University of Zurich UZH

ROBOTICS & PERCEPTION GROUP

Vision Algorithms for Mobile Robotics

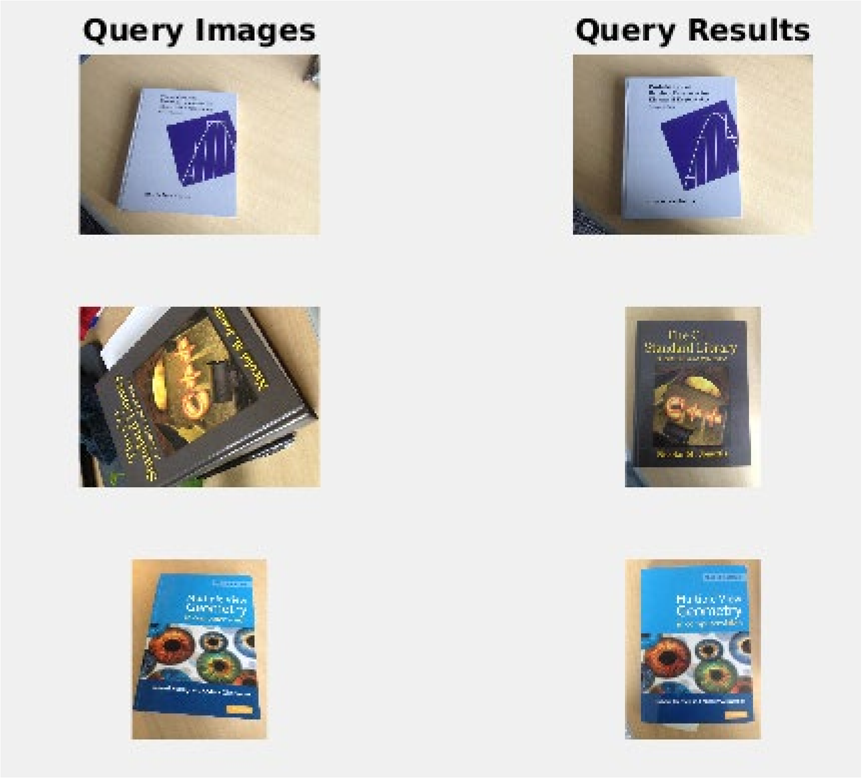
Lecture 12a
Place Recognition

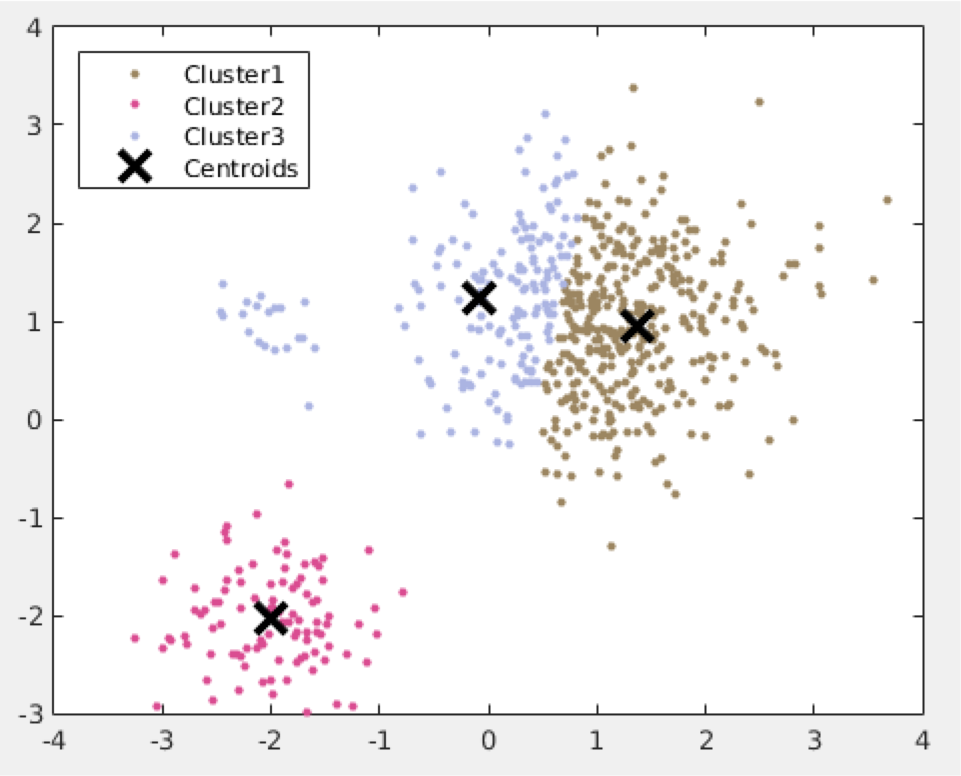
Davide Scaramuzza

http://rpg.ifi.uzh.ch

Deep Learning Tutorial Today

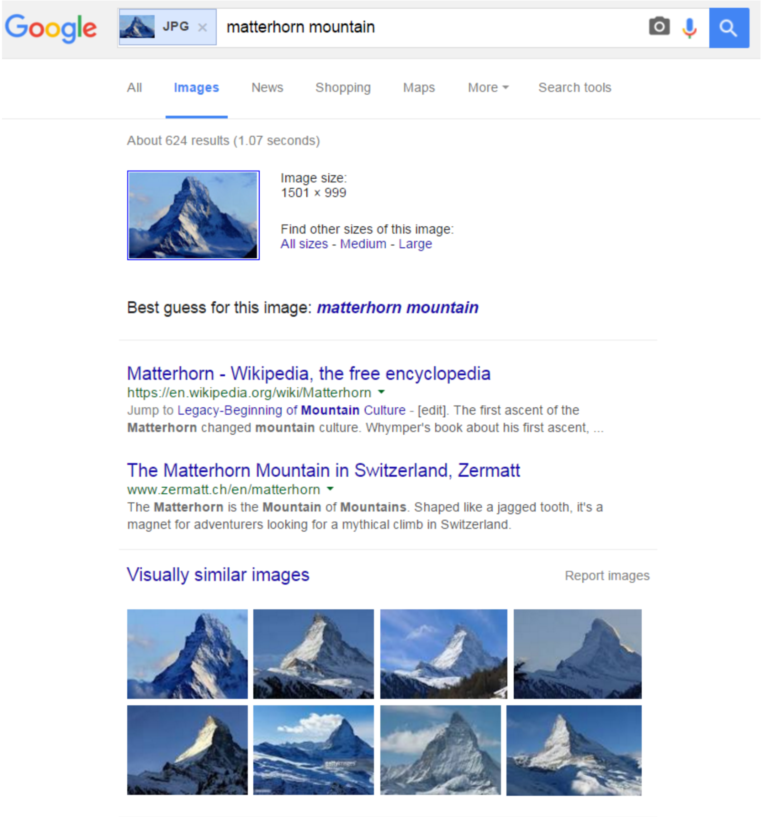
- **Deep Learning tutorial** is given by my PhD students [Daniel Gehrig]and [Elia Kaufmann]
- **Optional lab exercise** is online: **K-means clustering** and **place recognition** with Bag of Words





