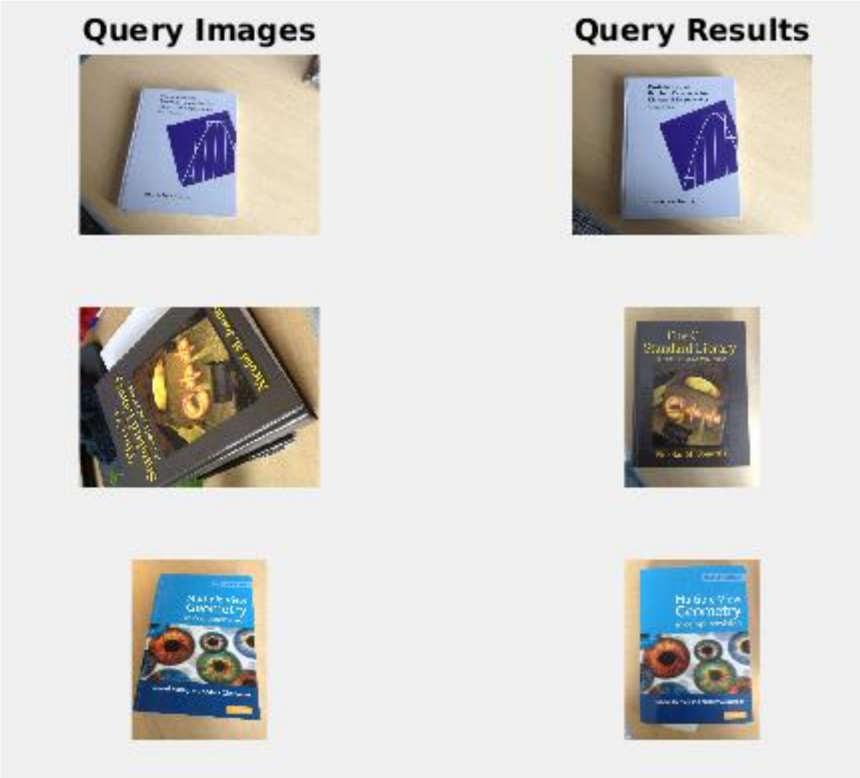
University of Zurich[UZH] | ETH zürich
Institute of Informatics – Institute of Neuroinformatics
ROBOTICS & PERCEPTION GROUP

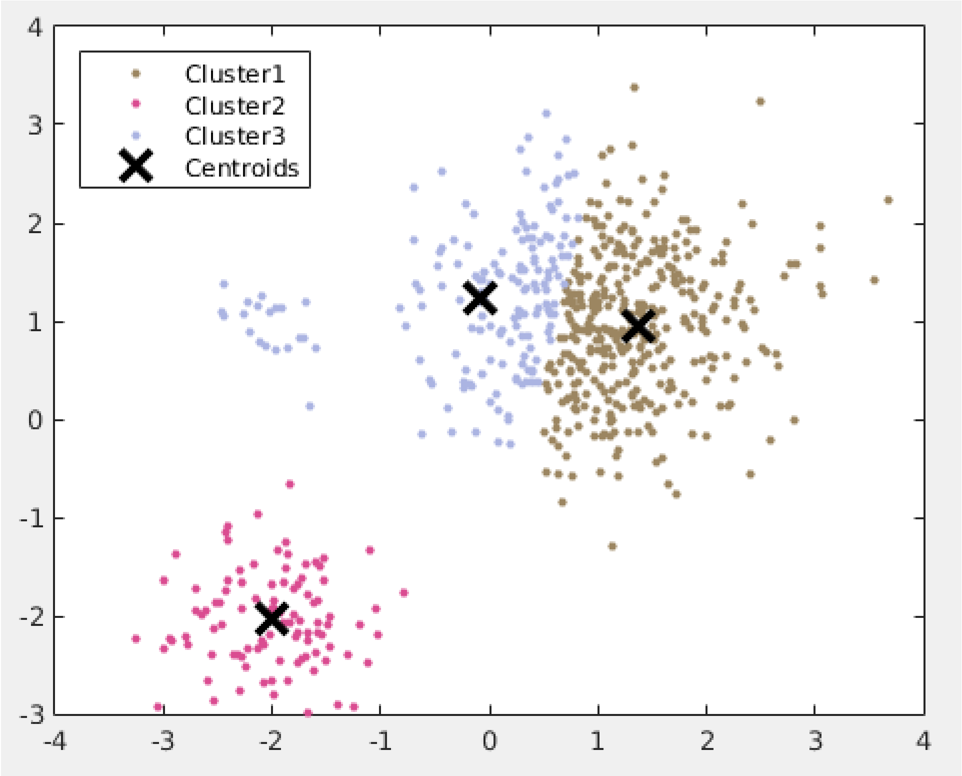
Lecture 12b
Place Recognition

Davide Scaramuzza
http://rpg.ifi.uzh.ch/

Lab exercise today replaced by Deep Learning Tutorial by Daniel Gehrig

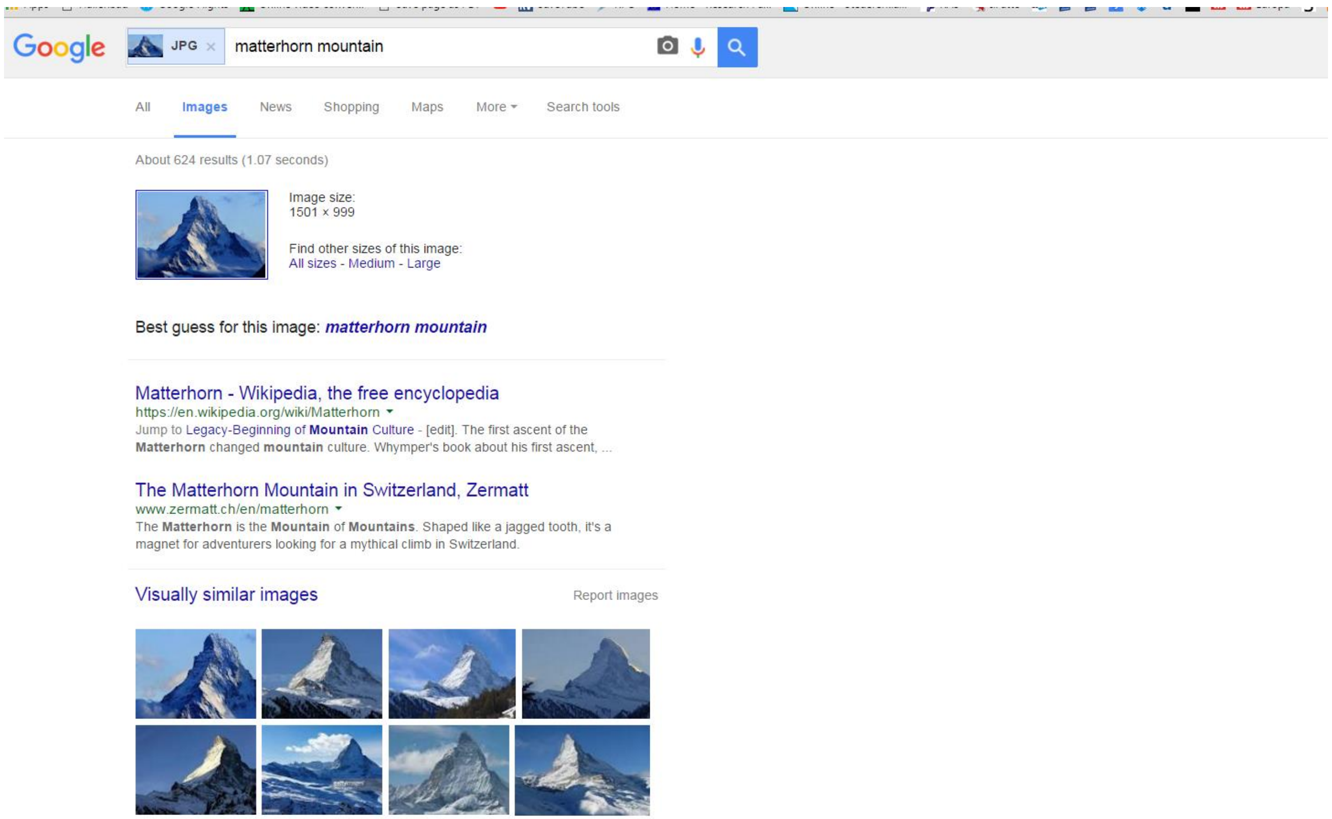
- Room ETH HG E 1.1 from 13:15 to 15:00
- Optional lab exercise is online: K-means clustering and place recognition with Bag of Words





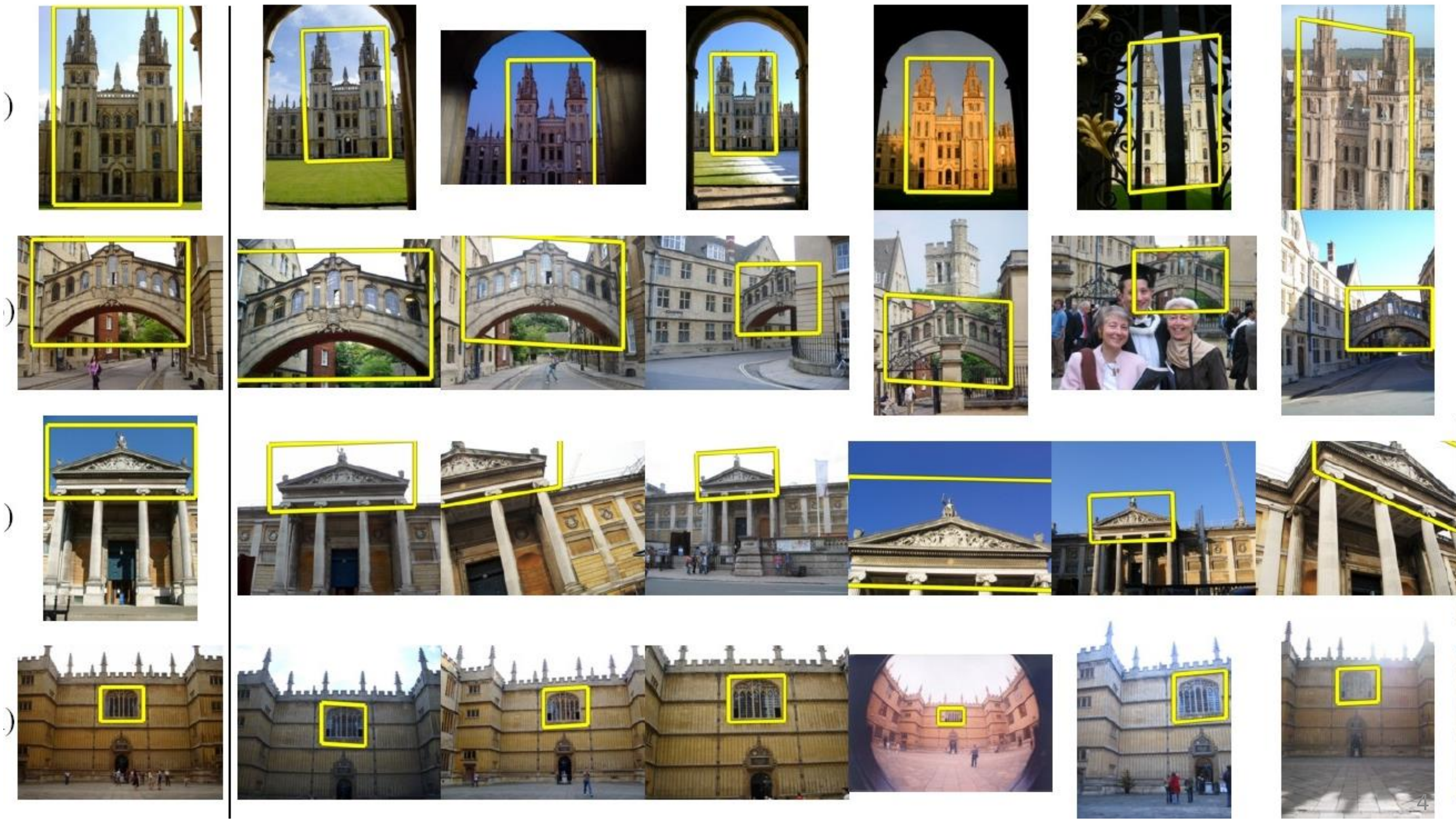
Place Recognition

- **Robotics**: Has the robot been to this place before? Which images were taken around the same location?
- **Image retrieval**: Have I seen this image before? Which images in my database look similar to it? E.g., Google Reverse Image Search



Place Recognition/Image Retrieval

Query image Results on a database of 100 million images



How much is 100 million images?

If each sheet of paper was 0.1 mm thick…

Slide Credit: Nister

Slide Credit: Nister

Slide Credit: Nister

Fast visual search

How do we query an image in a database of 100 million images in less than 6 seconds?

"Video Google", Sivic and Zisserman, ICCV 2003

"Scalable Recognition with a Vocabulary Tree", Nister and Stewenius, CVPR 2006.

Visual Place Recognition

- **Goal**: **query** an image in a database of $\boldsymbol{N}$ **images**
- **Complexity:** NM^2 feature comparisons (assumes each image has ***M*** features)
  - Example:
    - assume 1,000 SIFT features per image → $\mathrm{M} = 1{,}000$
    - assume $N = 100{,}000{,}000$
    - → NM^2= $100{,}000{,}000{,}000{,}000$ feature comparisons!
    - If we assume 0.1 ms per feature comparison → 1 image query would take **317 years**!

