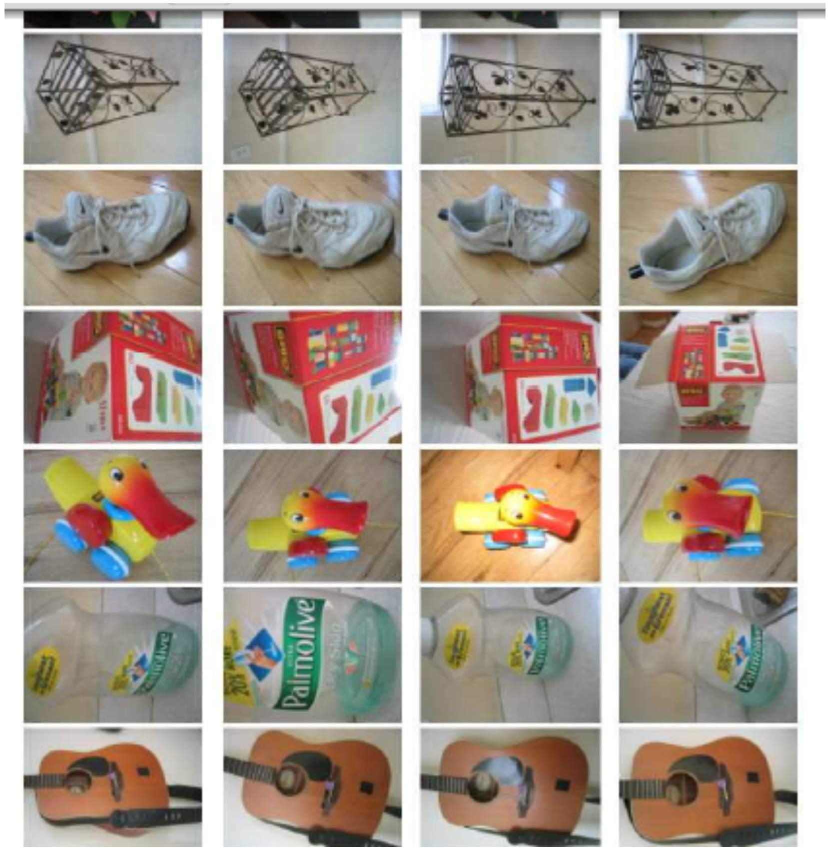


Vocabulary size



Influence on performance, sparsity

Results for recognition task with 6347 images



Nister & Stewenius, CVPR 2006

Kristen Grauman

Visual vocabulary formation

Issues:

- Vocabulary size, number of words
- Sampling strategy: where to extract features?
- Clustering / quantization algorithm
- Unsupervised vs. supervised
- What corpus provides features (universal vocabulary?)

Kristen Grauman

Visual words/bags of words

+ flexible to geometry / deformations / viewpoint
+ compact summary of image content
+ provides vector representation for sets
+ very good results in practice

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear
- basic model ignores geometry – must verify afterwards, or encode via features

Kristen Grauman

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- **Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?**

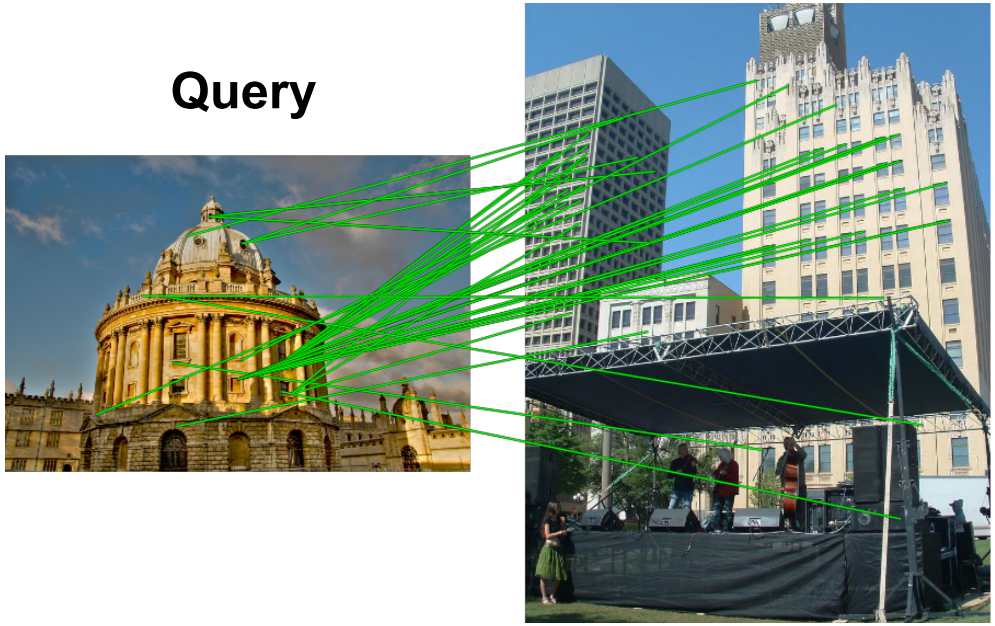
Kristen Grauman

Spatial Verification

Query

DB image with high BoW similarity

Query



DB image with high BoW similarity

Both image pairs have many visual words in common.