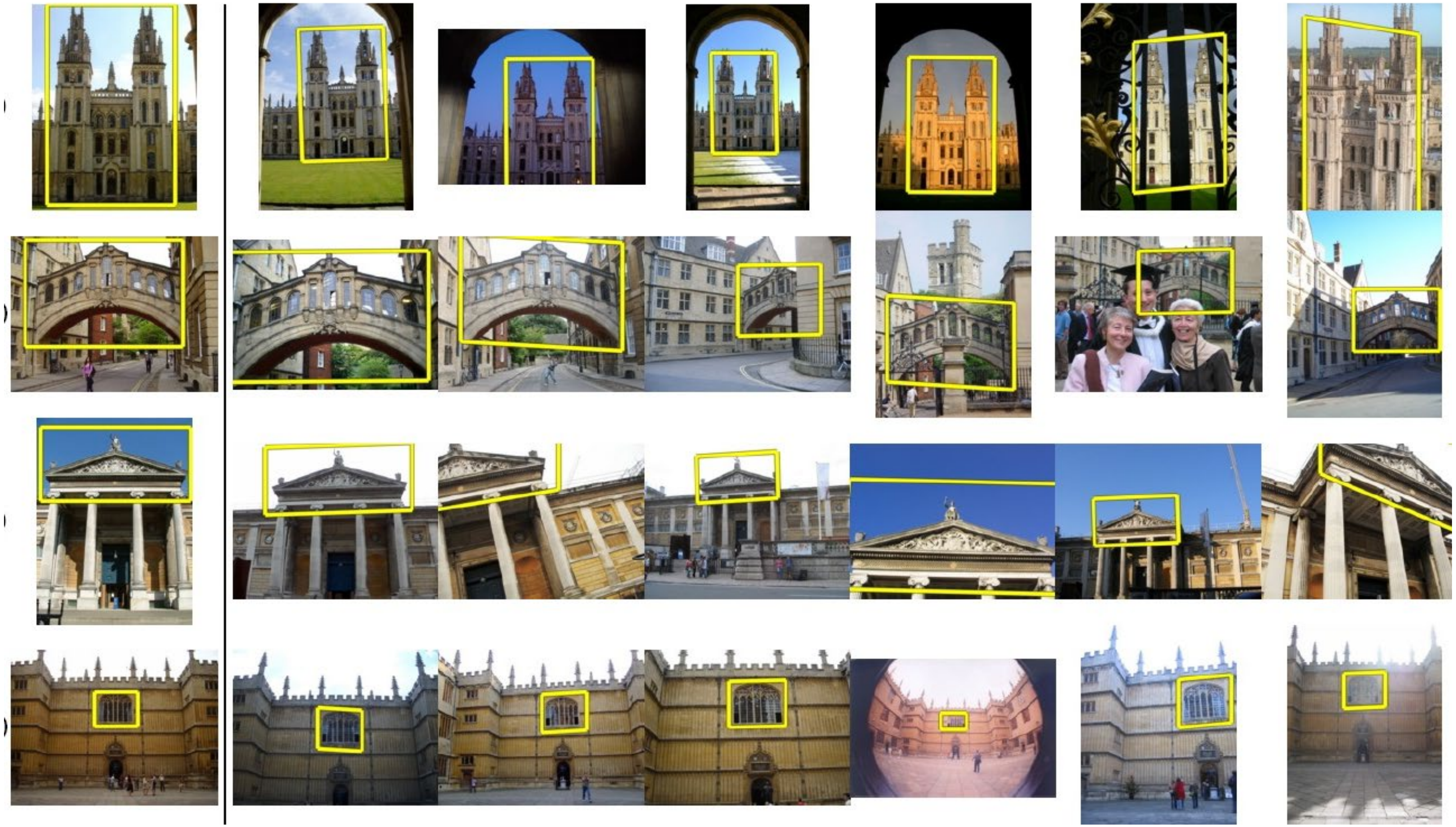
Place Recognition

- **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



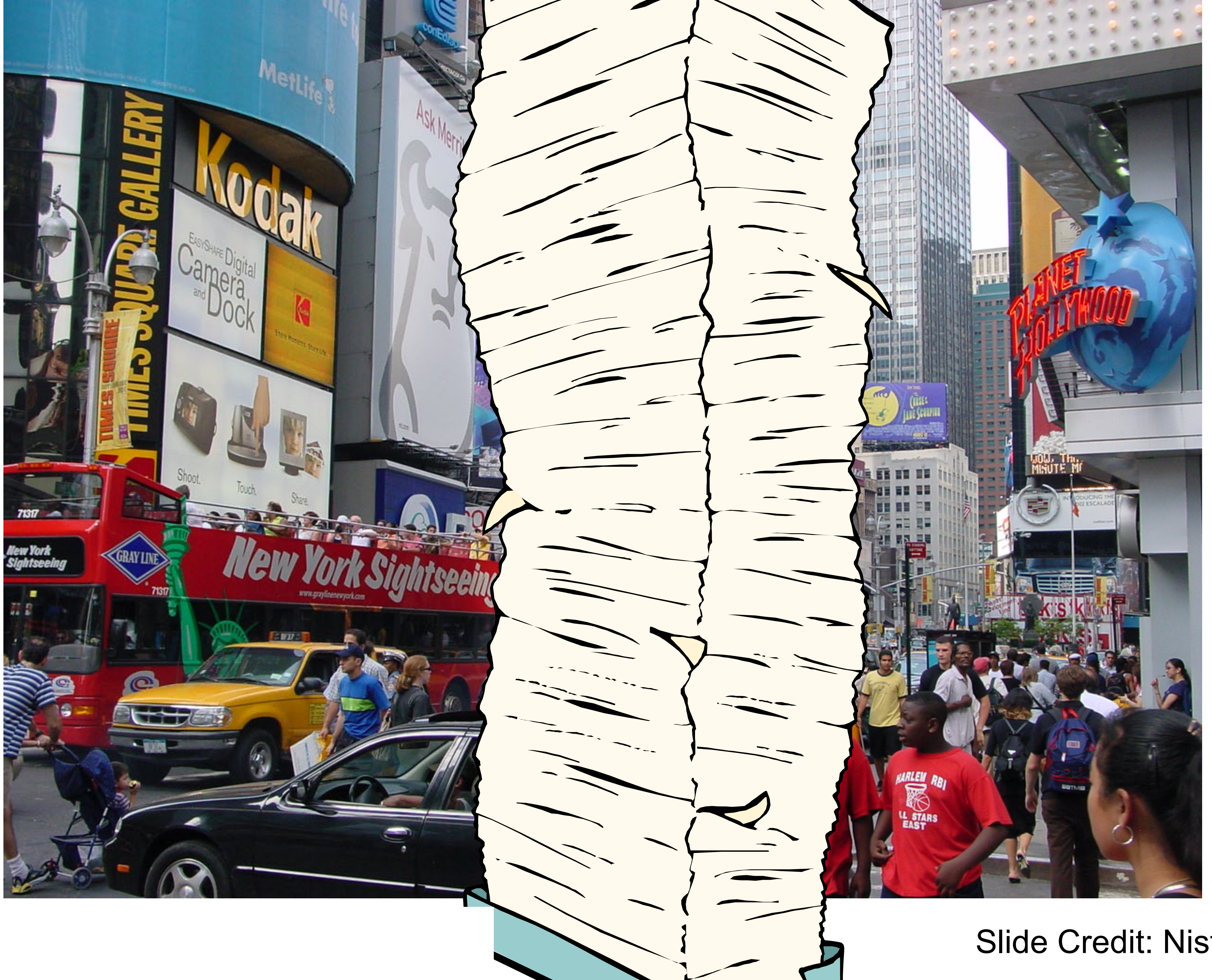
Place Recognition/Image Retrieval

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

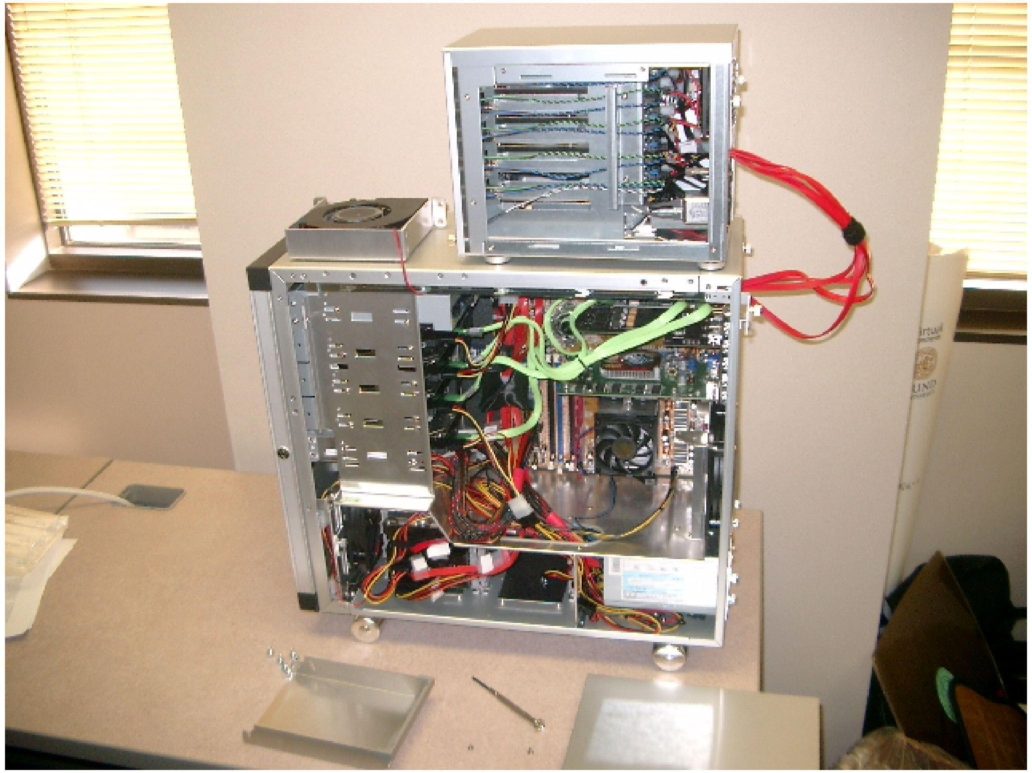
Visual Place Recognition

- **Goal**: **query** an image in a database of $\boldsymbol{N}$ **images**
- **Complexity:** $O(NF^2)$ feature comparisons (assumes each image has ***F*** features)
  - Example:
    - assume 1,000 SIFT features per image → $\mathrm{F} = 1{,}000$
    - assume $N = 100{,}000{,}000$
    - → NF^2= $100{,}000{,}000{,}000{,}000$ feature comparisons!
    - If we assume 10 microseconds per feature comparison → 1 image query would take **32 years**!