Solution: Use an inverted file index!
Complexity reduces to $\boldsymbol{O(M)}$

[“Video Google”, Sivic & Zisserman, ICCV’03]
[“Scalable Recognition with a Vocabulary Tree”, Nister & Stewenius, CVPR’06]
See also FABMAP and Galvez-Lopez’12’s (DBoW2)]

Indexing local features: inverted file text

- For text documents, an efficient way to find all *pages* in which a *word* occurs is to use an index

- We want to find all *images* in which a *feature* occurs

- How many distinct SIFT or BRISK features exist?
  - SIFT → Infinite
  - BRISK-128 → $2^{128} = 3.4 \cdot 10^{38}$

- Since the number of image features may be *infinite,* before we build our visual vocabulary we need to map our features to "*visual words*"

- Using analogies from text retrieval, we should:
  - Define a "Visual Word"
  - Define a "Vocabulary" of Visual Words
  - This approach is known as "Bag of Words" (BOW)

Index

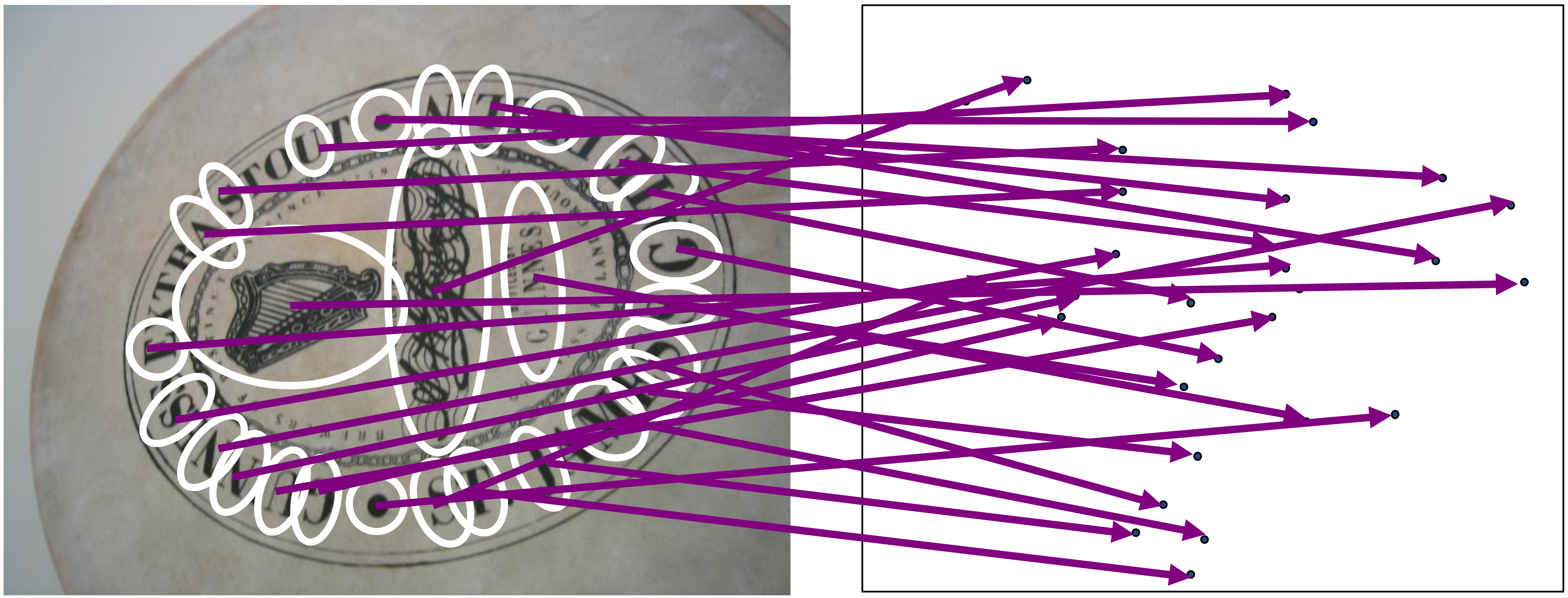
How to extract Visual Words from descriptors

- **Collect a large enough dataset** that is representative of all possible images that are relevant to your application (e.g., for automotive place recognition, you may want to collect million of street images sampled around the world)
- Extract features and descriptors from each image and map them into the **descriptor space** (e.g., for SIFT, 128 dimensional descriptor space)
- **Cluster the descriptor space into K clusters**
- **The centroid of each cluster is a visual word**.
  - This is computed by taking the arithmetic average of all the descriptors within the same cluster:
    - e.g., for SIFT, each cluster contains SIFT features that are very similar to each other;
    - the visual word then is the average all the SIFT descriptors in that cluster

Let’s see an example…

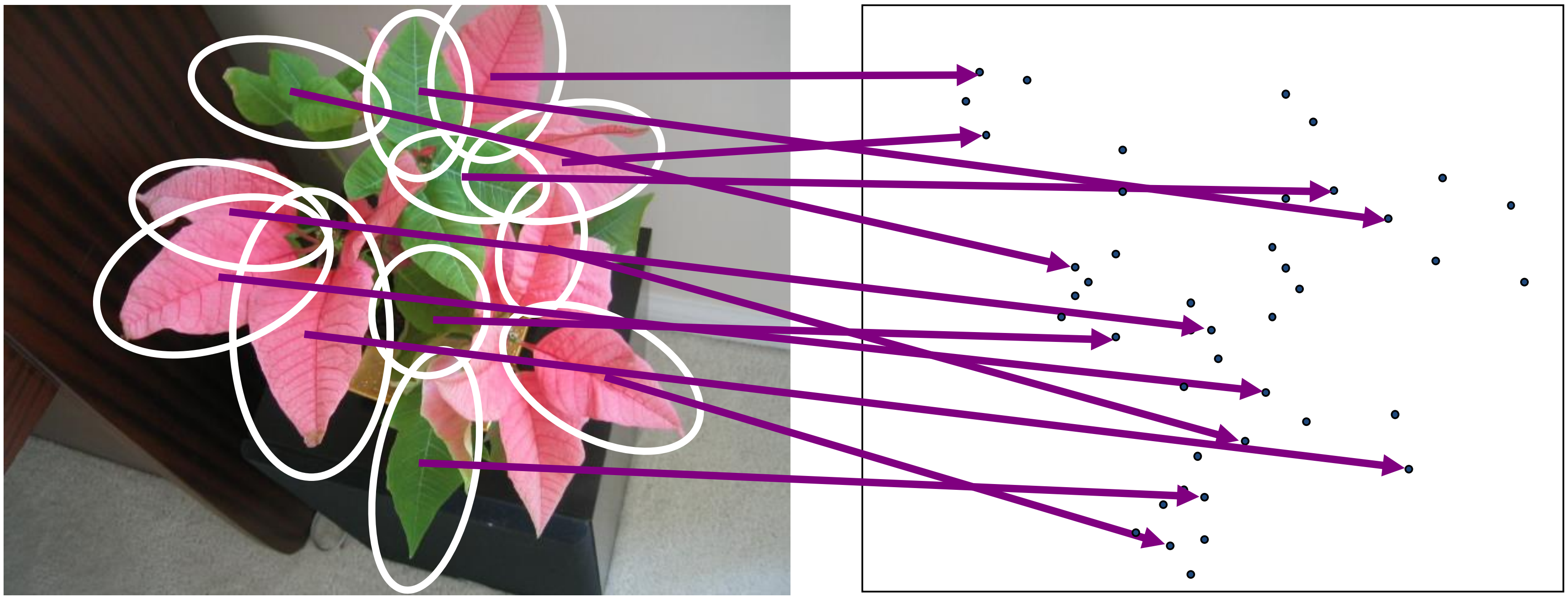
Extracting and mapping descriptors into the descriptor space

Feature descriptor space



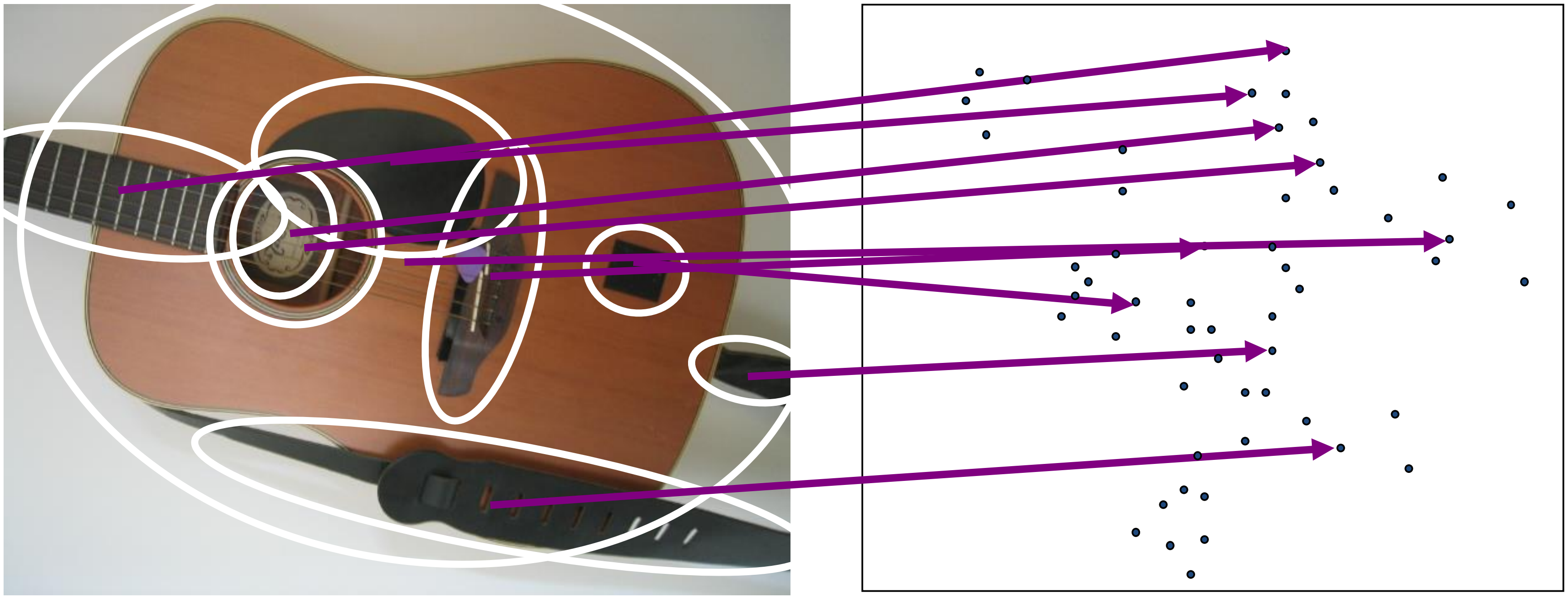
For convenience we assume that the descriptor space has 2 dimensions.
In the case of SIFT, the descriptor space would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



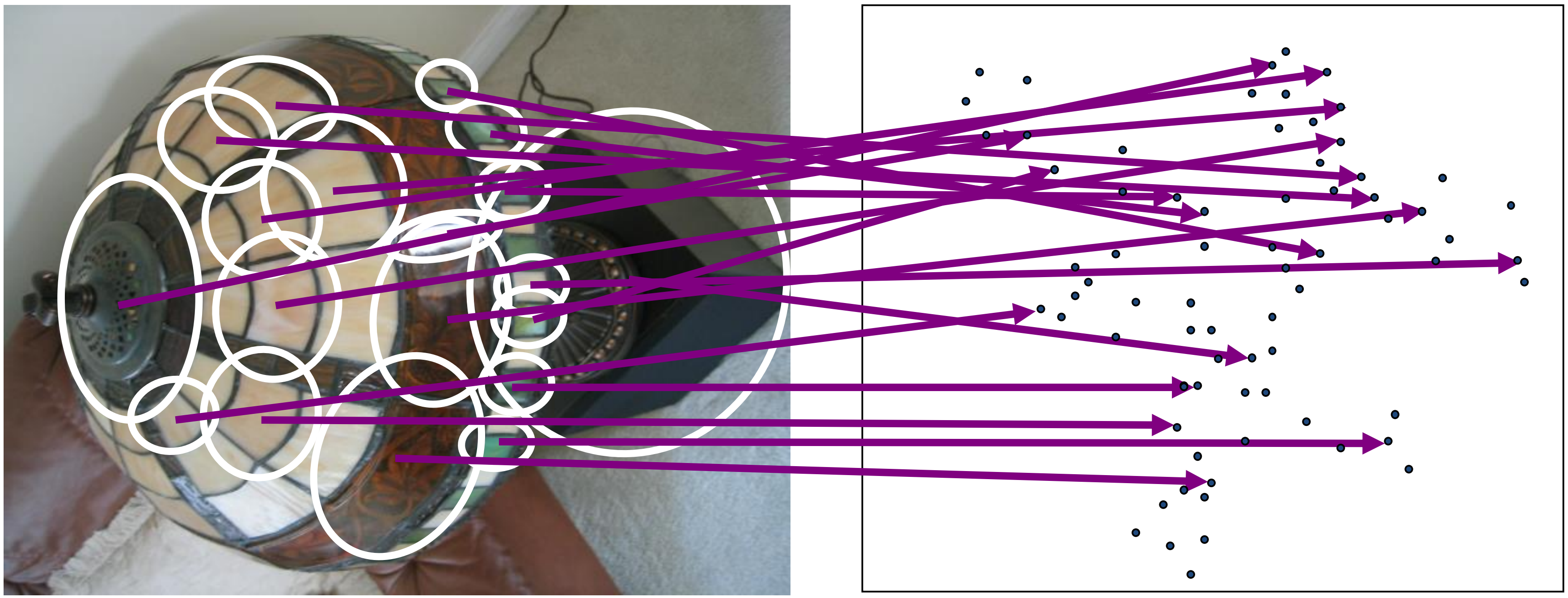
For convenience we assume that the
descriptor space has 2 dimensions.
In the case of SIFT, the descriptor space
would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