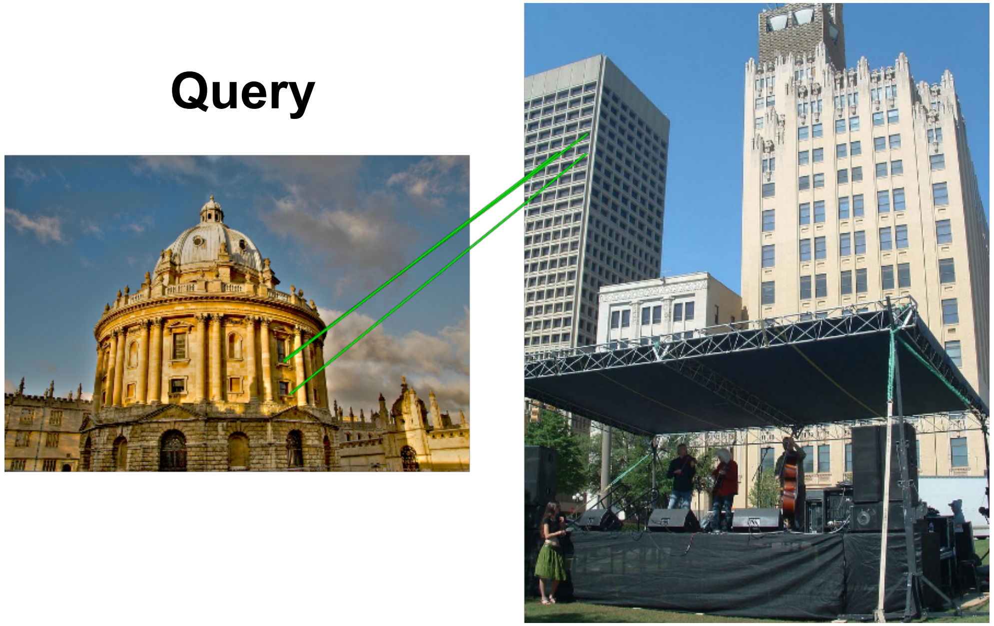
Slide credit: Ondrej Chum

Spatial Verification

Query

DB image with high BoW similarity

Query

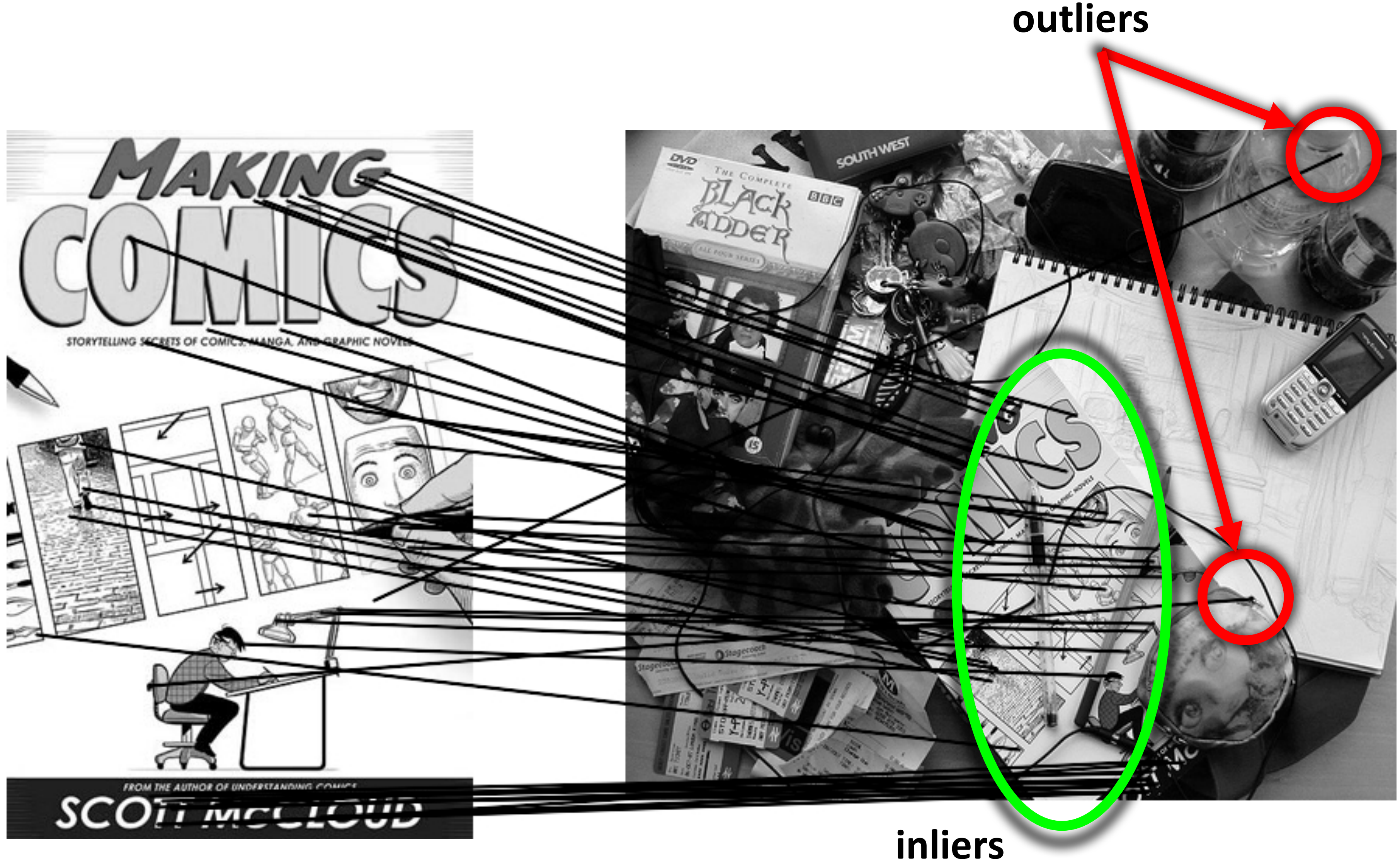


DB image with high BoW similarity

Only some of the matches are mutually consistent

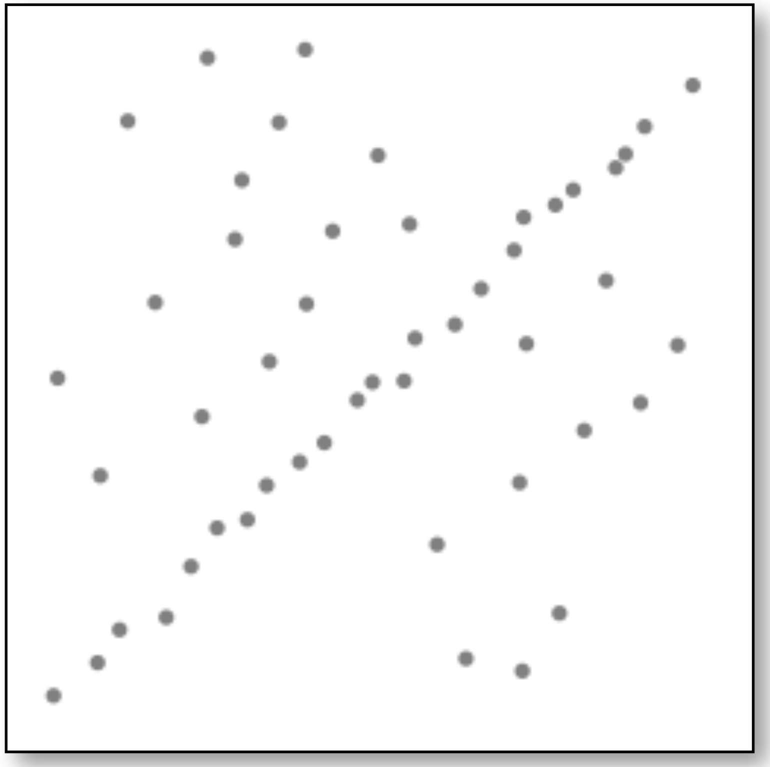
Slide credit: Ondrej Chum

Outliers

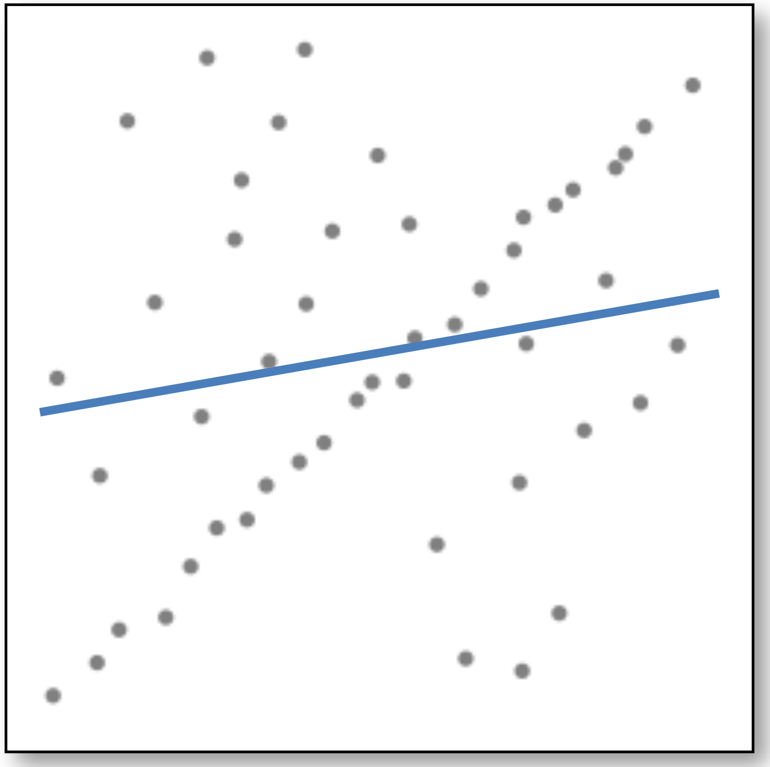

outliers
MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD
DVD