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

Fast visual search

How do we **query an image in a database of 100 million images in just 0.6 seconds**?



Sivic, Zisserman, *Video Google: A Text Retrieval Approach to Object Matching in Videos*, International Conference on Computer Vision (ICCV), 2003. PDF.
Nister, Stewenius, *Scalable Recognition with a Vocabulary Tree*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2006. PDF.

Text Retrieval

- Image retrieval takes inspiration from **text retrieval**
- For text documents, an efficient way to find all **pages** in which a **word** occurs is to use an **index file**
- To **retrieve a given text query**, it is then sufficient to use a **voting scheme**

RIFLE, U.S. CAL .30, M1: DIAGRAMS & PICTURES

ALPHABETICAL INDEX

Alphabetical Index

Text Retrieval Example

- Suppose that we have a **document with 10 pages** and we want to determine on which page this sequence of words appears:

”Zurich is a city of Switzerland”

- *”Zurich is a city of Switzerland”* is the **query text**
- Suppose that this is our index file:

Word	Page numbers
Zurich	3, 5, 7
City	1, 3, 5, 9
Switzerland	2, 3, 6, 8

Text Retrieval Example

- The solution is to use a **voting array** that has as many cells as the number of pages in the document
- We first set all the cell values to 0
- Then, we add 1 to each cell corresponding to the page numbers where the words of the query text appear according to the index file

Page number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Cell value	0	0	0	0	0	0	0	0	0	0

Text Retrieval Example

- The solution is to use a **voting array** that has as many cells as the number of pages in the document
- We first set all the cell values to 0
- Then, we add 1 to each cell corresponding to the page numbers where the words of the query text appear according to the index file

”Zurich” {3, 5, 7}

Page number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Cell value	0	0	1	0	1	0	1	0	0	0

Text Retrieval Example

- The solution is to use a **voting array** that has as many cells as the number of pages in the document
- We first set all the cell values to 0
- Then, we add 1 to each cell corresponding to the page numbers where the words of the query text appear according to the index file

”City” {1, 3, 5, 9}

Page number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Cell value	1	0	2	0	2	0	1	0	1	0

Text Retrieval Example

- The solution is to use a **voting array** that has as many cells as the number of pages in the document
- We first set all the cell values to 0
- Then, we add 1 to each cell corresponding to the page numbers where the words of the query text appear according to the index file

"Switzerland" {2, 3, 6, 8}

Page number	1	2	3	4	5	6	7	8	9	10
Cell value	1	1	3	0	2	1	1	1	1	0

Bag of Words

Using the analogy from text retrieval, we need to:

- define what a "*visual word*" is and
- define a "*vocabulary*" of visual words

This approach is known as "Bag of Words" (BOW)

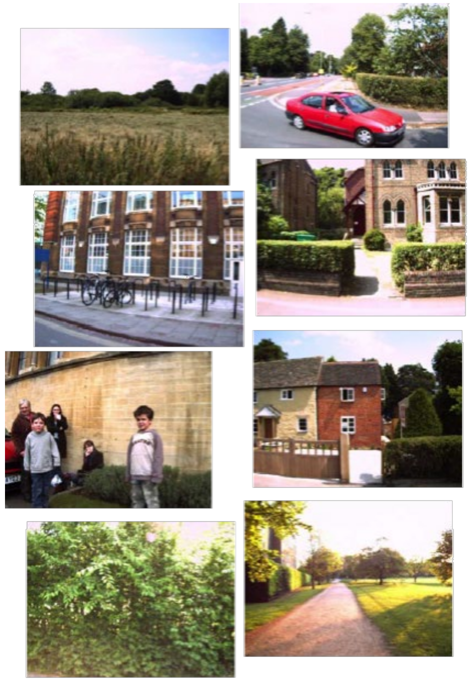
- Can a SIFT descriptor be used as a visual word? And a BRISK descriptor?
  - SIFT → $128 \times 4\ bytes\ float = 512\ bytes = 4096\ bits = 2^{4096}$ possible SIFT descriptors!
  - BRISK-128 → $128\ \text{bits} = 2^{128}$ possible BRISK descriptors!
  - **Usually, 1 million visual words is enough**
  - **Idea**: **cluster SIFT descriptors into visual words**

How to extract Visual Words from descriptors

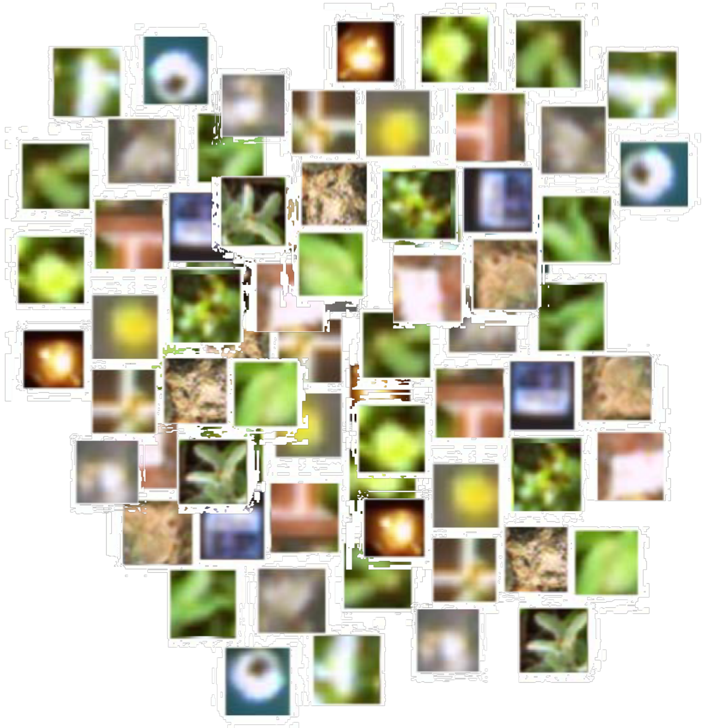
- **Collect a large enough dataset** that is representative of all possible images that are relevant to your application (e.g., for automotive place recognition, you may want to collect million of street images sampled around the world)
- Extract features and descriptors from each image and map them 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

Extracting Visual Words

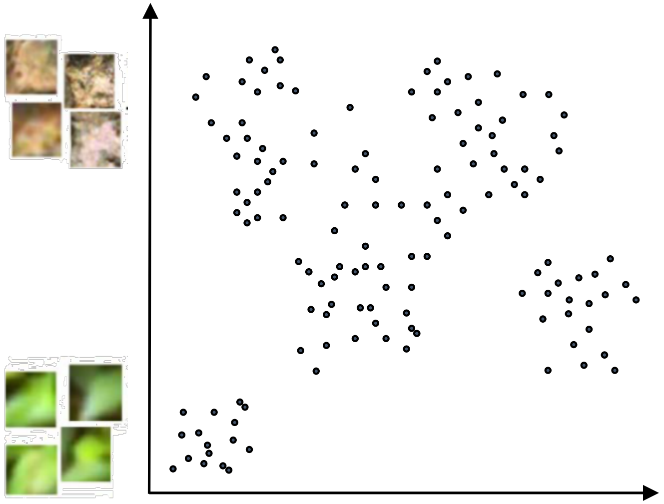
Image database
(e.g., 100 million images)



Feature extraction
(~1,000 features per image)