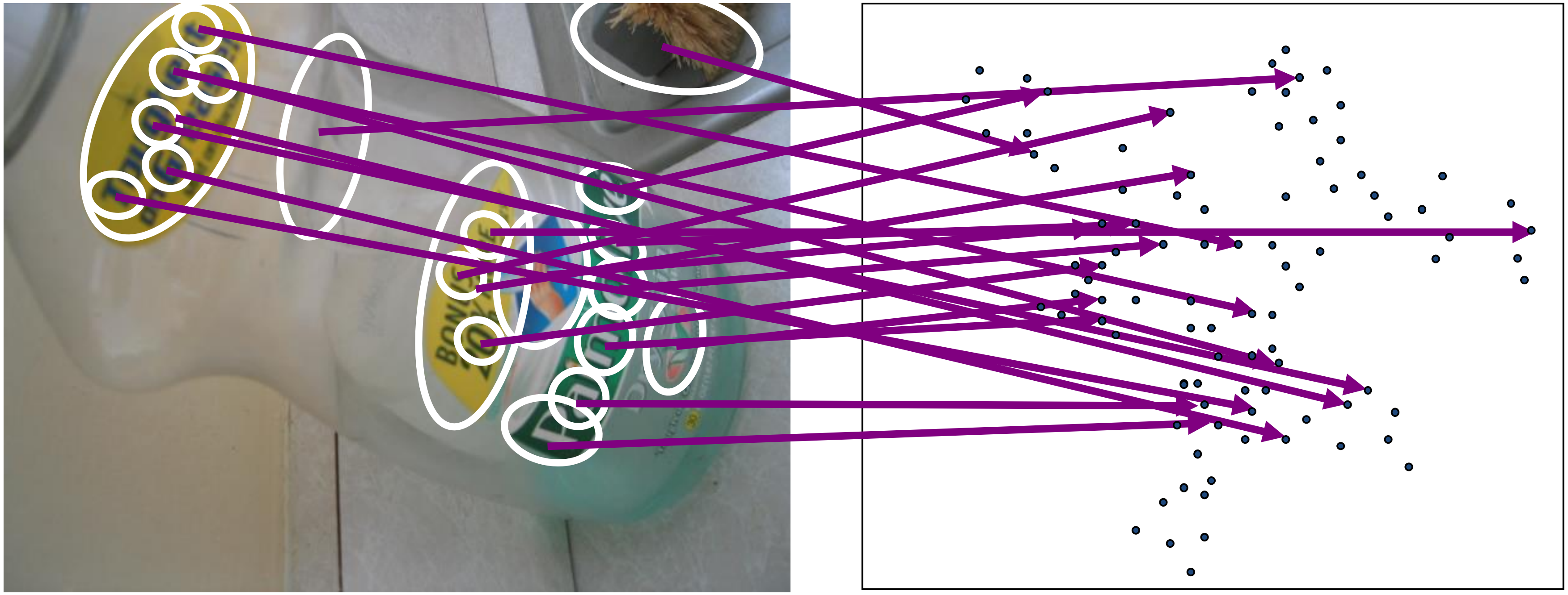
For convenience we assume that the descriptor space has 2 dimensions.
In the case of SIFT, the descriptor space would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



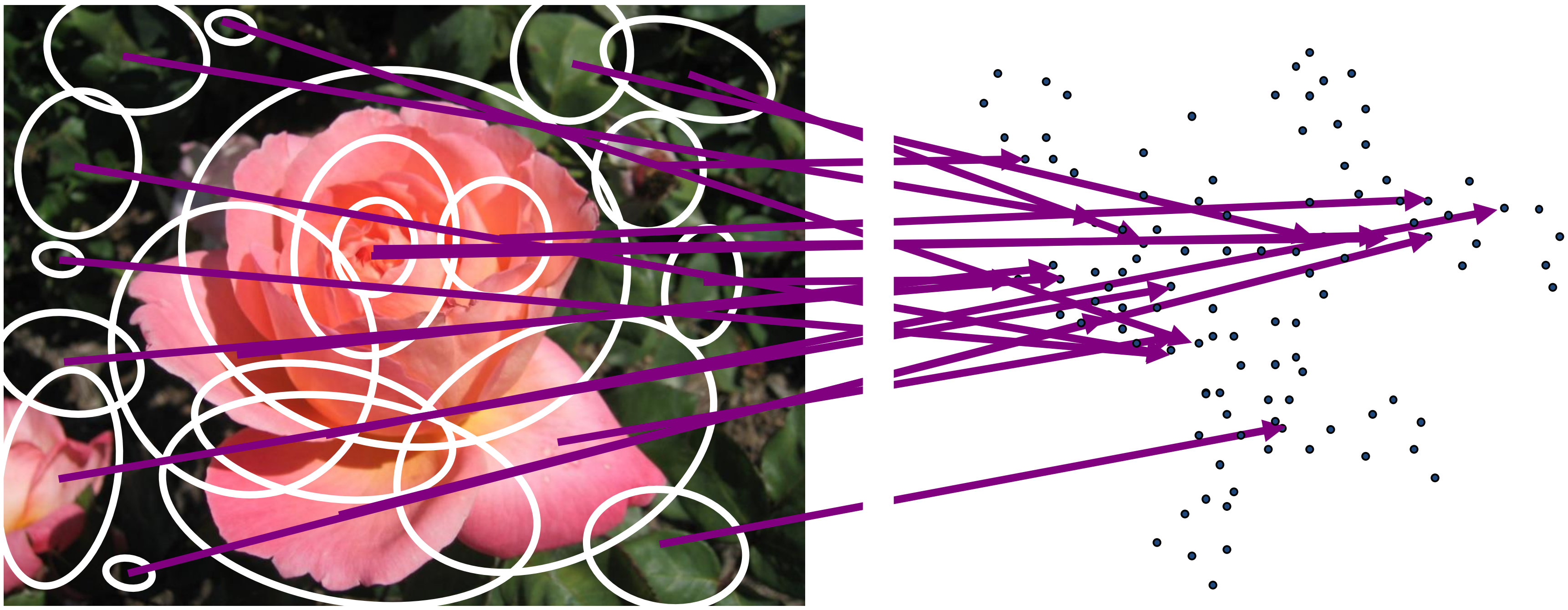
For convenience we assume that the descriptor space has 2 dimensions.
In the case of SIFT, the descriptor space would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space

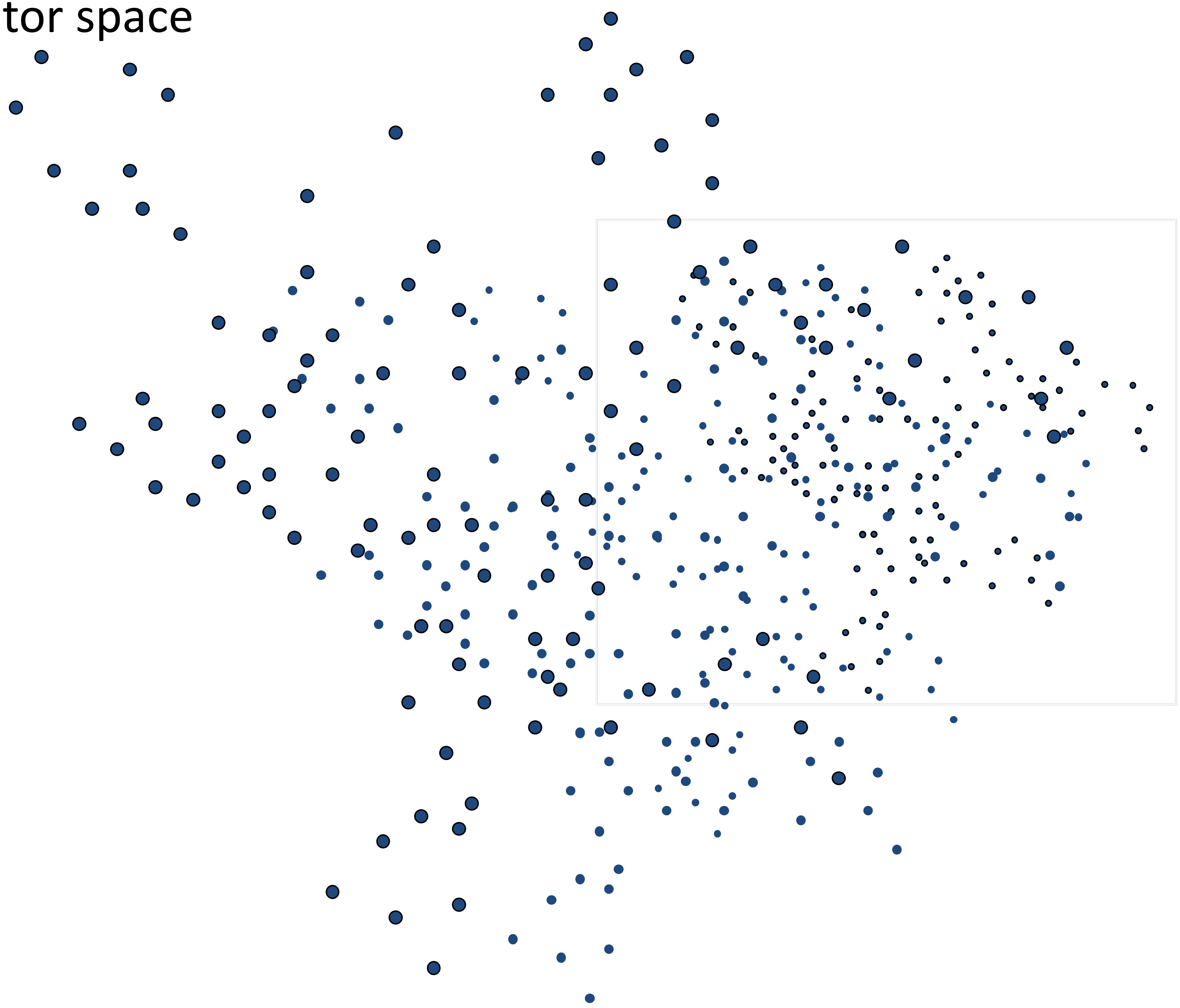


For convenience we assume that the descriptor space has 2 dimensions.
In the case of SIFT, the descriptor space would have 128 dimensions!

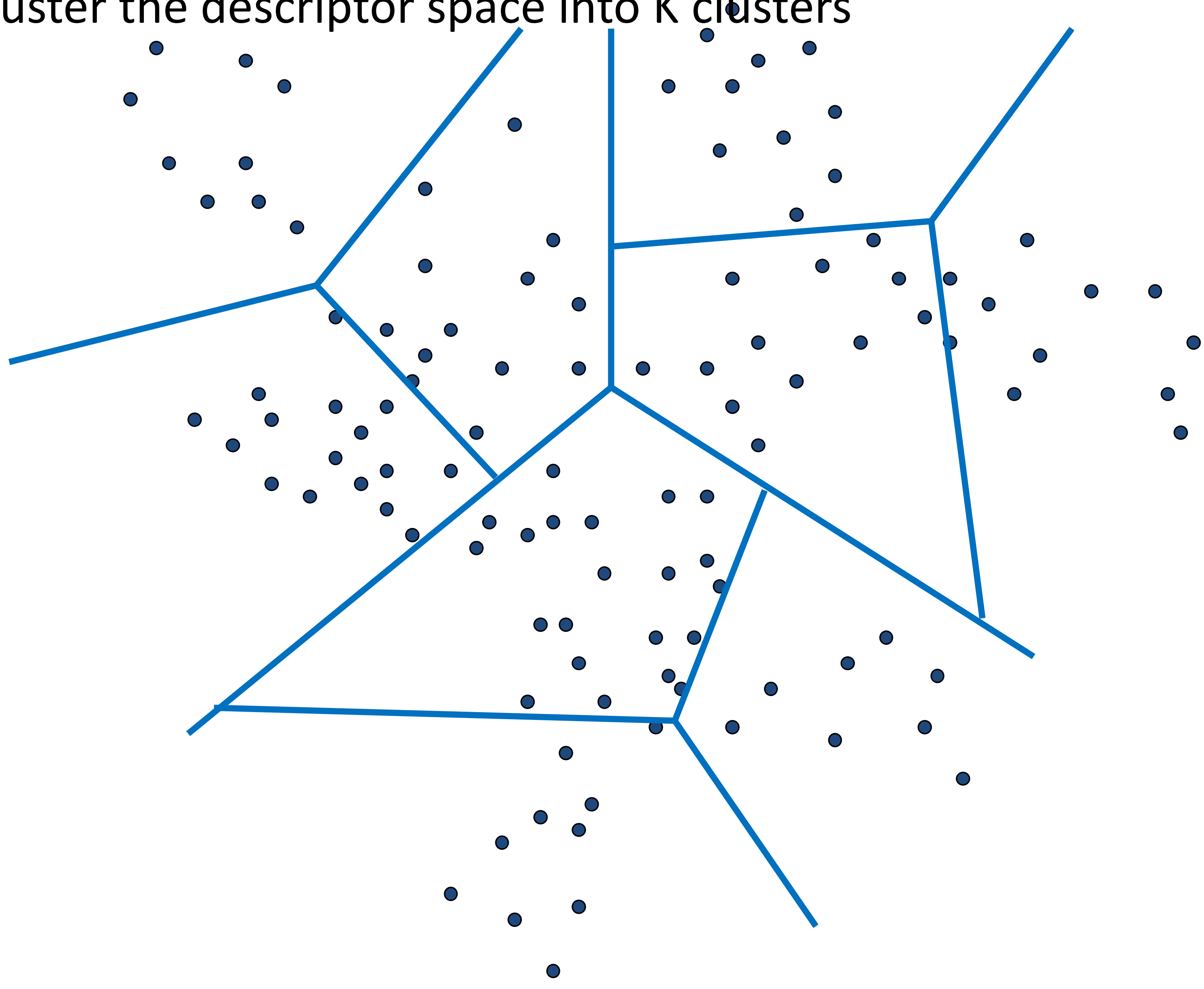
Extracting and mapping descriptors into the descriptor space