The Complete
BLACK ADDER
BBC
ALL FOUR SERIES
SOUTH WEST
Stagecoach
inliers

Robustness

- A simpler example: linear regression



Problem: Fit a line to these datapoints



Least squares fit

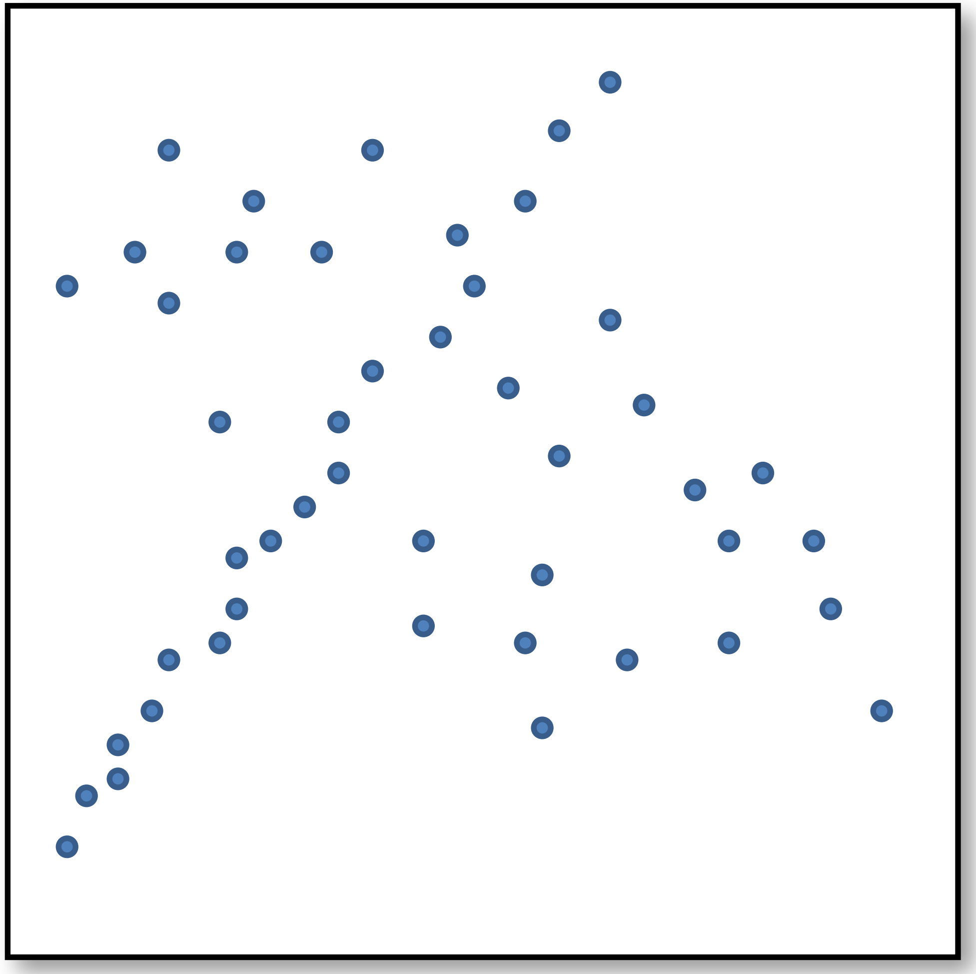
- How can we fix this?

Idea

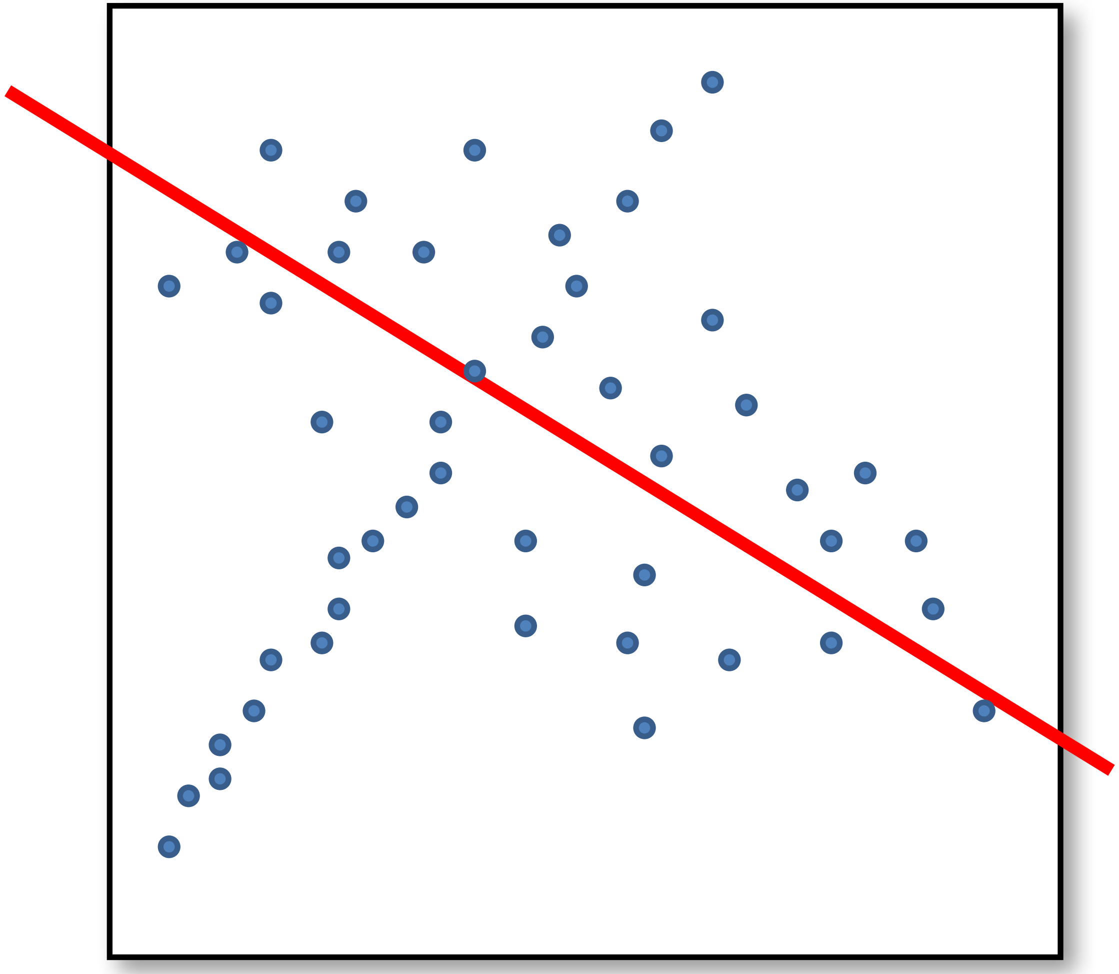
- Given a hypothesized line
- Count the number of points that "agree" with the line
  - "Agree" = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