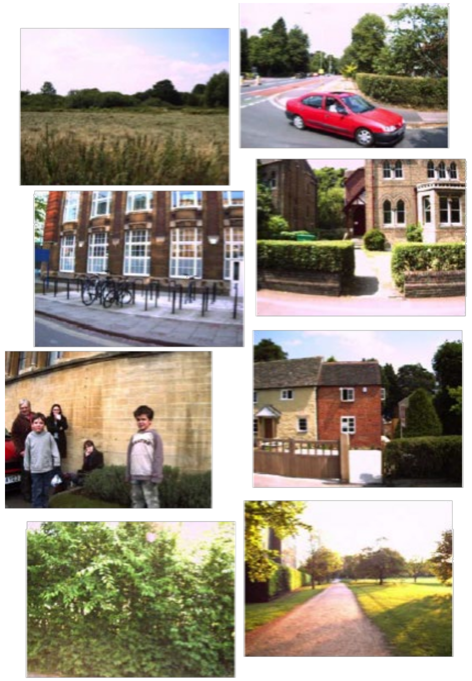


Map all features into the descriptor space
(~100 billion descriptors)

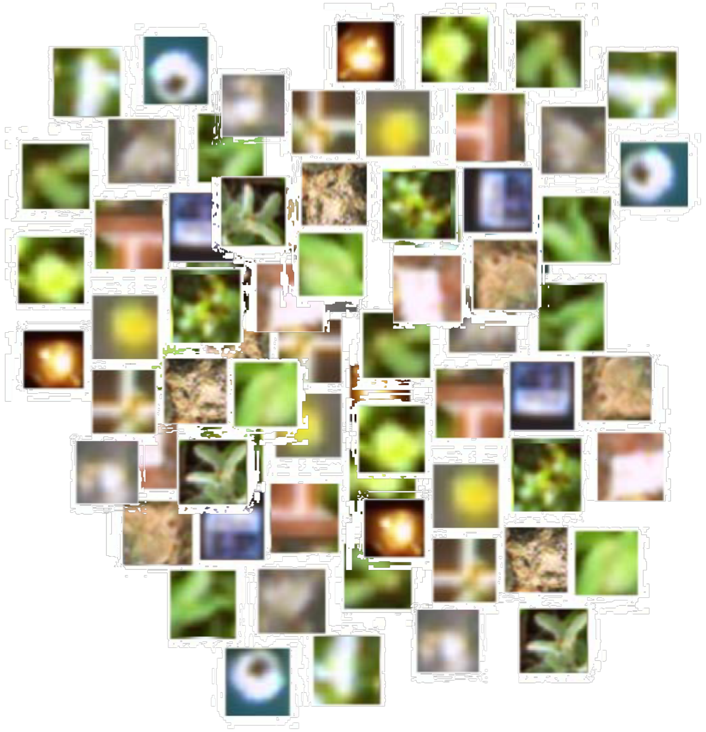


Extracting Visual Words

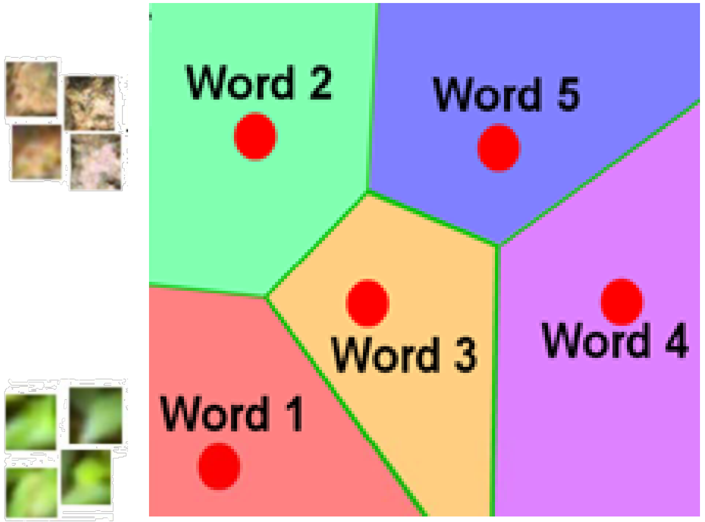
Image database
(e.g., 100 million images)



Feature extraction
(~1,000 features per image)

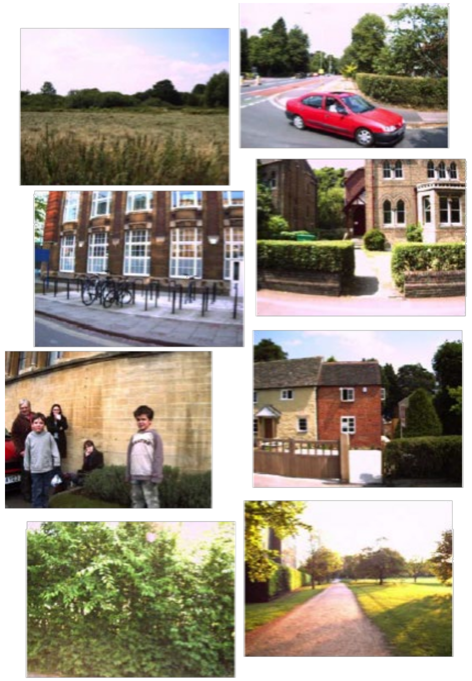


Feature clustering
(~1 million words)

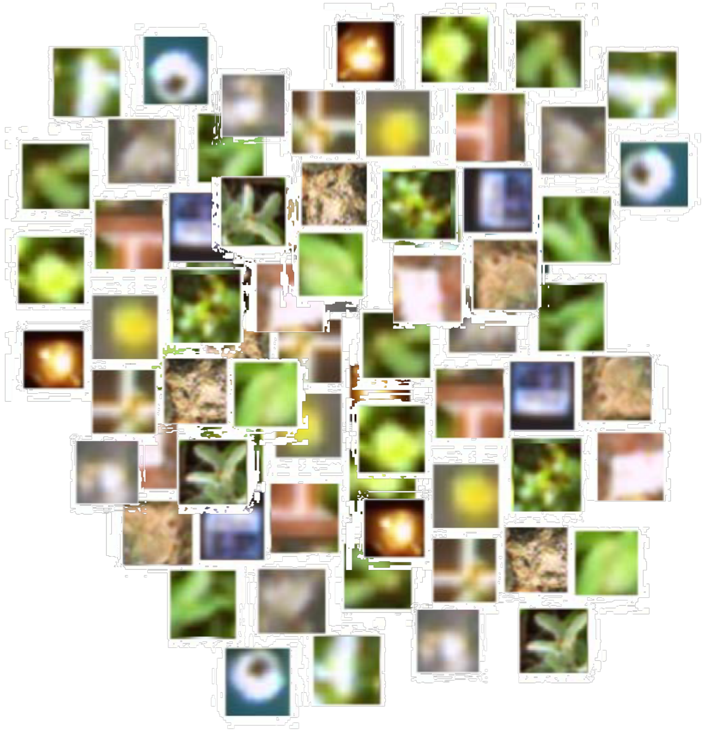


Extracting Visual Words

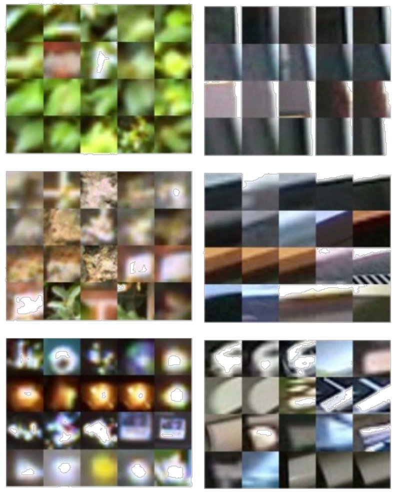
Image database
(e.g., 100 million images)



Feature extraction
(~1,000 features per image)



Feature clustering
(~1 million words)



Examples of features belonging to the same clusters (i.e., to the same visual word)

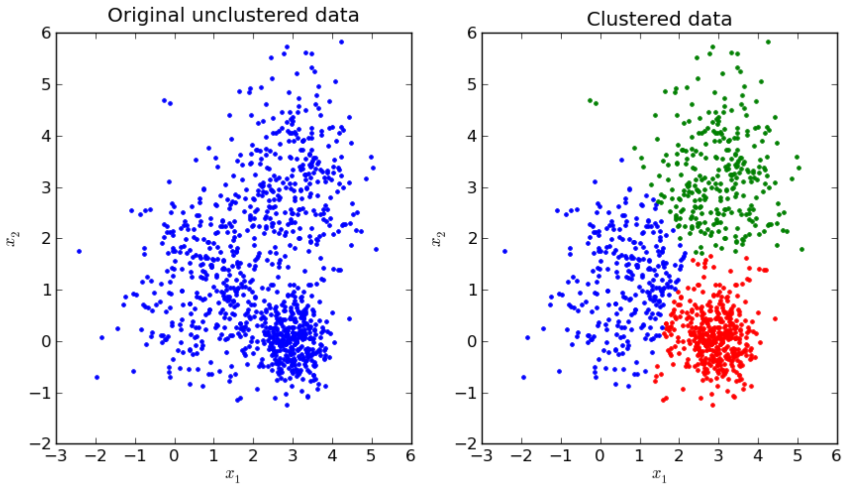
Extracting Visual Words

- Extracting SIFT features from a VGA images takes ~20 ms on an i7 CPU
- This means that the **extraction of features from 100 million images would take ~23 days** without accounting for the time needed for clustering all these features
- However, notice that **this is ok since this process only needs to be done once**, when the database has been created.
- **If the database grows** as new images are collected, **new features can be extracted and the visual words can be updated accordingly**. This update process **does not need to run in real time**

How do we cluster the descriptor space?