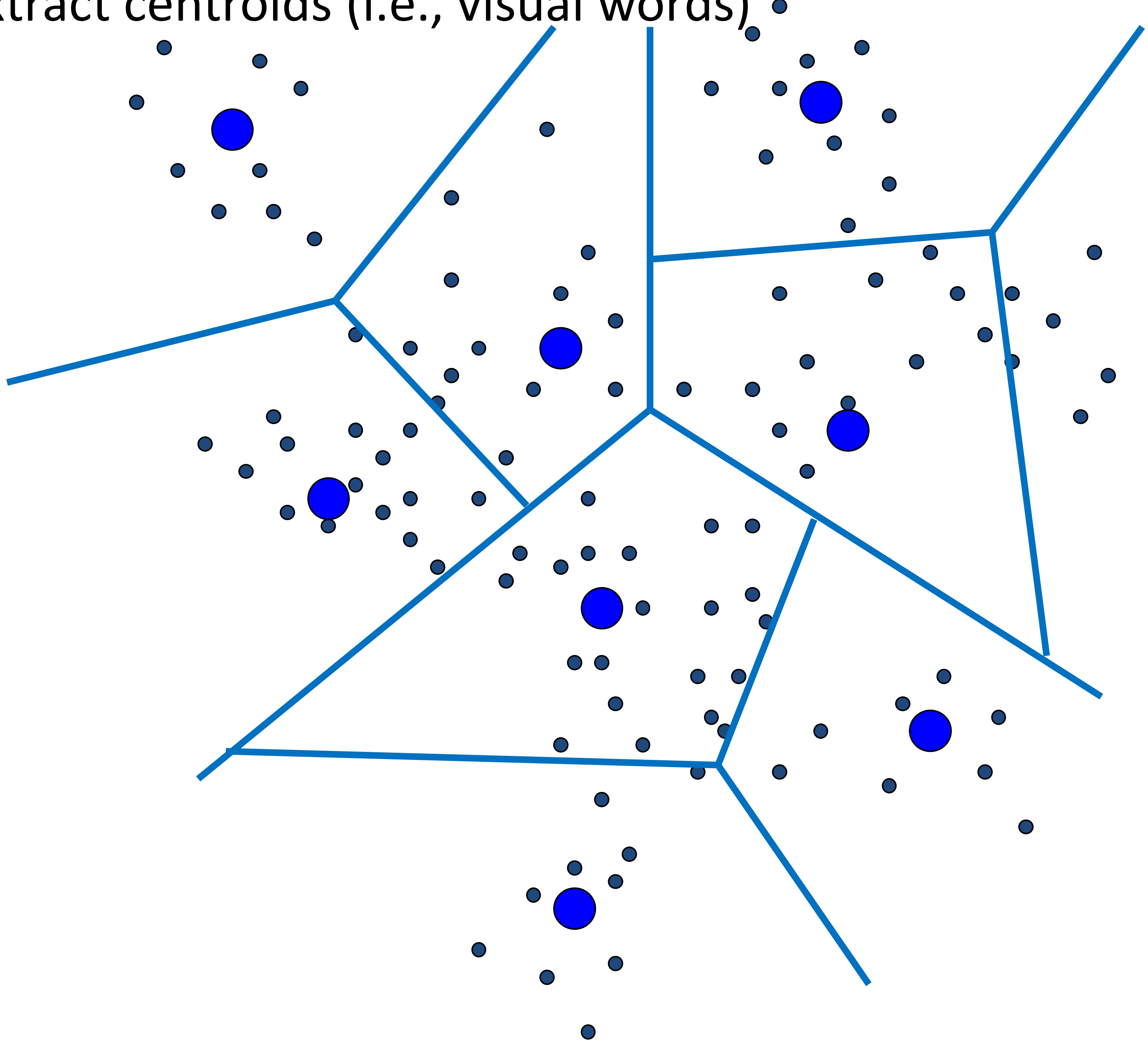
Descriptor space



Cluster the descriptor space into K clusters



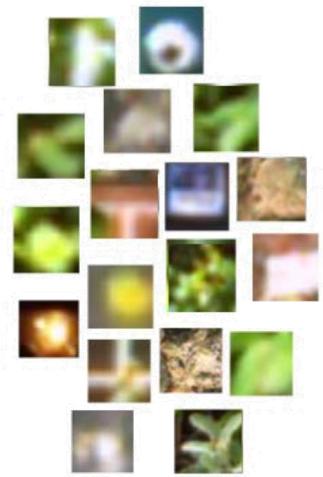
Extract centroids (i.e., visual words)



Summary

Image Collection

Extract Features

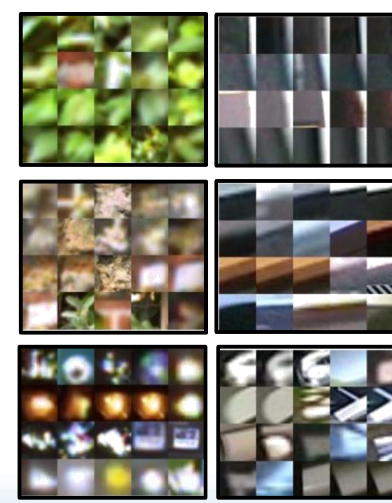


What is a visual word?

A visual word is the **centroid** of a cluster of similar features (i.e., similar descriptors)

Cluster Descriptors

Examples of Features belonging to the same clusters



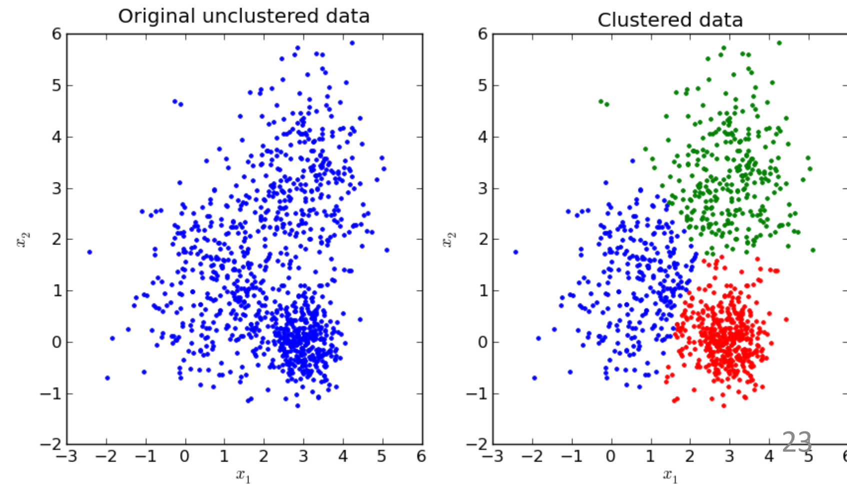
How do we cluster the descriptor space?

- *k-means clustering* is an algorithm to partition n data point into k clusters in which each data point $\boldsymbol{x}$ belongs to the cluster $\boldsymbol{S_i}$ with center $\boldsymbol{m}_i$
- It minimizes the sum of squared Euclidean distances between points $\boldsymbol{x}$ and their nearest cluster centers $\boldsymbol{m}_i$

$$D(X,M) = \sum_{i=1}^{k} \sum_{x \in S_i} (x - m_i)^2$$

Algorithm:

- Randomly initialize k cluster centers
- Iterate until convergence:
  - Assign each data point $\boldsymbol{x}_j$ to the nearest center $\boldsymbol{m}_i$
  - Recompute each cluster center as the mean of all points assigned to it

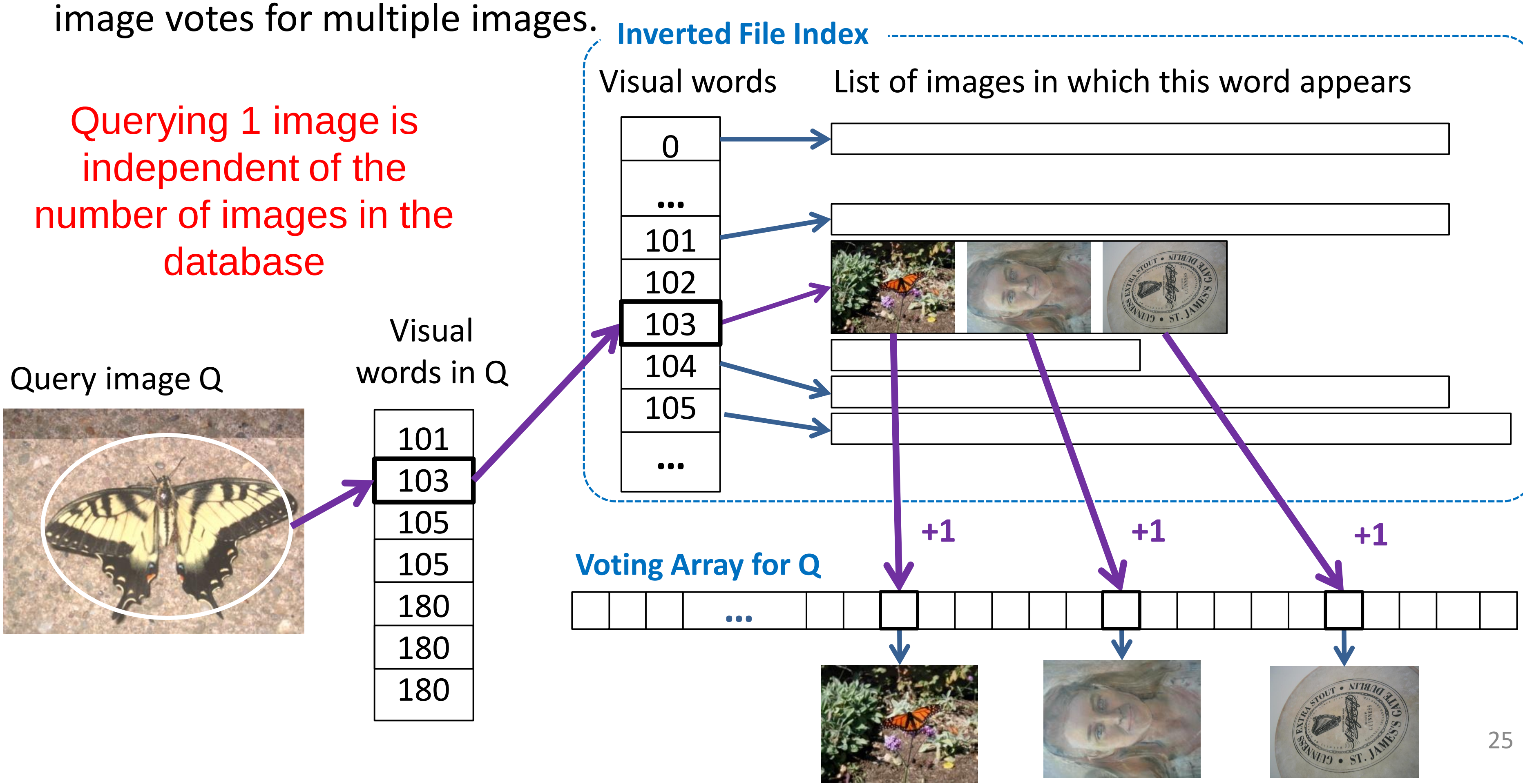


K-means demo

Source: http://shabal.in/visuals/kmeans/1.html

Applying Visual Words to Image Retrieval

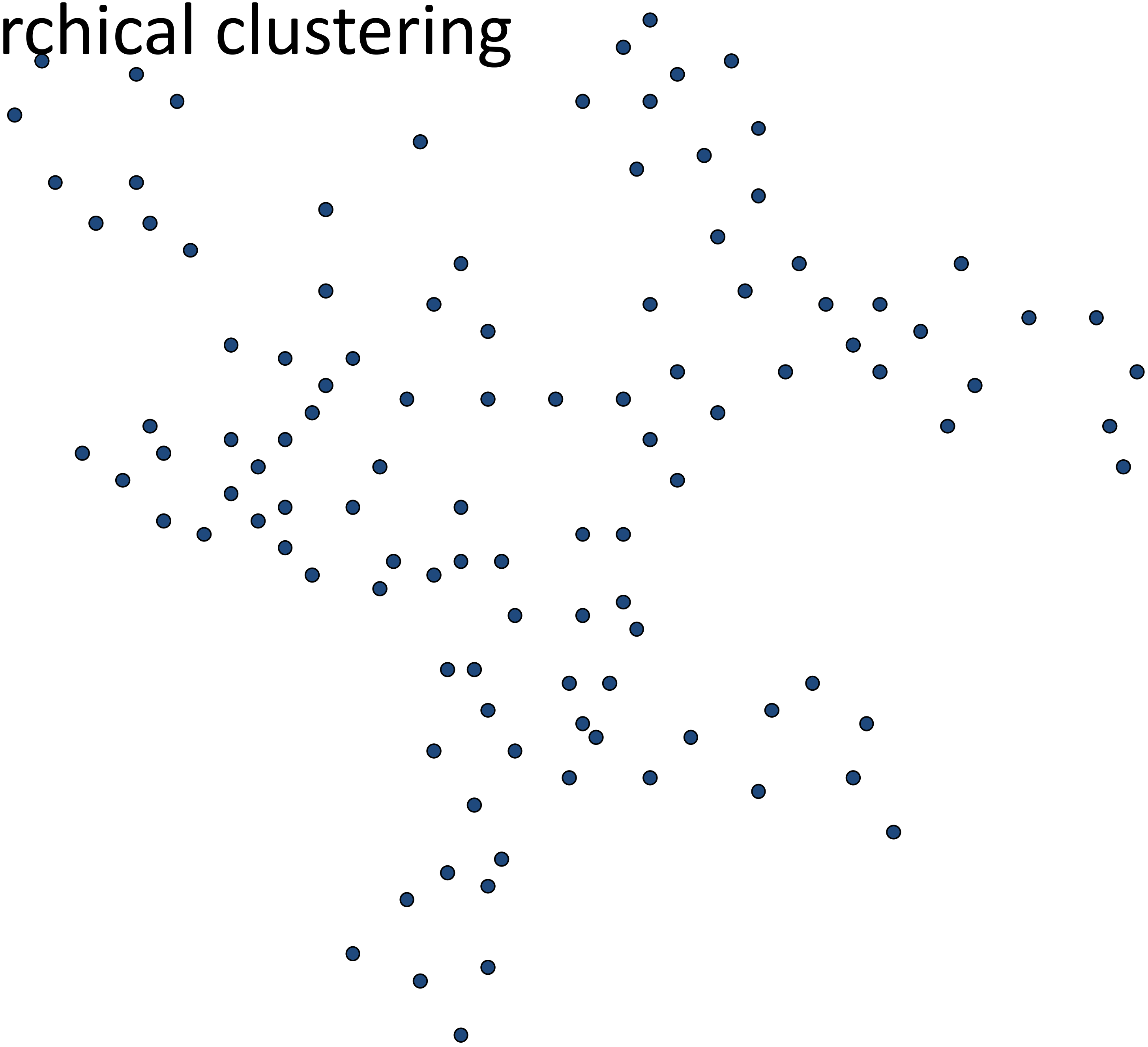
- **Inverted File Index** lists all visual words in the vocabulary (extracted at training time)
- Each word points to a **list of images**, from the entire image Data Base (DB), in which that word appears. The DB grows as the robot navigates and collects new images.
- **Voting array**: has as many cells as the images in the DB. Each word in the query image votes for multiple images.