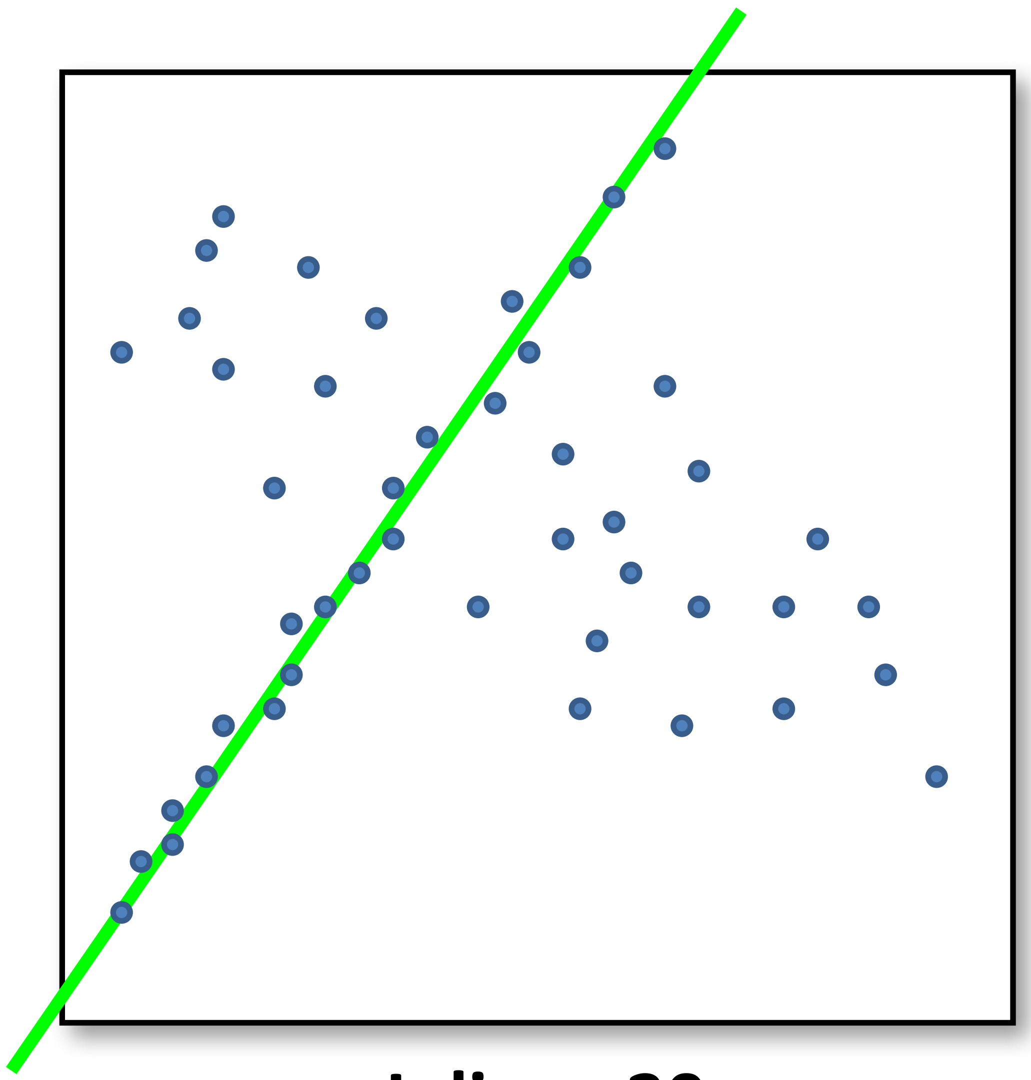


Counting inliers



Inliers: 3

Counting inliers



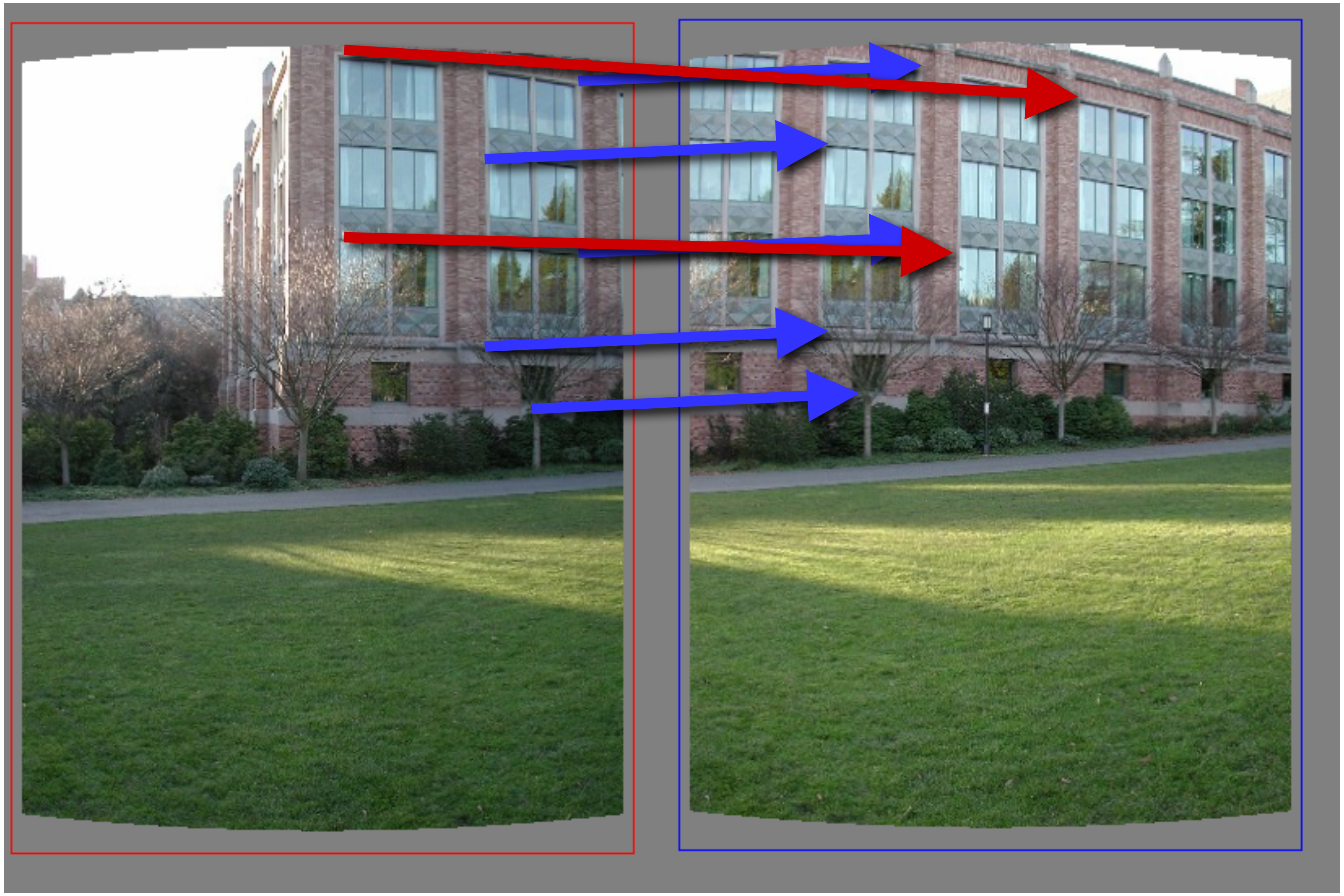