- **k-means clustering** is an algorithm to partition n data points 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 **squared Euclidean distance** 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:

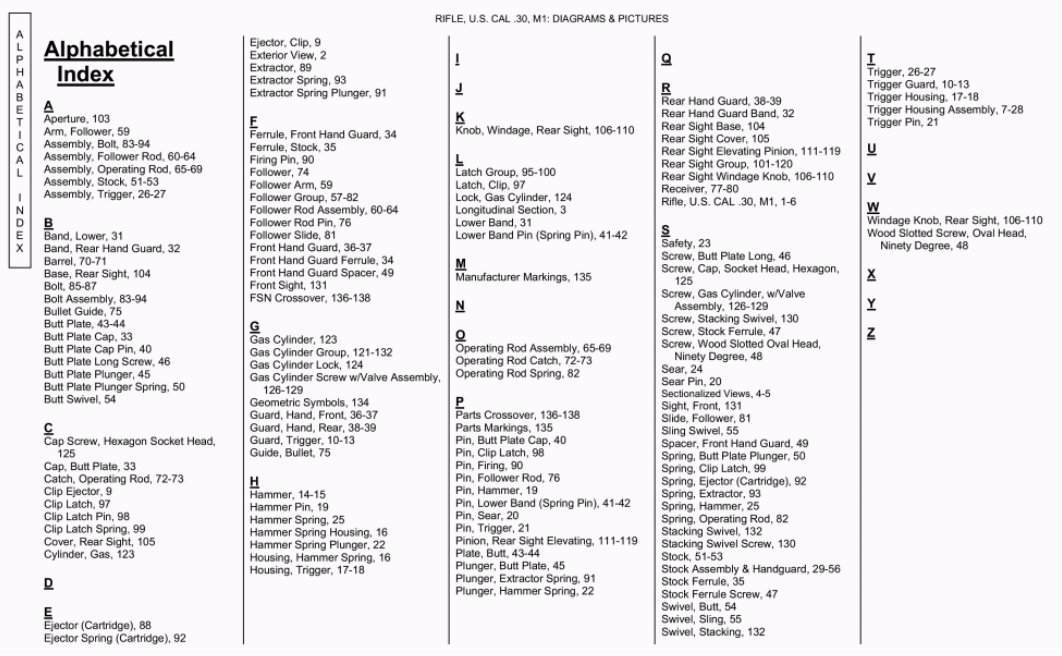
- Randomly initialize k cluster centers
- Iterate until convergence:
  - Assign each data point $\boldsymbol{x}_j$ to the nearest center $\boldsymbol{m}_i$
  - Recompute each cluster center as the mean of all points assigned to it

K-means demo

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

Building the Image Vocabulary

- The **Image Vocabulary** is a data structure that lists all extracted visual words
- Each visual word is assigned a unique identifier (an **integer number**)
- **Each visual word** in the image vocabulary **points to a list of images** (from the entire image database) in which that word appears
- If the database grows, the vocabulary is updated accordingly


RIFLE, U.S. CAL .30, M1: DIAGRAMS & PICTURES

ALPHABETICAL INDEX

Alphabetical Index

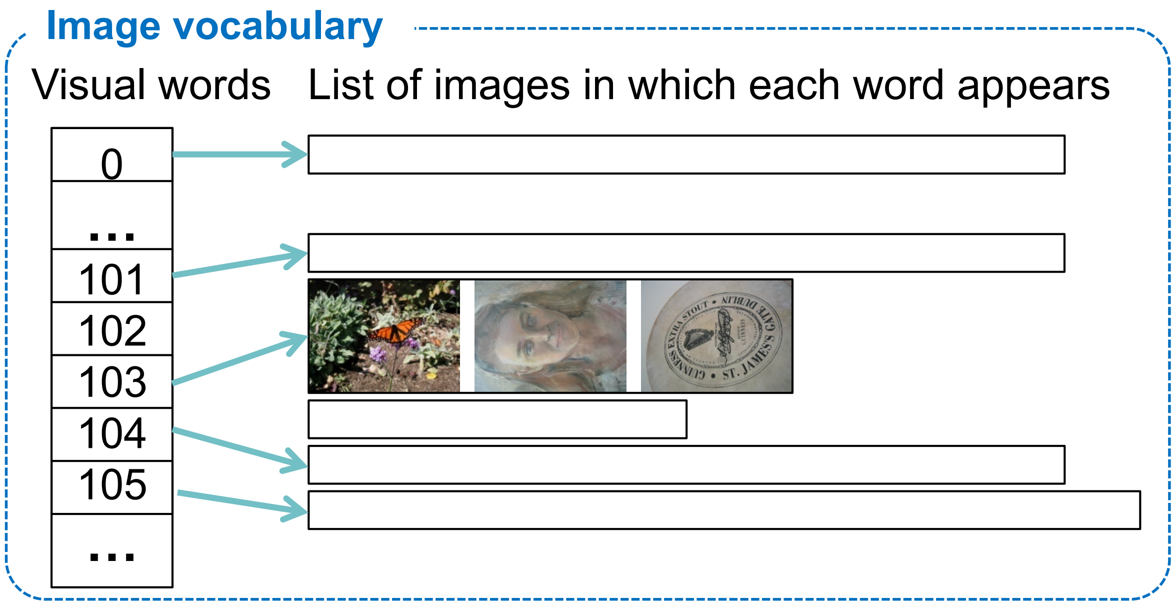


Image Retrieval

1. To query an image in the database, we must first extract features from the query image (this takes about 20 ms for ~1,000 SIFT features)

Query image Q

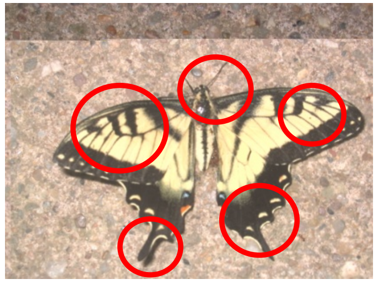
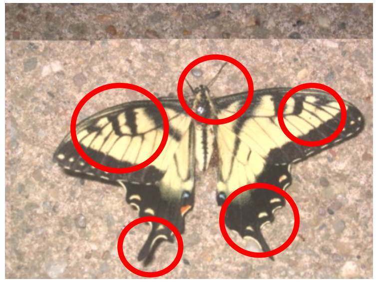


Image Retrieval

2. Then, we initialize the **voting array** to 0 (the voting array has as many cells as the number of images in the database).

Query image Q



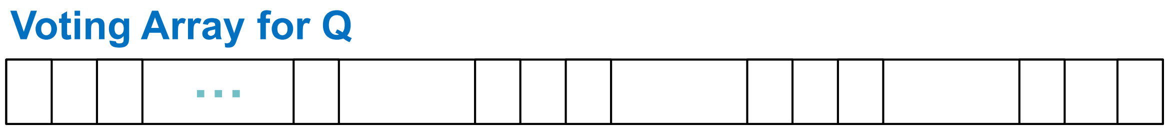


Image Retrieval

3. Then, we **look-up** each feature in the image vocabulary (basically, we look for the **closest visual word** in the vocabulary)