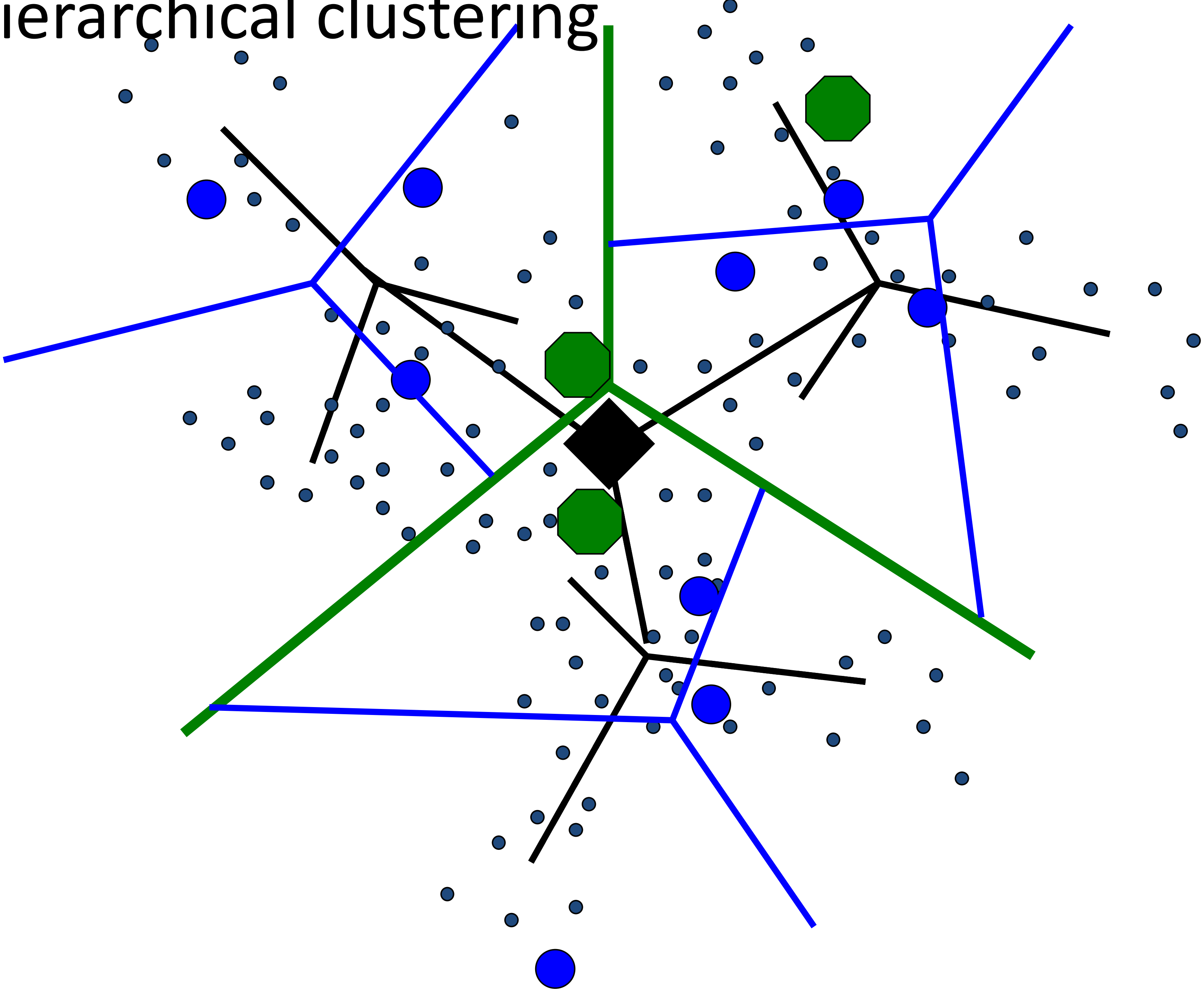
Drawback

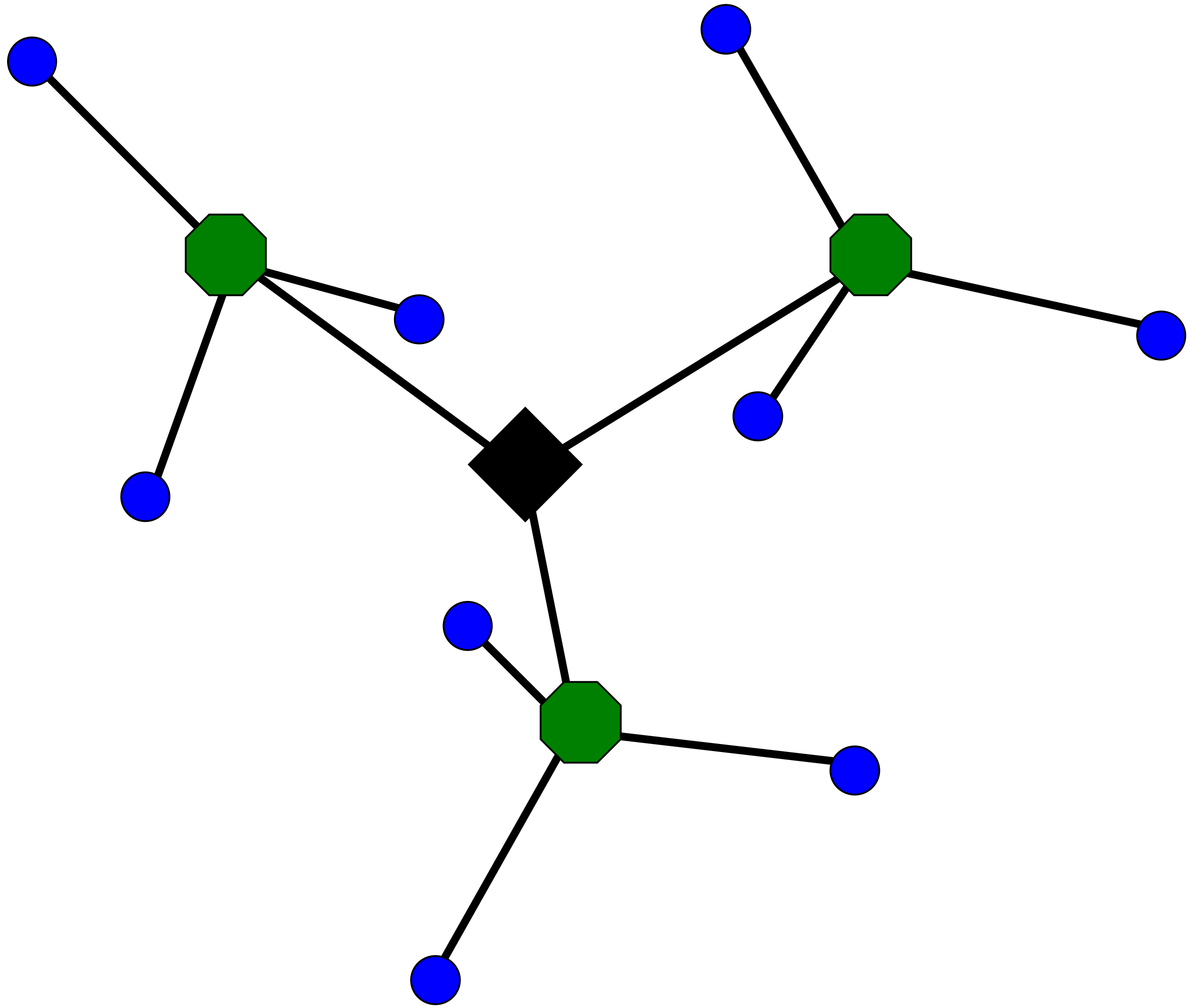
- Every feature in the query image still needs to be compared against all features in the vocabulary:
  - Example:
    - Assume our query image has 1,000 SIFT features → $\mathrm{M} = 1{,}000$
    - assume $1{,}000{,}000$ visual words
    - → Number of feature comparisons $= 1{,}000{,}000{,}000$
    - If we assume 0.1 ms per feature comparison → 1 image query would take **28 hours**!
- How can we make the comparison cheaper, e.g., less than 6 seconds?
  - Solution: use hierarchical clustering: "Scalable Recognition with a Vocabulary Tree", [Nister & Stewenius, CVPR'06]