Inliers: 20

How do we find the best line?

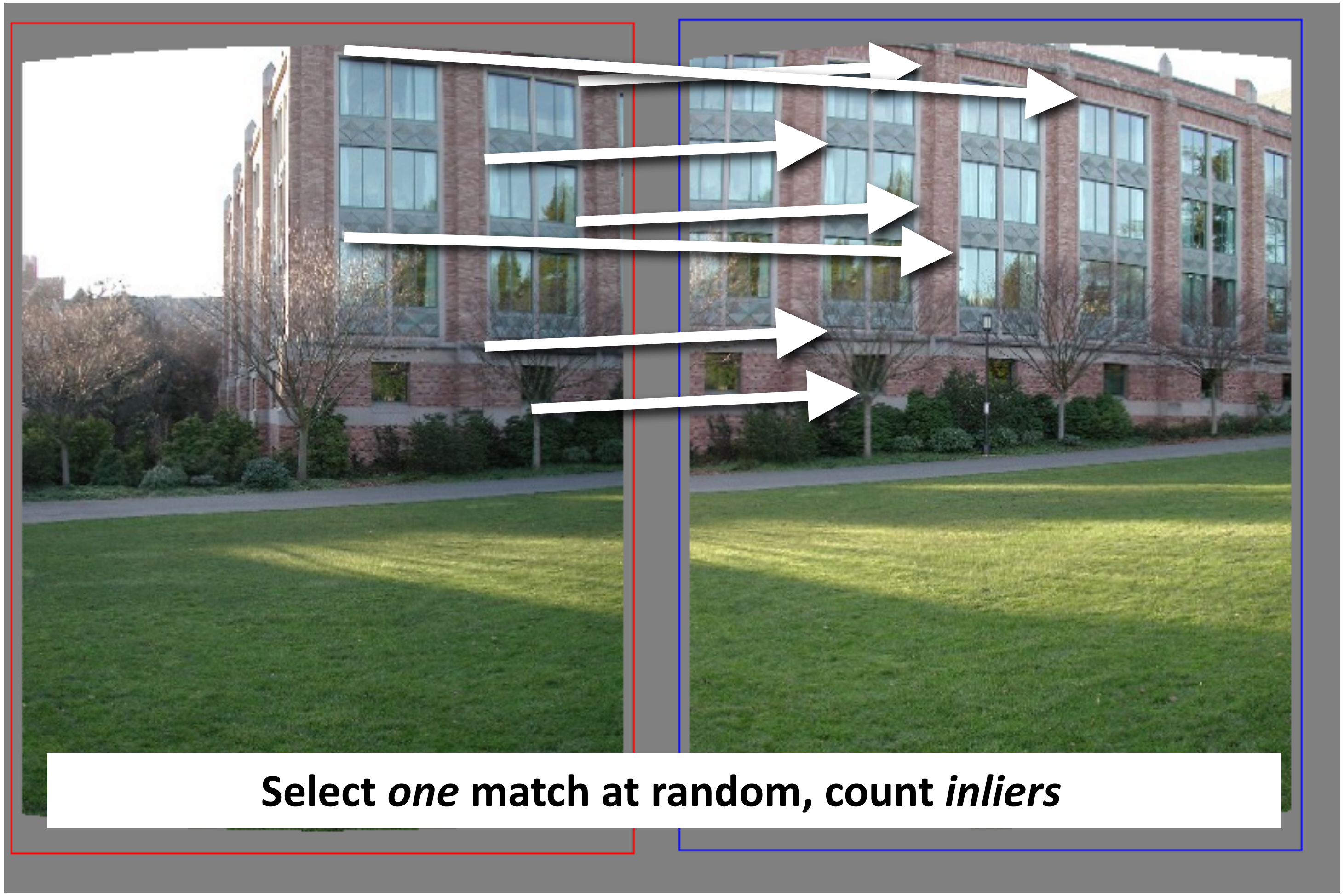
- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