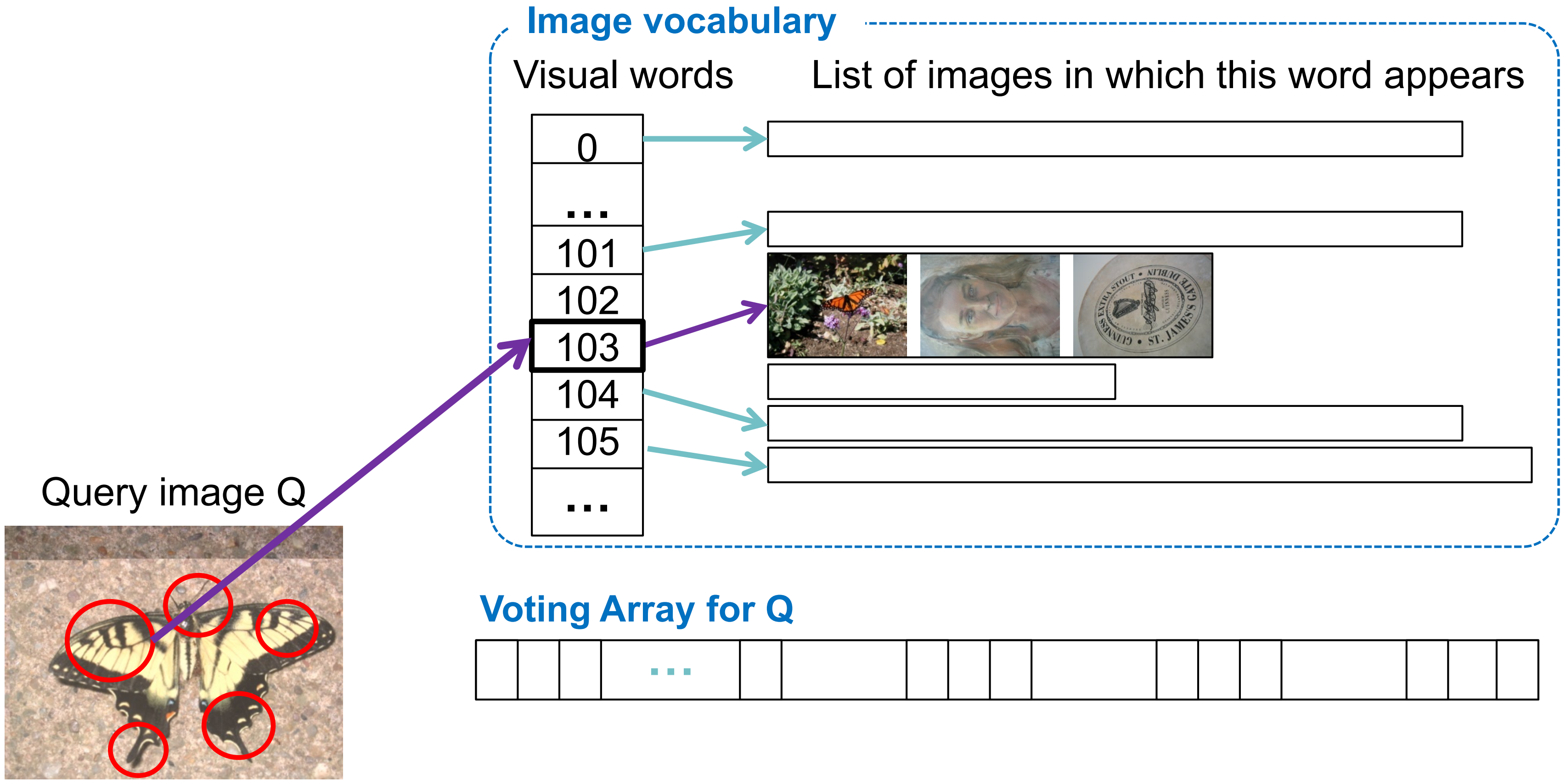
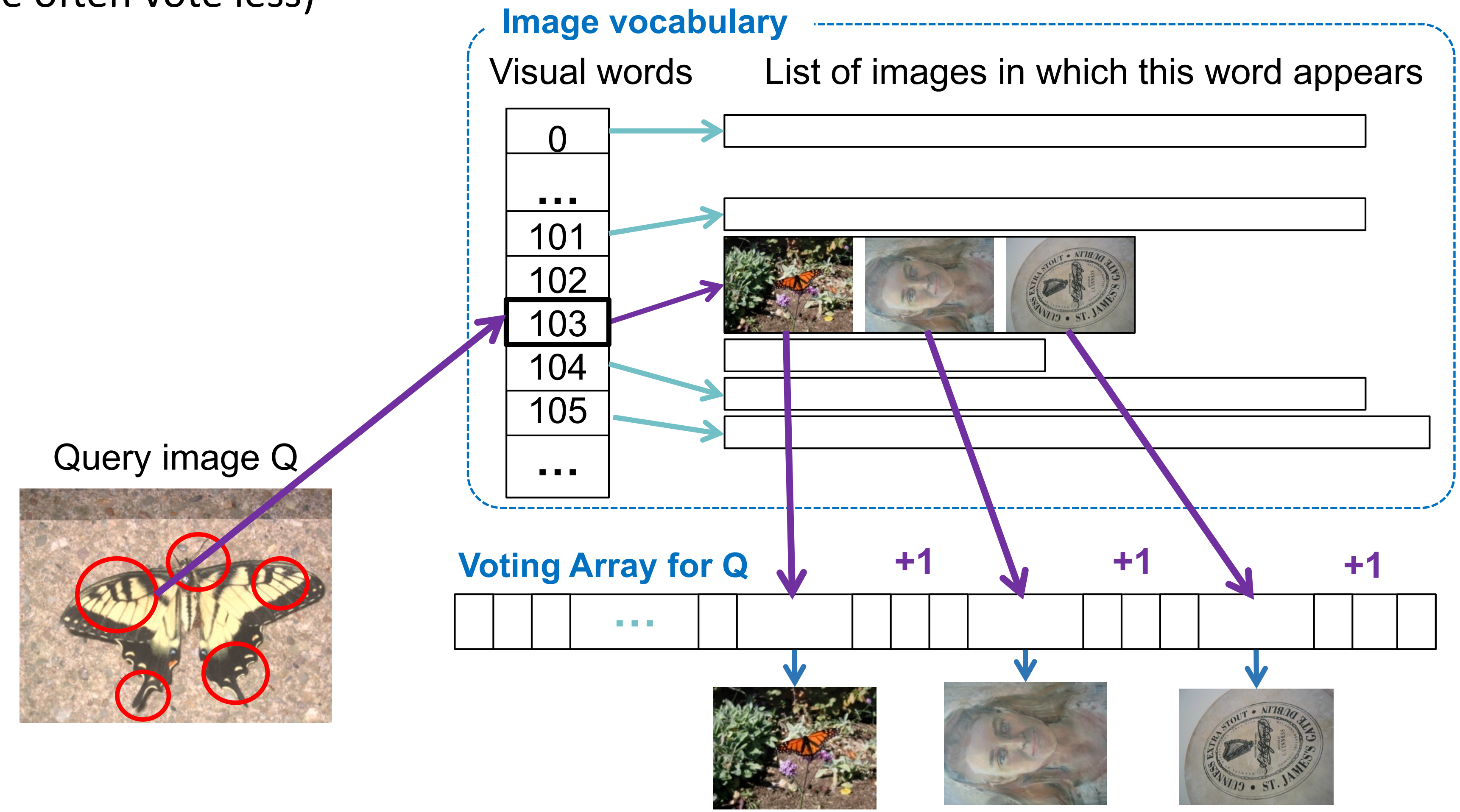


Image Retrieval

4. Finally, **each visual word votes for multiple images** as we saw for the case of text retrieval; however, the **voting is not uniform but is weighted by the inverse of the frequency of the word** (i.e., words that repeat more often vote less)



Issues

Every feature in the query image has to be compared against all visual words in the vocabulary:

- Example:
  - assume our query image has 1,000 features;
  - assume 1 million visual words → number of feature comparisons would be equal to **1 billion**!
  - If we assume 10 microseconds per feature comparison, then querying one image would take ~**3 hours**!