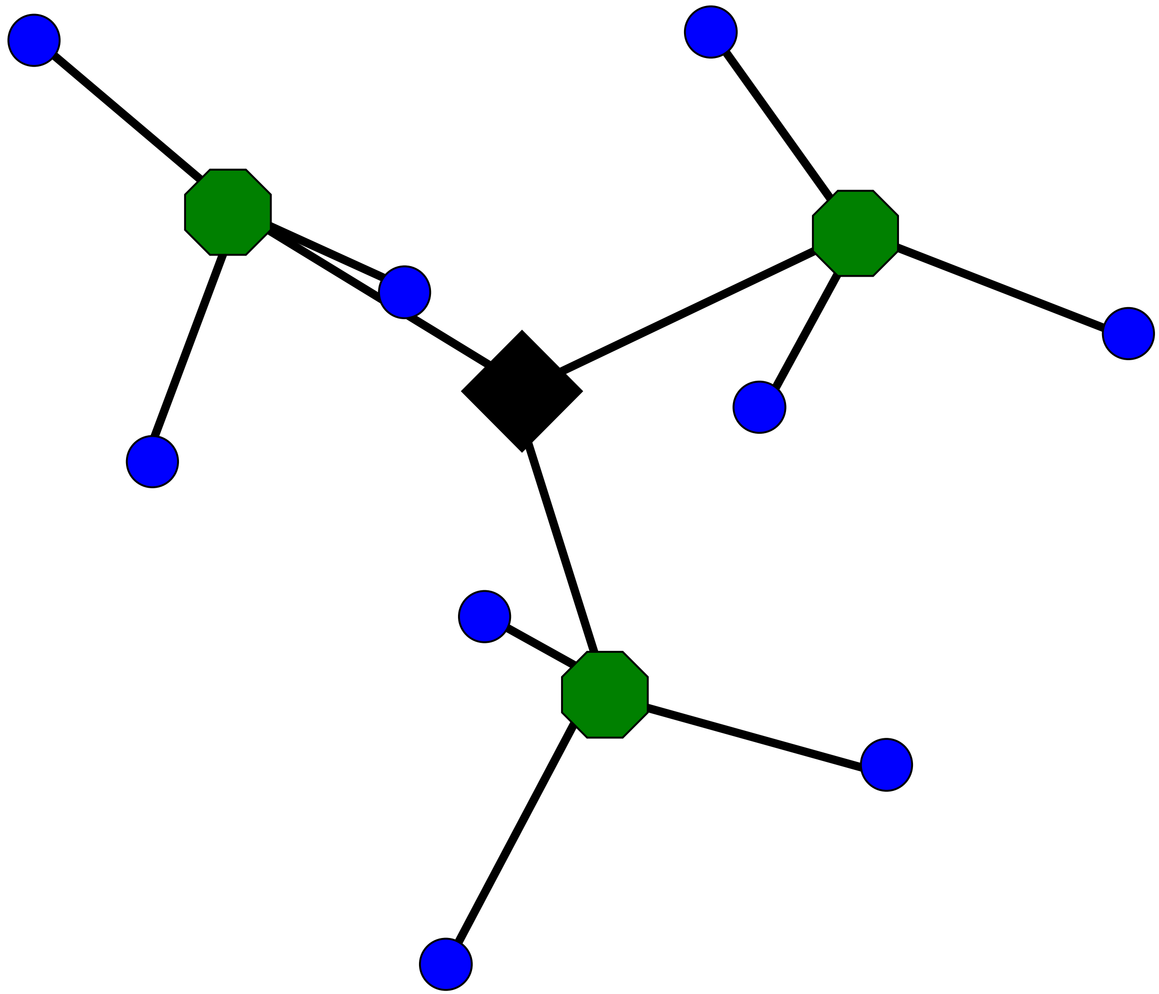
Hierarchical clustering

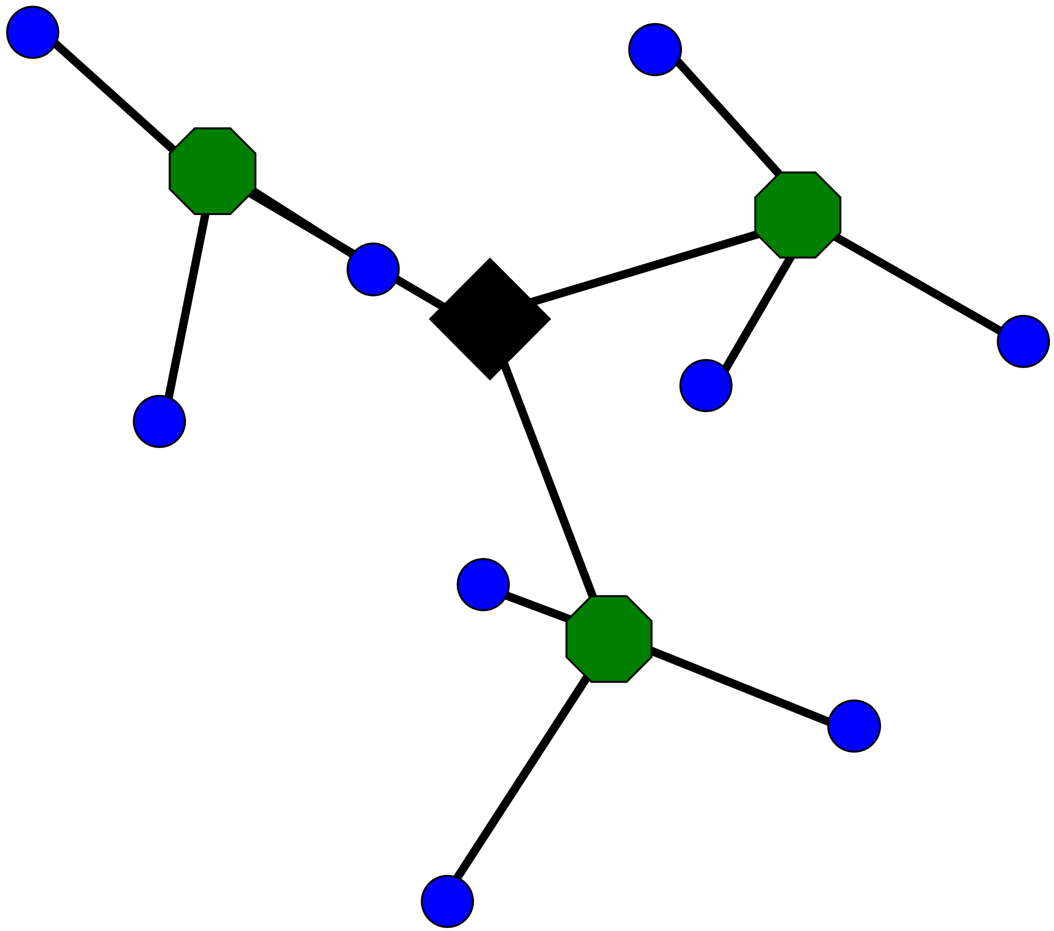


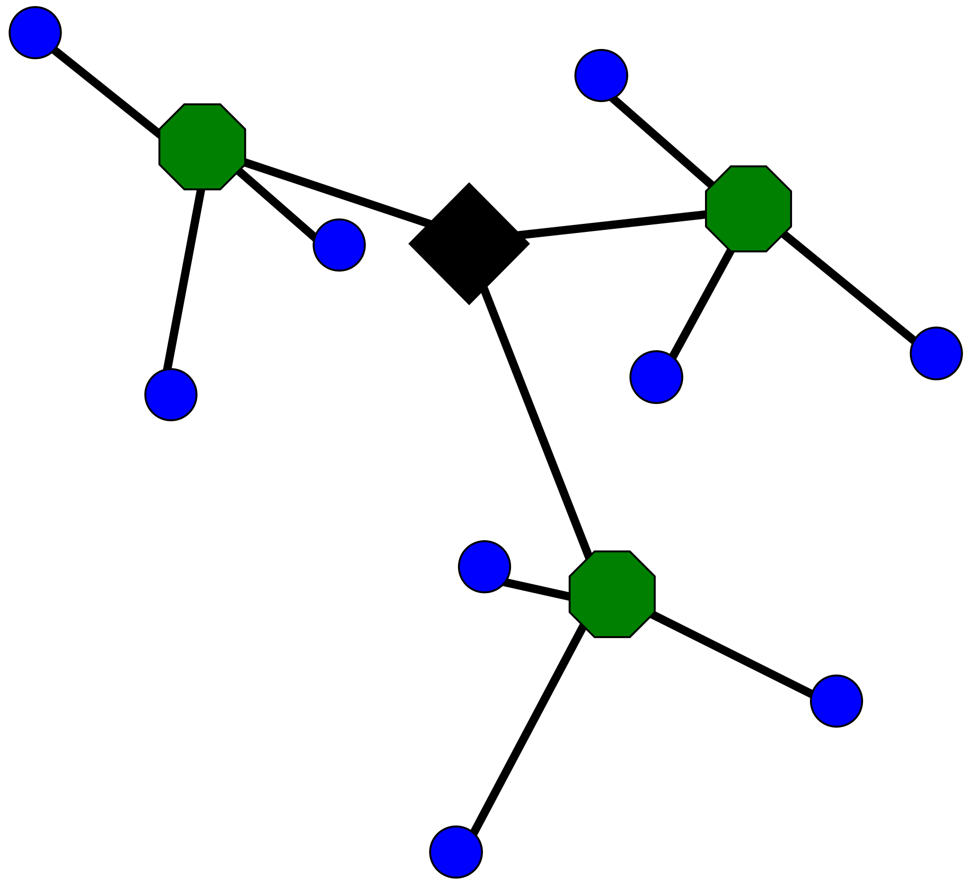
Hierarchical clustering

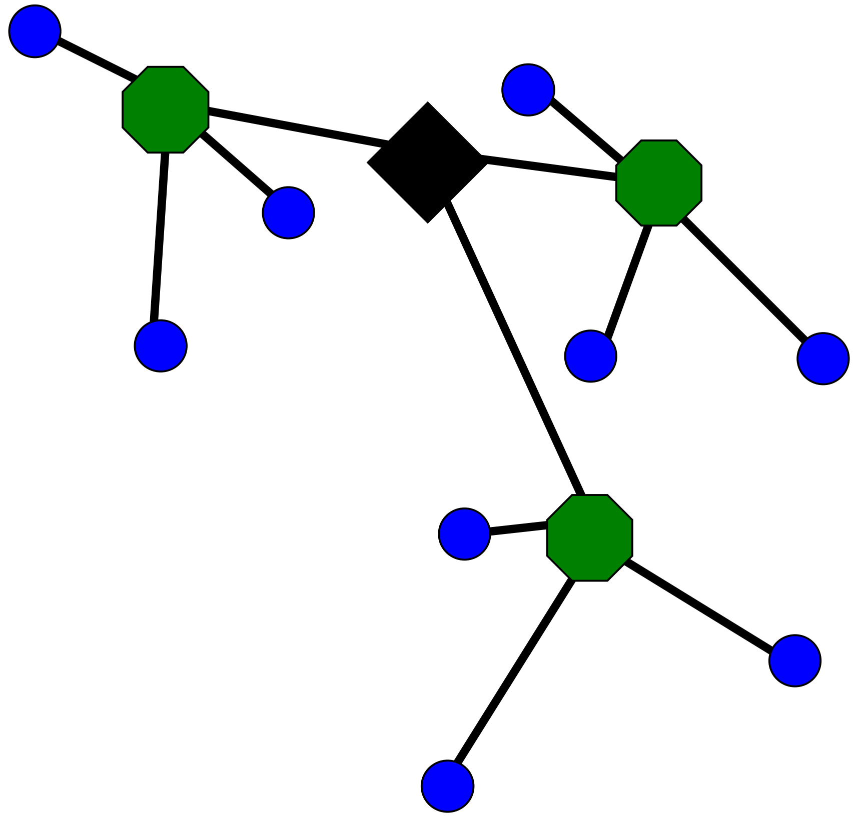


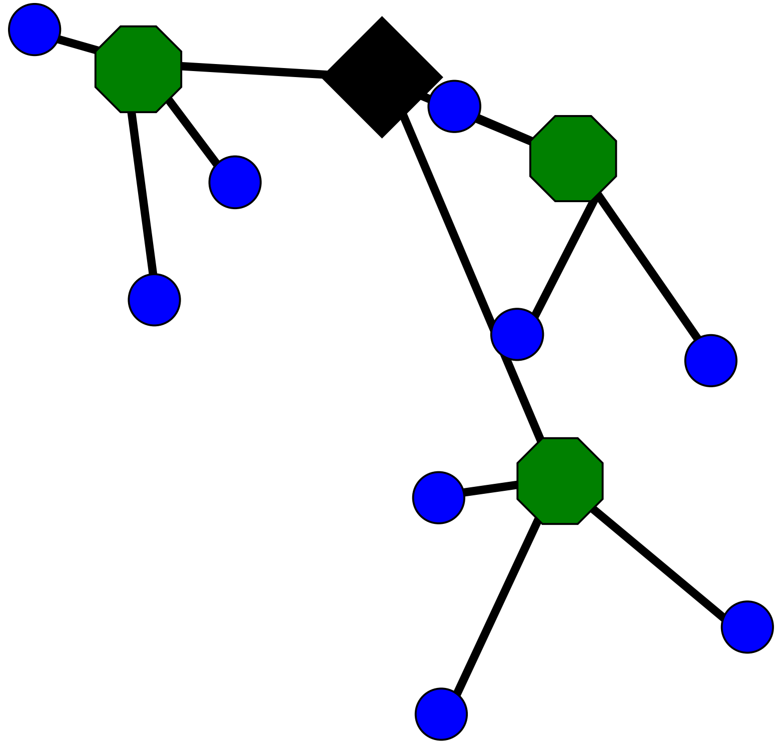


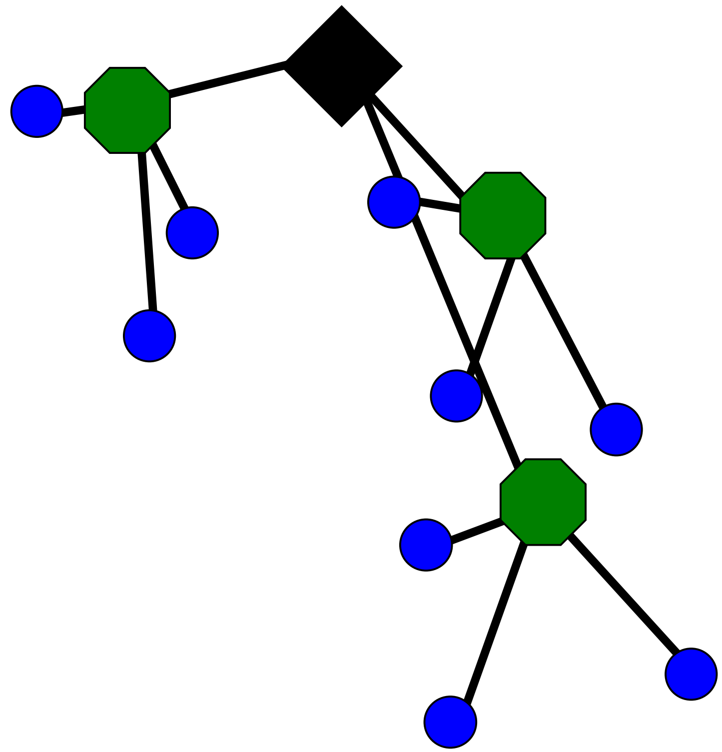


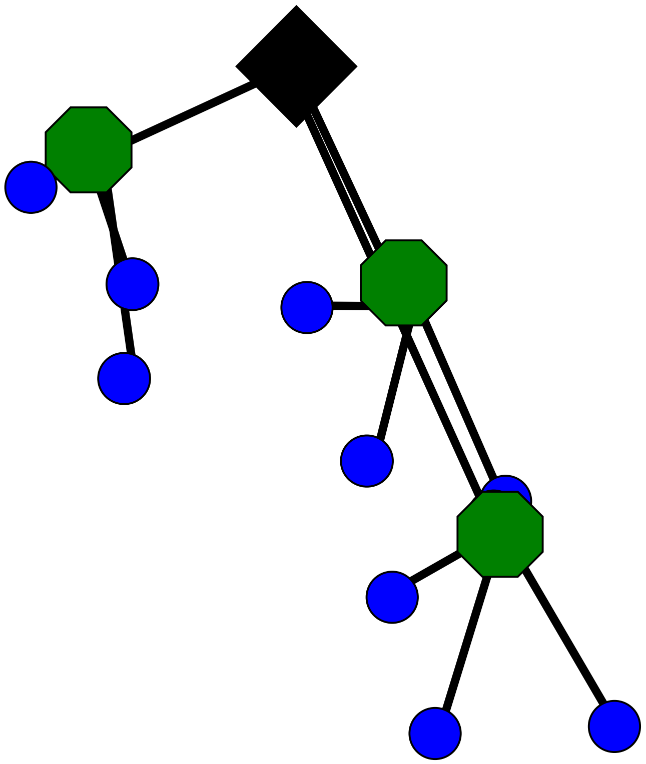


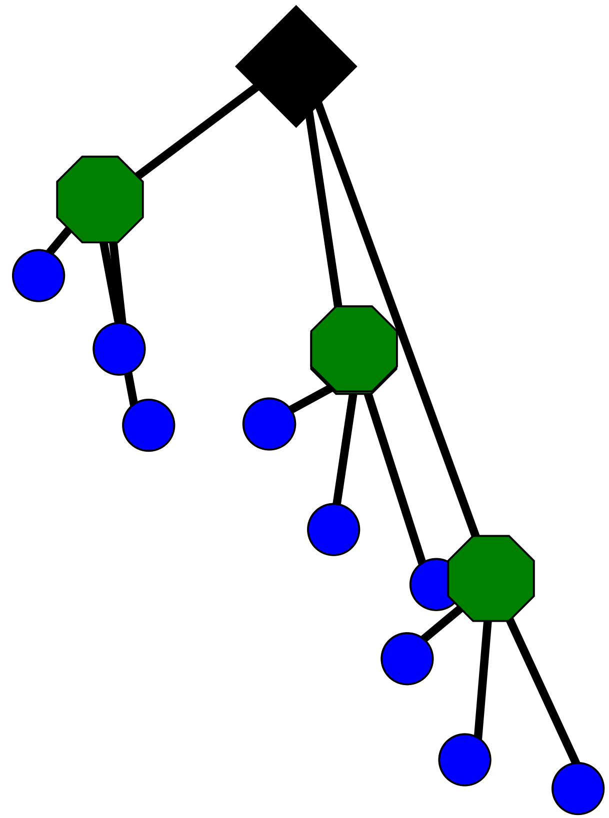


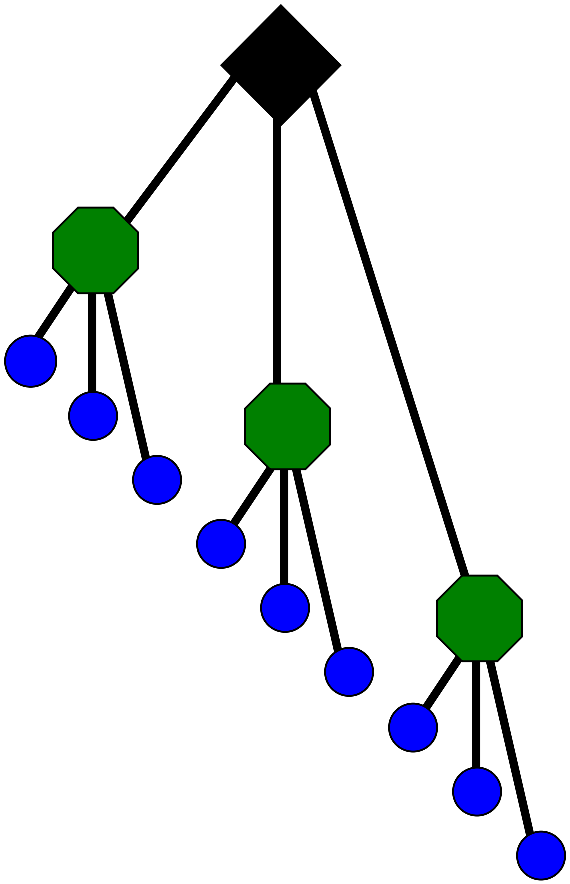




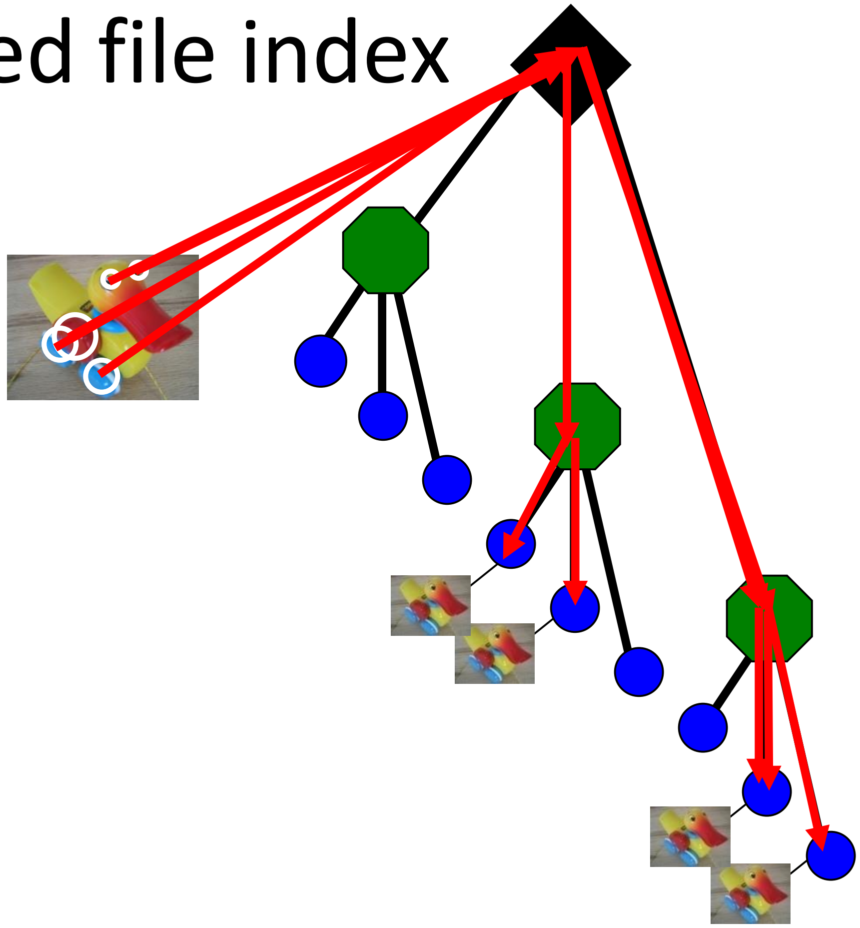




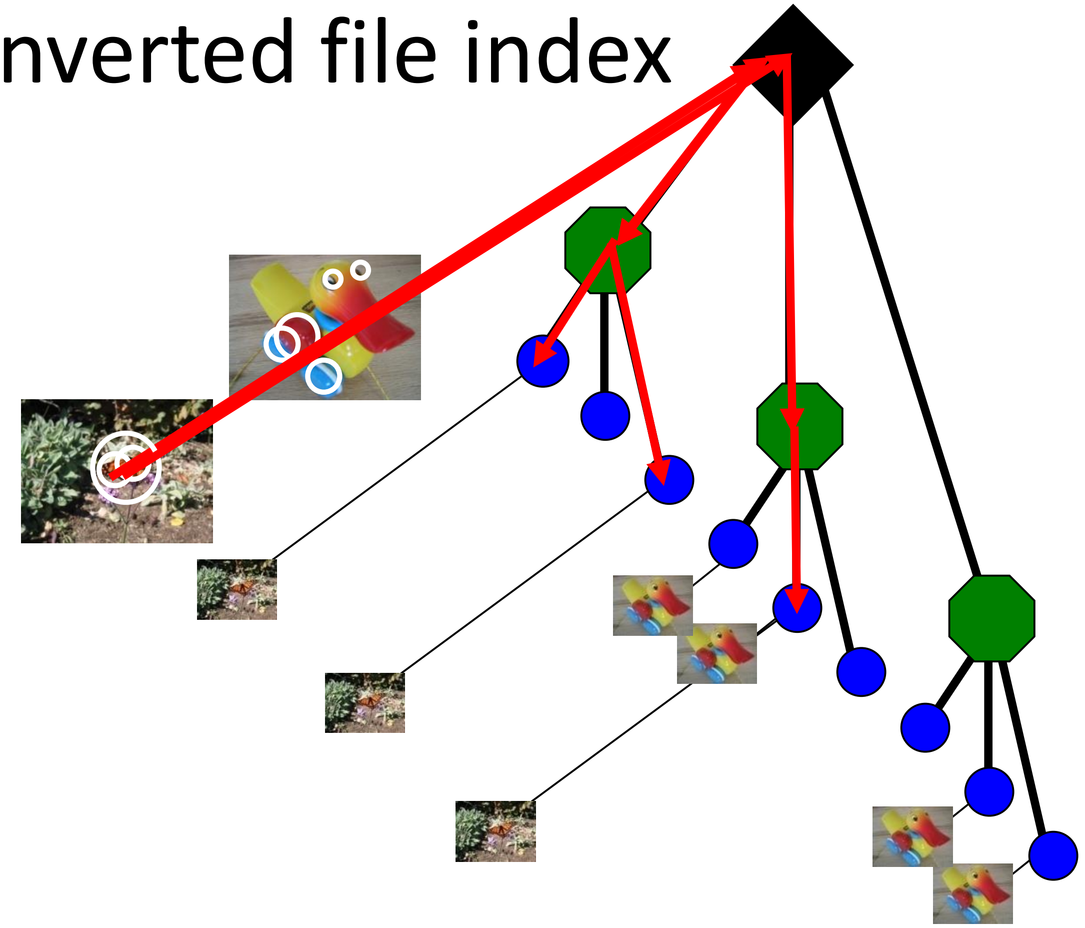




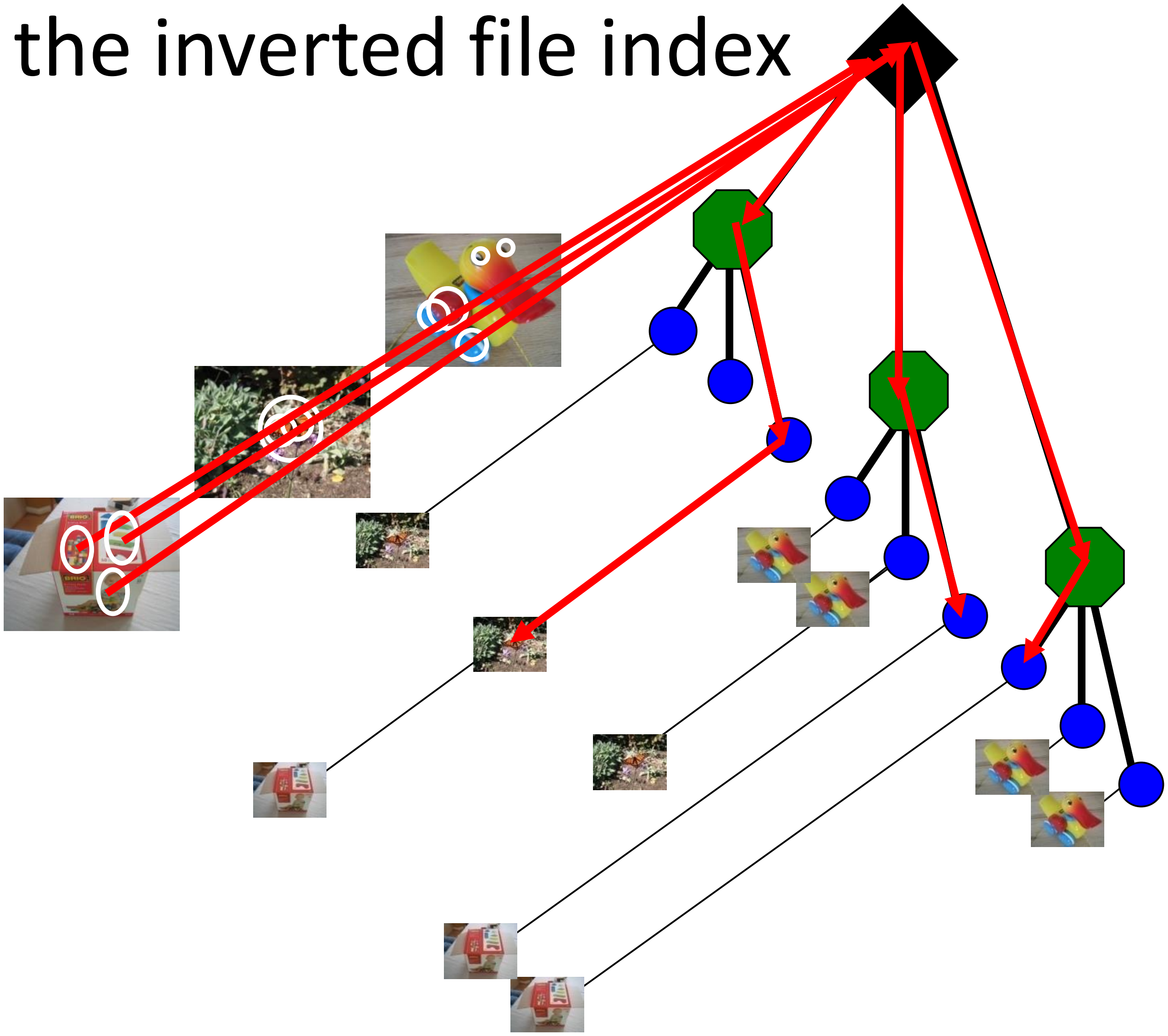
Building the inverted file index



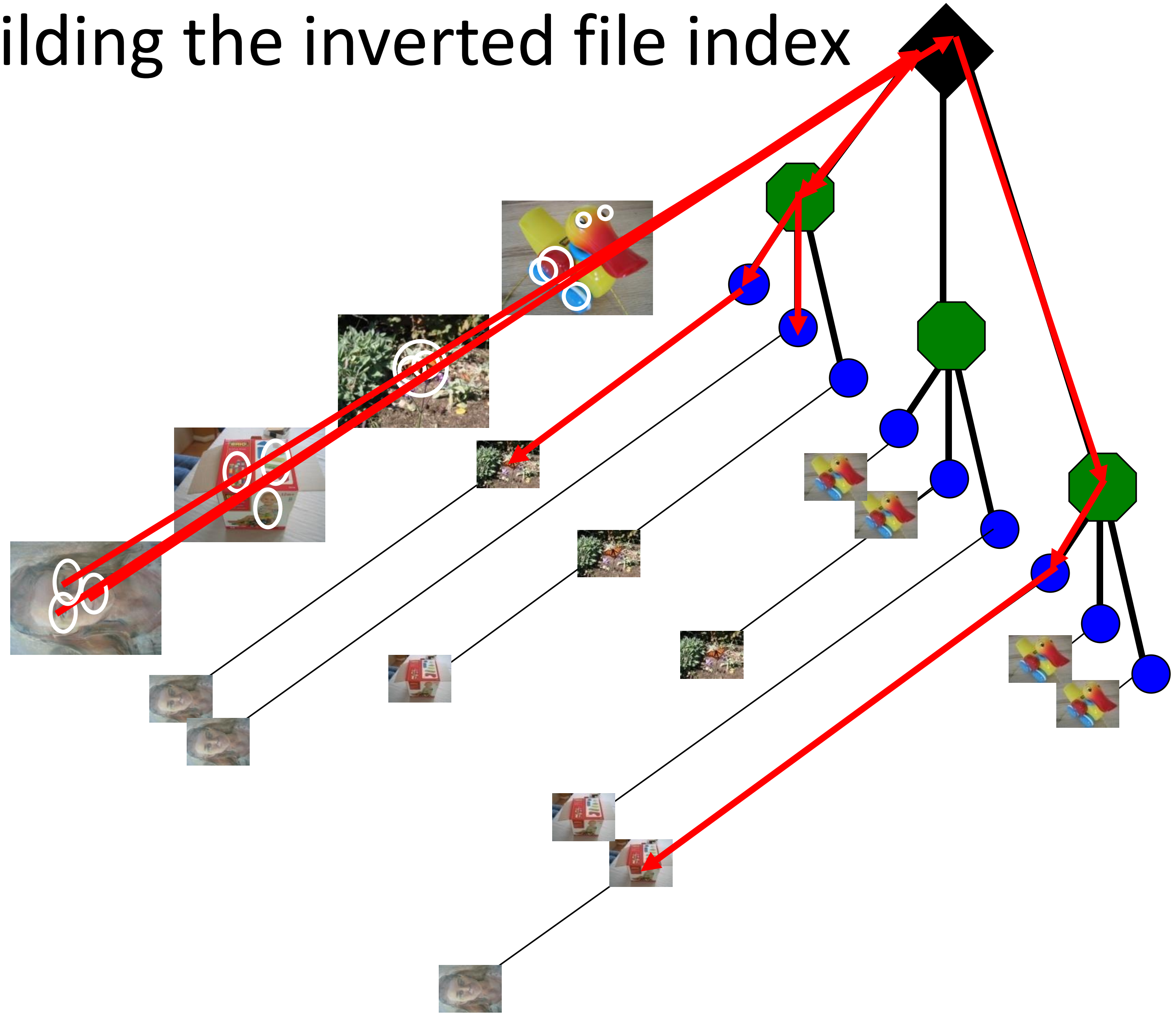
Building the inverted file index



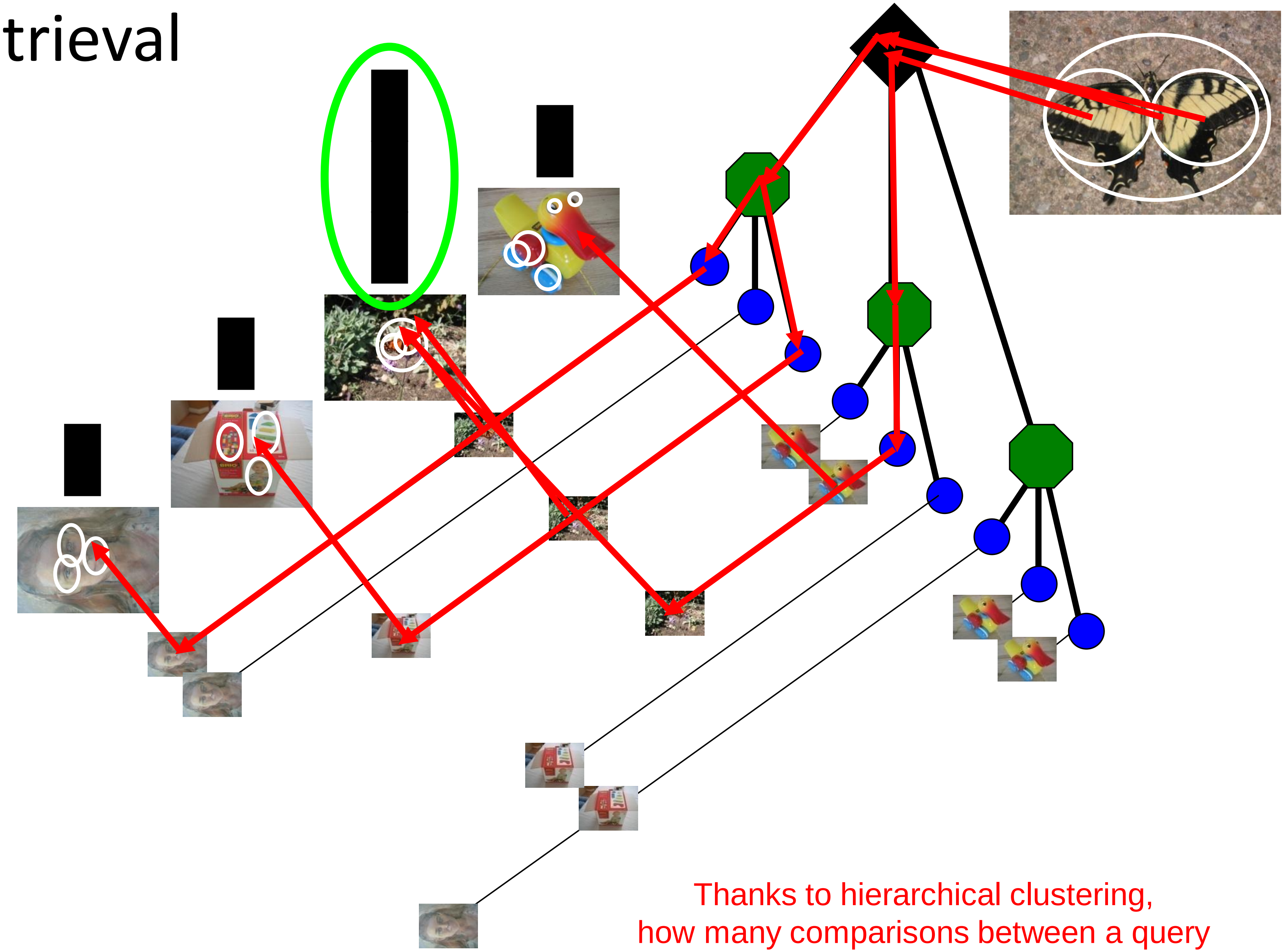
Building the inverted file index



Building the inverted file index



Retrieval



Thanks to hierarchical clustering,
how many comparisons between a query
feature and the visual words need to
be done with B branches and L depth levels?

Example

Querying an image in a database of **100 million images**

- assume a query image with 1,000 SIFT features → $\mathrm{M} = 1{,}000$
- assume 10 branches and 6 depth levels (i.e., $b^L = 1{,}000{,}000$ visual words)