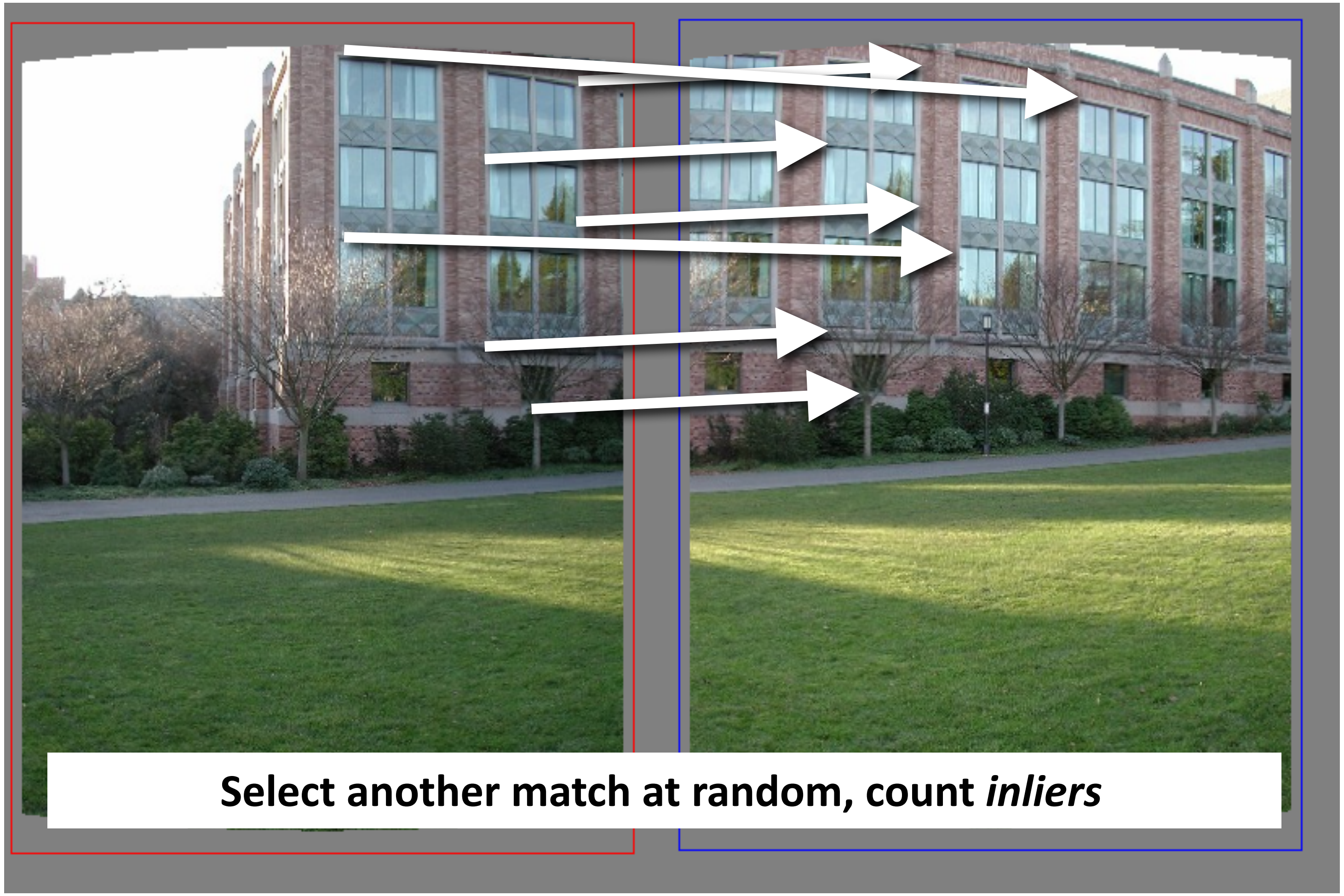
Translations



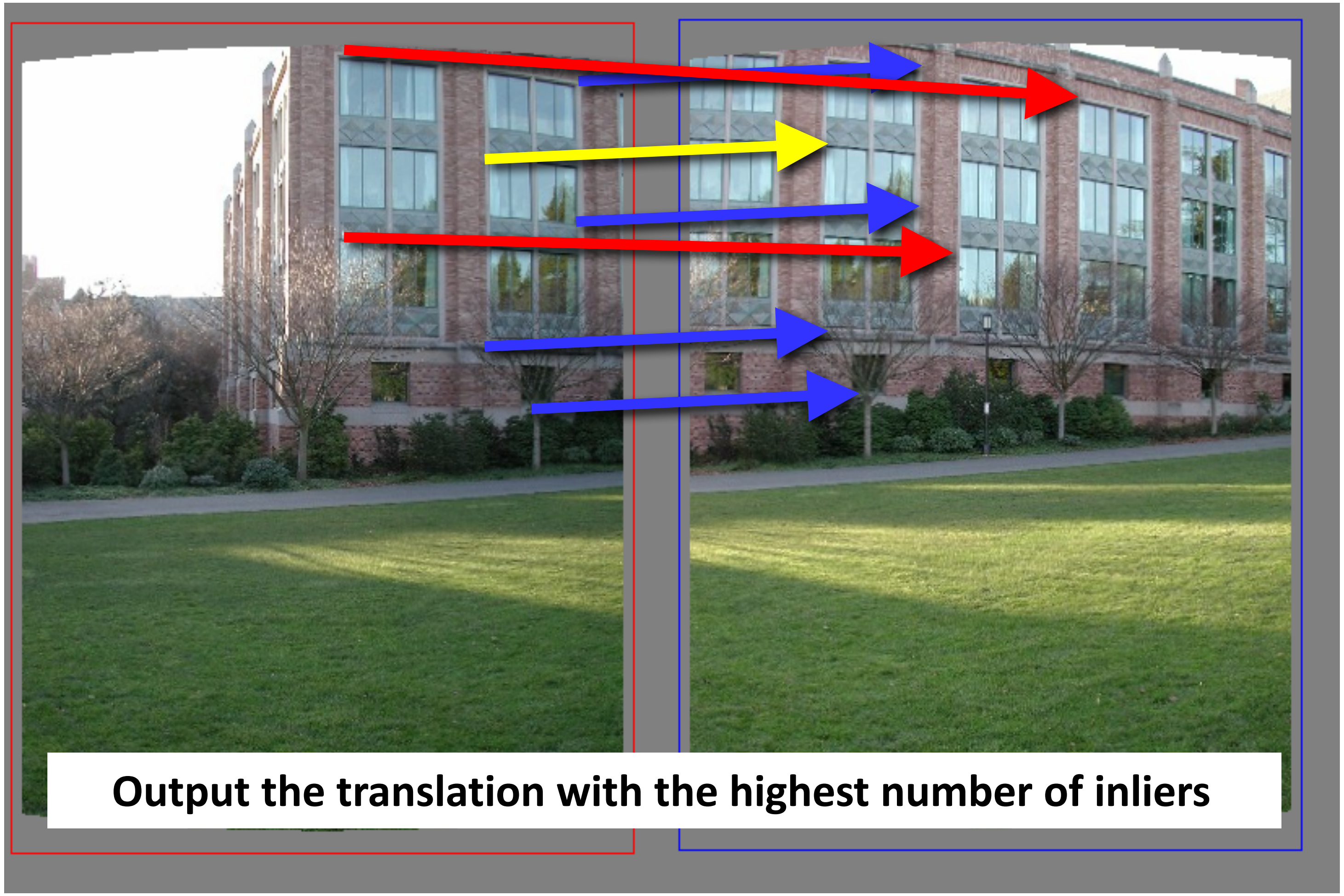
RAndom SAmple Consensus



<u>RA</u>ndom <u>SA</u>mple <u>C</u>onsensus



RAndom SAmple Consensus



RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other

- RANSAC only has guarantees if there are < 50% outliers

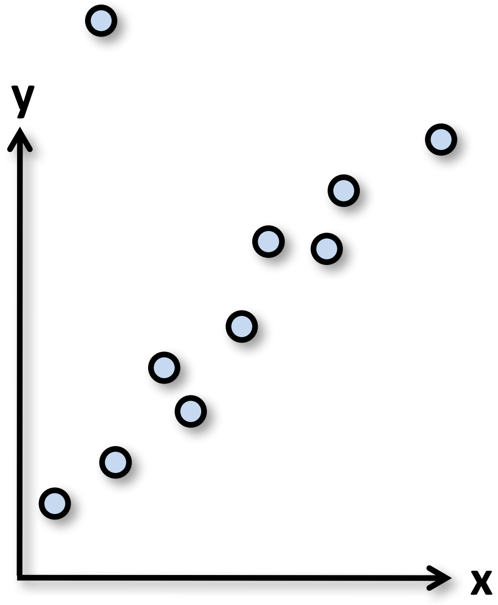
RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)

- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we would like to guarantee

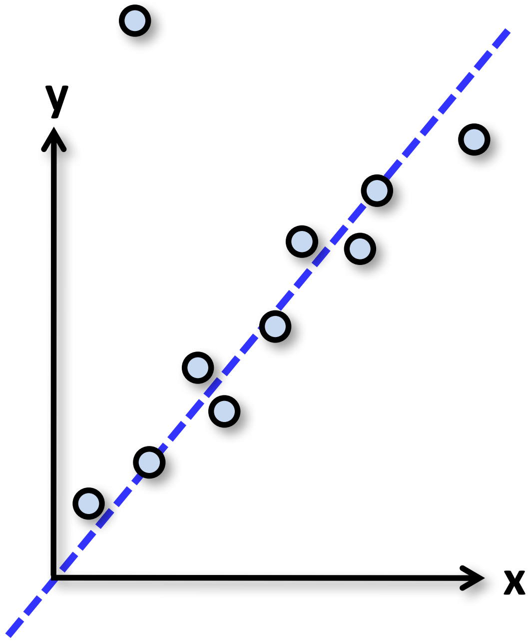
RANSAC

- Back to linear regression
- How do we generate a hypothesis?



RANSAC

- Back to linear regression
- How do we generate a hypothesis?



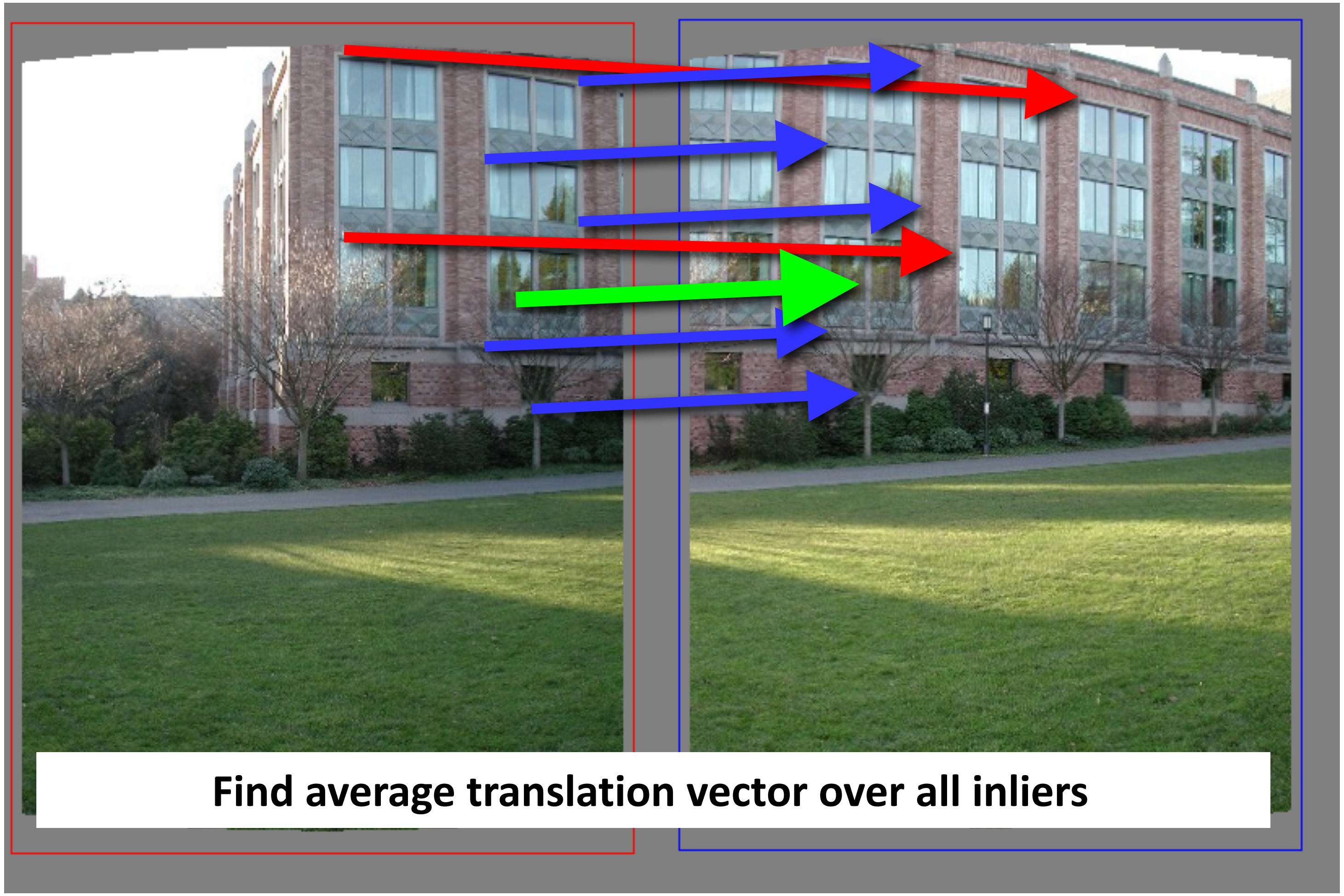
RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