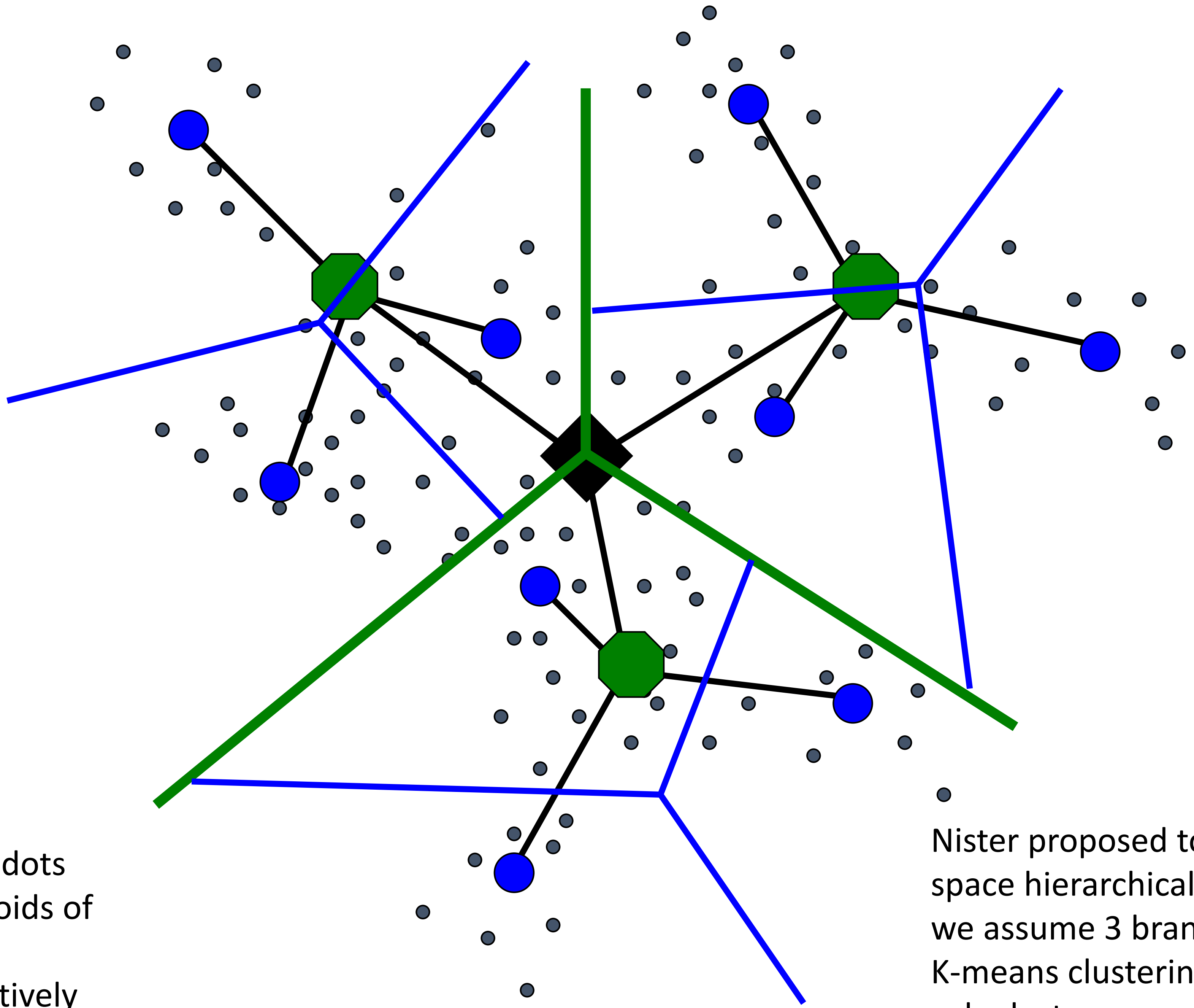
- How can we make the comparison cheaper?
  - Idea: use **hierarchical clustering**

Hierarchical Clustering

- David Nister proposed to cluster the feature space in a **coarse-to-fine manner** so that visual words could be represented as the **terminal vertices of a search tree**.
- As we will see, this significantly reduces the feature-to-word association, bringing image retrieval within the reach of resource-constrained mobile devices!

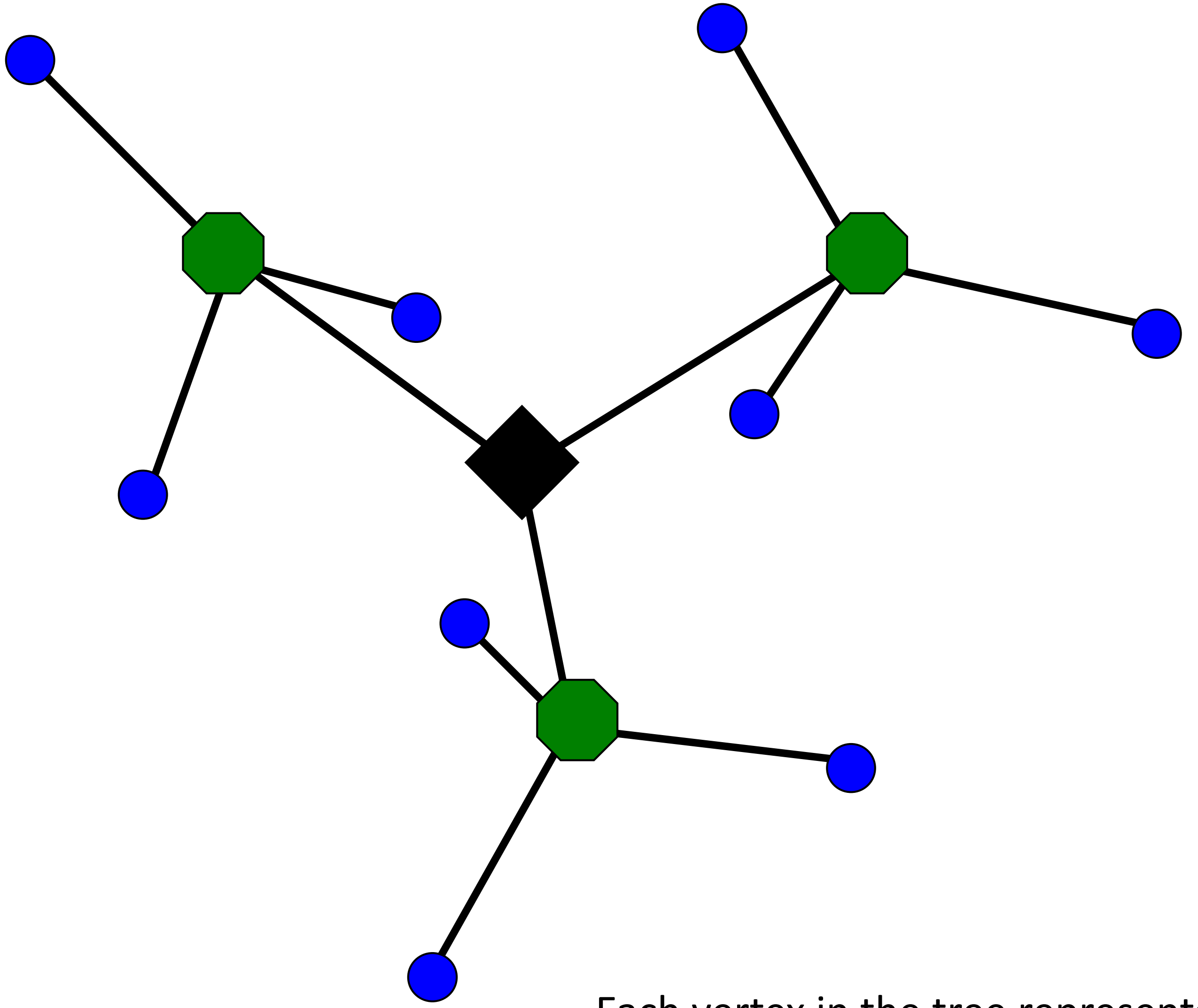
Nister, Stewenius, *Scalable Recognition with a Vocabulary Tree*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2006. PDF.

Each dot represents a SIFT feature.
SIFT features have 128 dimensions!
For convenience, here we assume only
2 dimensions ☺