  → Number of feature comparisons $= \mathrm{M} \cdot b \cdot L = 1{,}000 \cdot 10 \cdot 6 = 60{,}000$
- If we assume 0.1 ms per feature comparison → 1 image query would take **6 seconds**!

To conclude, for M features in the Query image, only $\mathrm{M} \cdot b \cdot L$ comparisons need to be made instead of $\mathrm{M} \cdot b^L$

Robust object/scene recognition

- Visual Vocabulary discards the spatial relationships between features
  - Two images with the same features *shuffled around* will return a 100% match when using only appearance information.
- This can be overcome using **geometric verification**
  - Test the h most similar images to the query image for geometric consistency (e.g. using 5- or 8-point RANSAC) and retain the image with the smallest reprojection error and largest number of inliers
  - Further reading (out of scope of this course):
    - [Cummins and Newman, IJRR 2011]
    - [Stewénius et al, ECCV 2012]

Video Google System

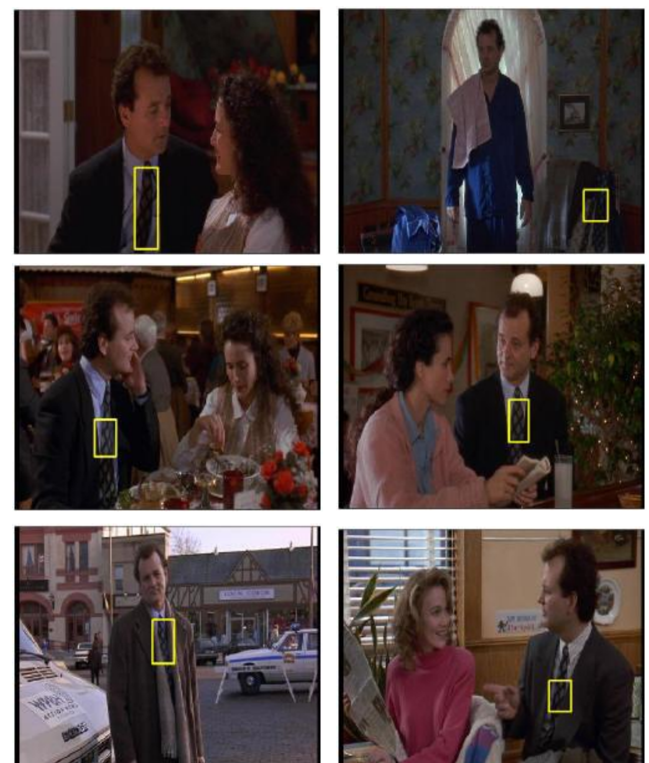
1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

Sivic & Zisserman, ICCV 2003

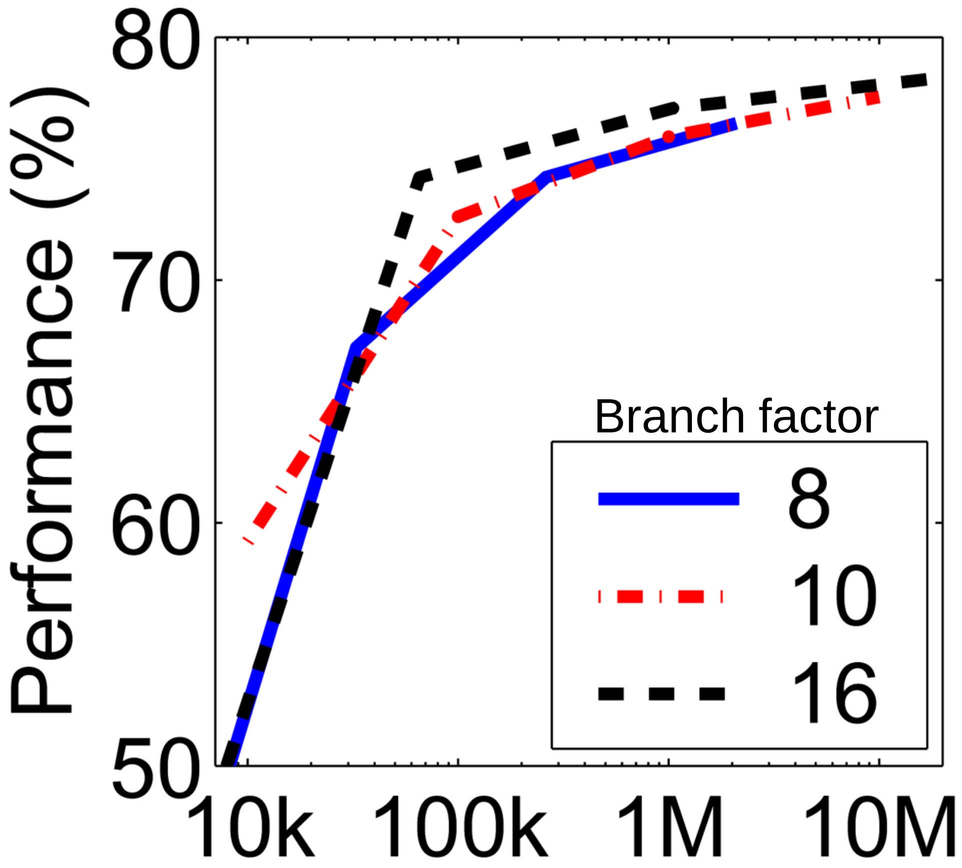
- Demo online at : http://www.robots.ox.ac.uk/~vgg/research/vgoogle/

Query region