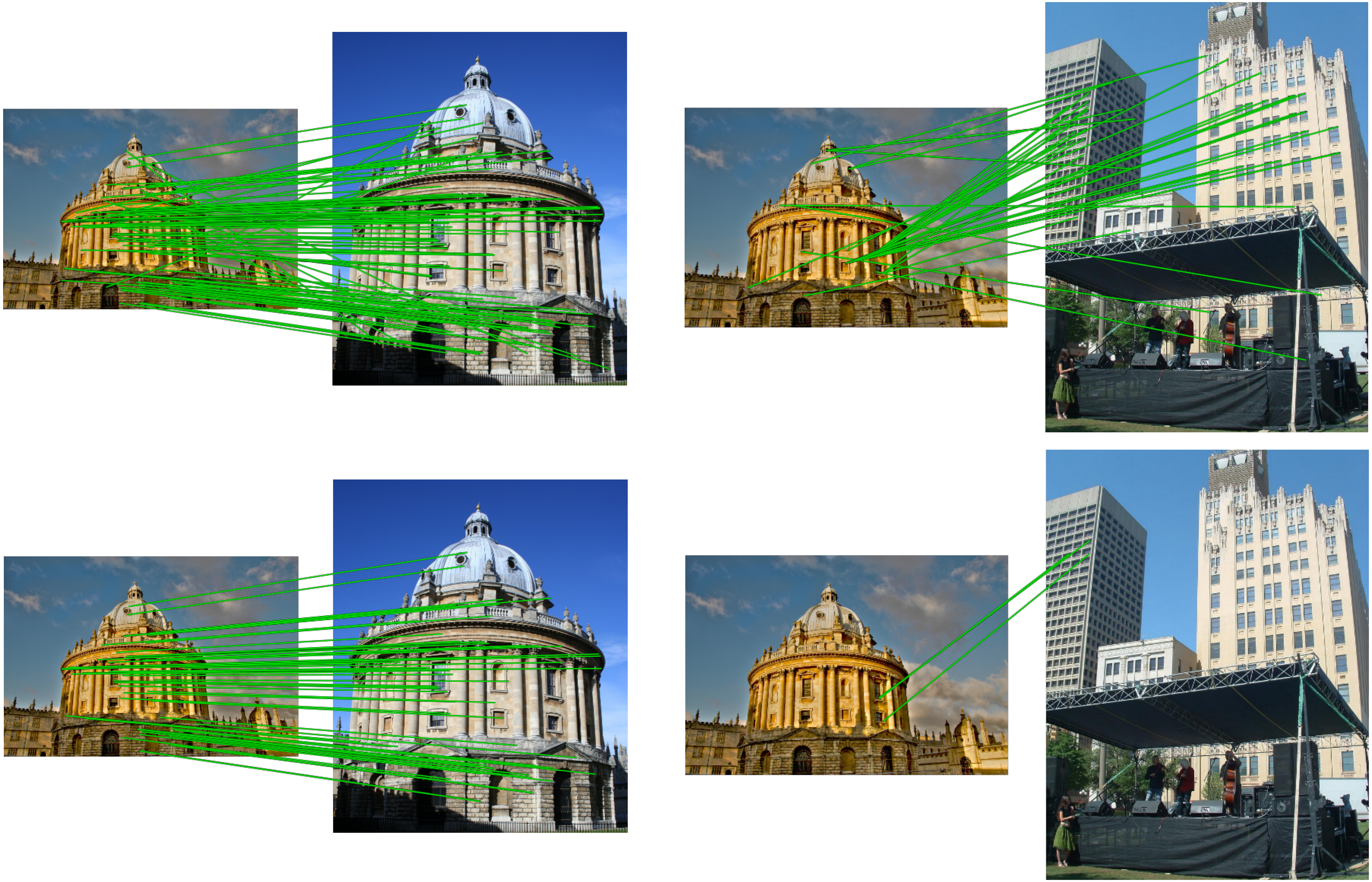
RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

Final step: least squares fit



RANSAC verification



Recognition via alignment

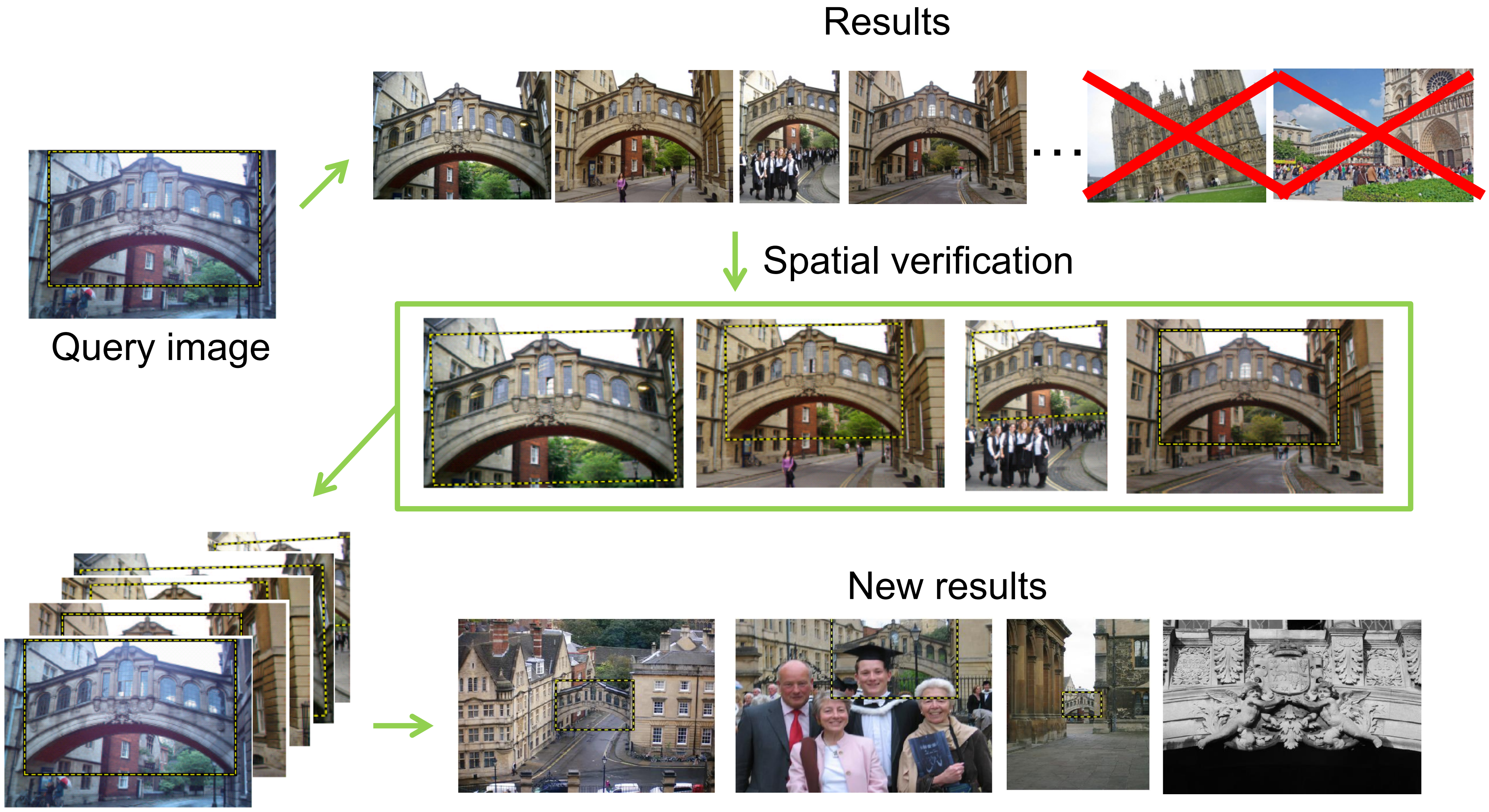
Pros:

- Effective when we are able to find reliable features within clutter
- Great results for matching specific instances

Cons:

- Scaling with number of models
- Spatial verification as post-processing – not seamless, expensive for large-scale problems
- Not suited for category recognition.

Query Expansion



Chum, Philbin, Sivic, Isard, Zisserman: Total Recall…, ICCV 2007

Slide credit: Ondrej Chum