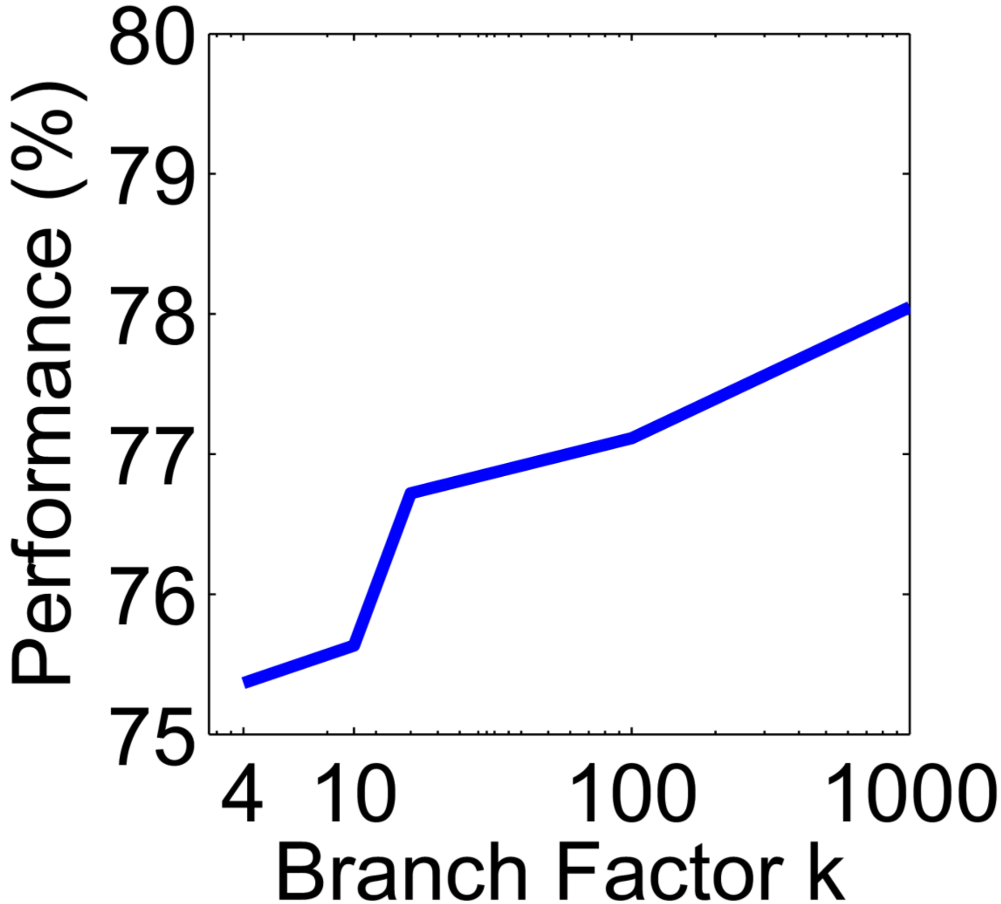
Retrieved frames



More words is better



Higher branch factor works better (but slower)

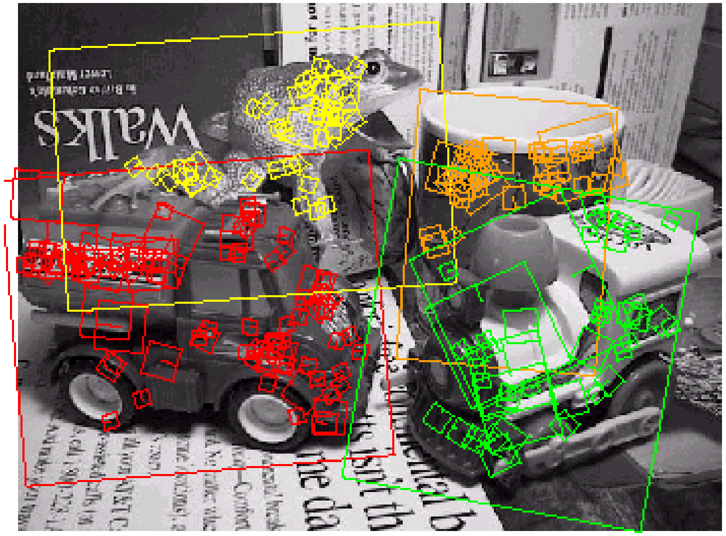


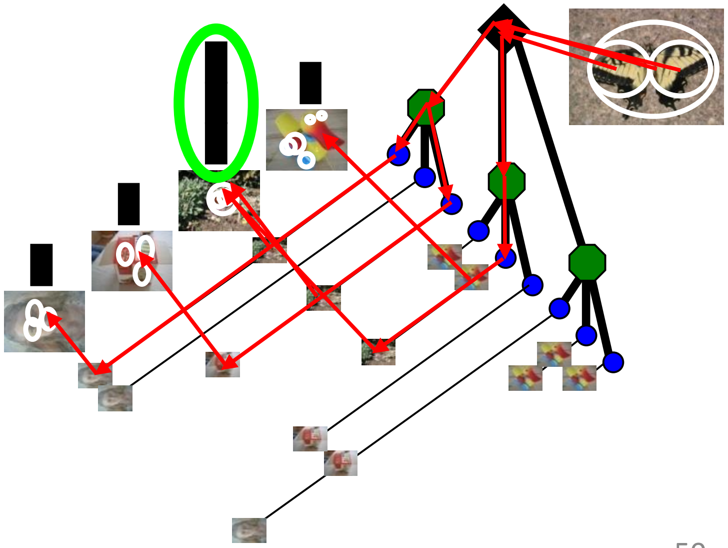
FABMAP [Cummins and Newman IJRR 2011]

- Place recognition for robot localization
- Uses training images to build the BoW database
- **Captures the spatial dependencies of visual words** to distinguish the most characteristic structure of each scene
- Probabilistic model of the world. At a new frame, compute:
  - P(being at a known place)
  - P(being at a new place)
- Very high performance
- Binaries available online
- Open FABMAP

Things to remember

- K-means clustering
- Bag of Words approach
  - What is visual word
  - Inverted file index
  - How it works
- **Chapter 14 of the Szeliski's book**





Understanding Check

Are you able to answer the following questions?

- What is an inverted file index?
- What is a visual word?
- How does K-means clustering work?
- Why do we need hierarchical clustering?
- Explain and illustrate image retrieval using Bag of Words.
- Discussion on place recognition: what are the open challenges and what solutions have been proposed?