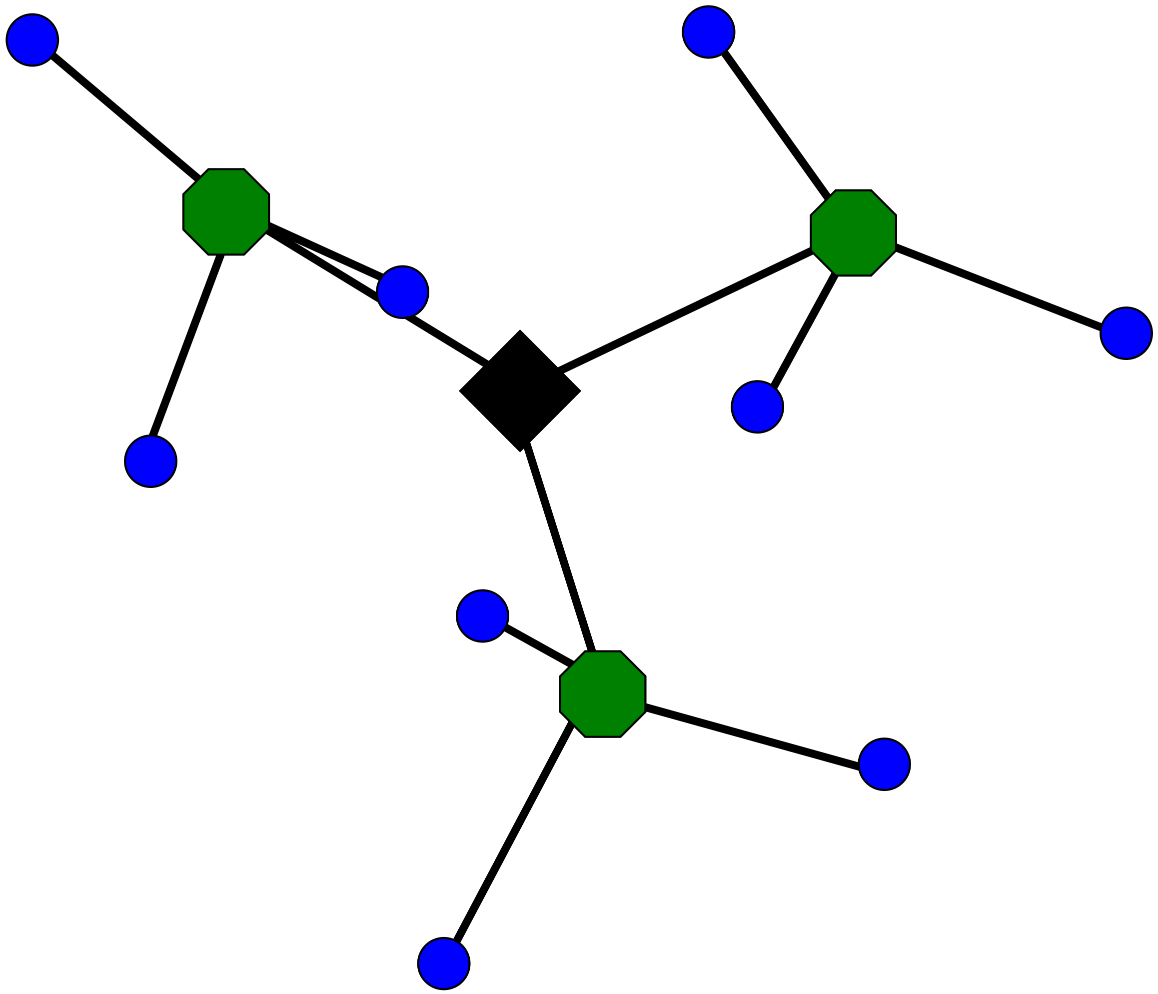


The green and blue dots represent the centroids of level 1 and level 2 sub-clusters, respectively

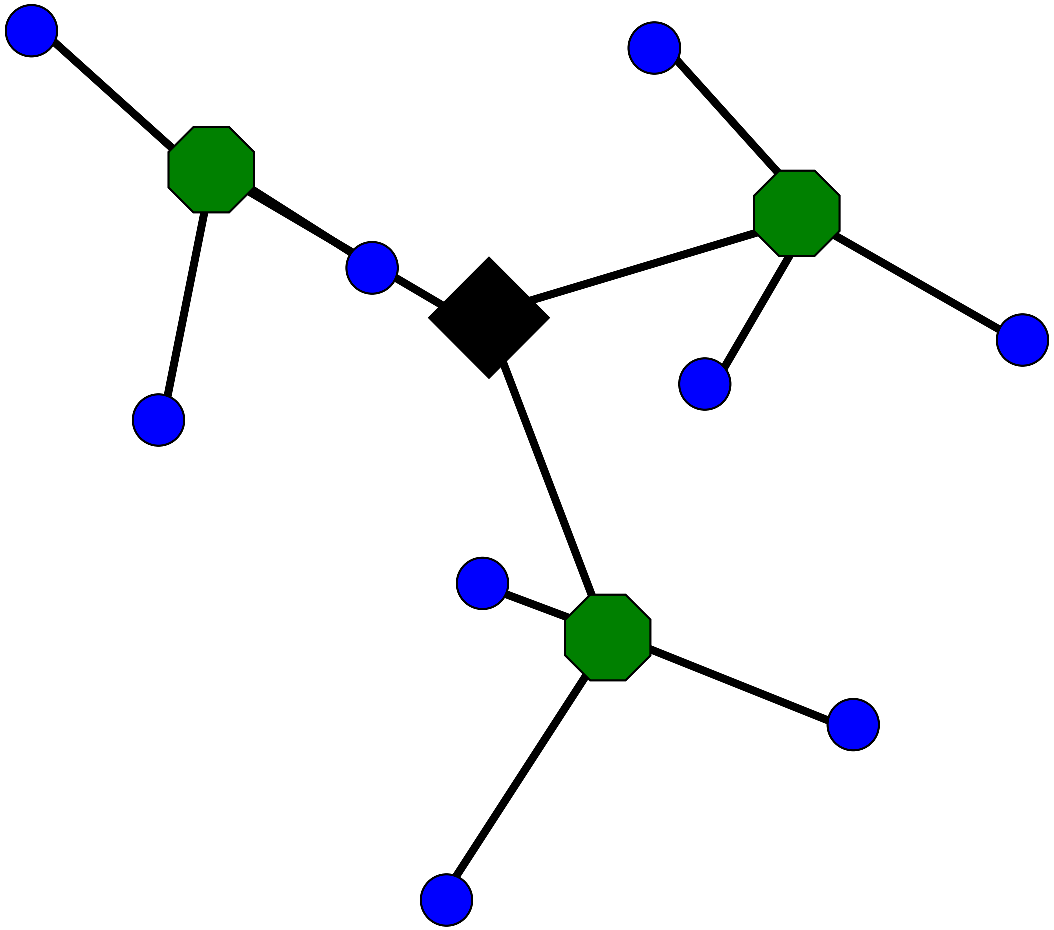
Nister proposed to cluster the feature space hierarchically. For example, here we assume 3 branches and 2 levels. K-means clustering is used to cluster each sub-cluster.



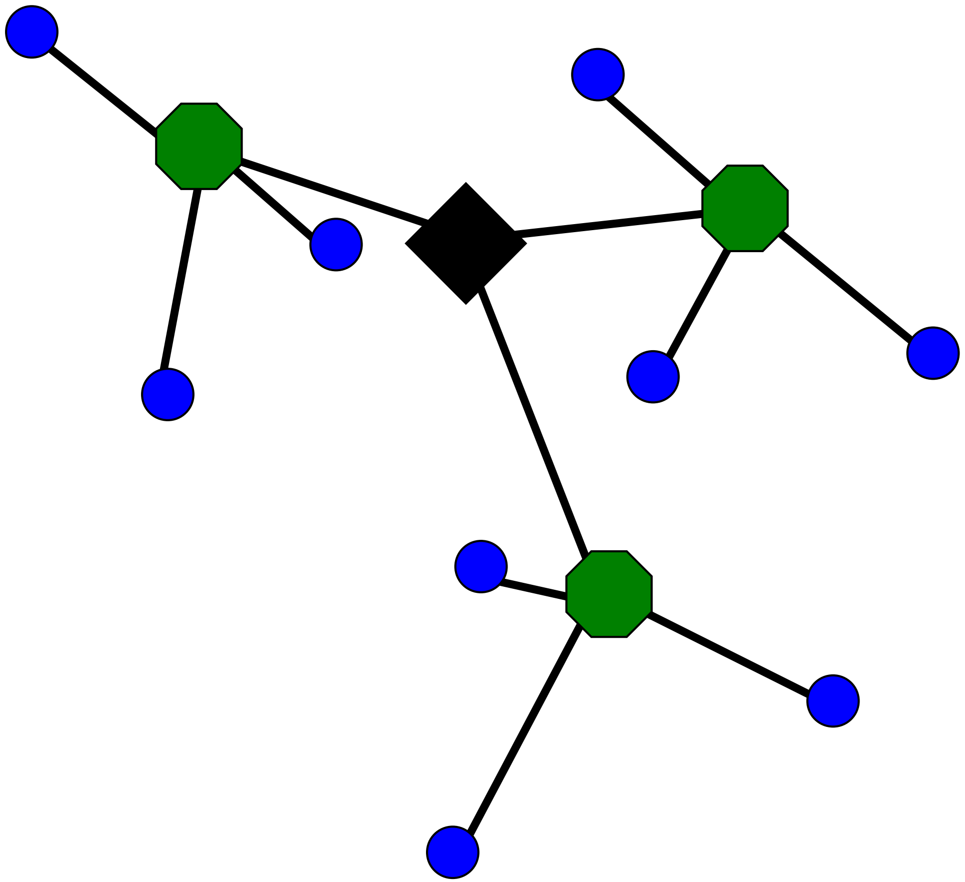
Each vertex in the tree represents a visual word.



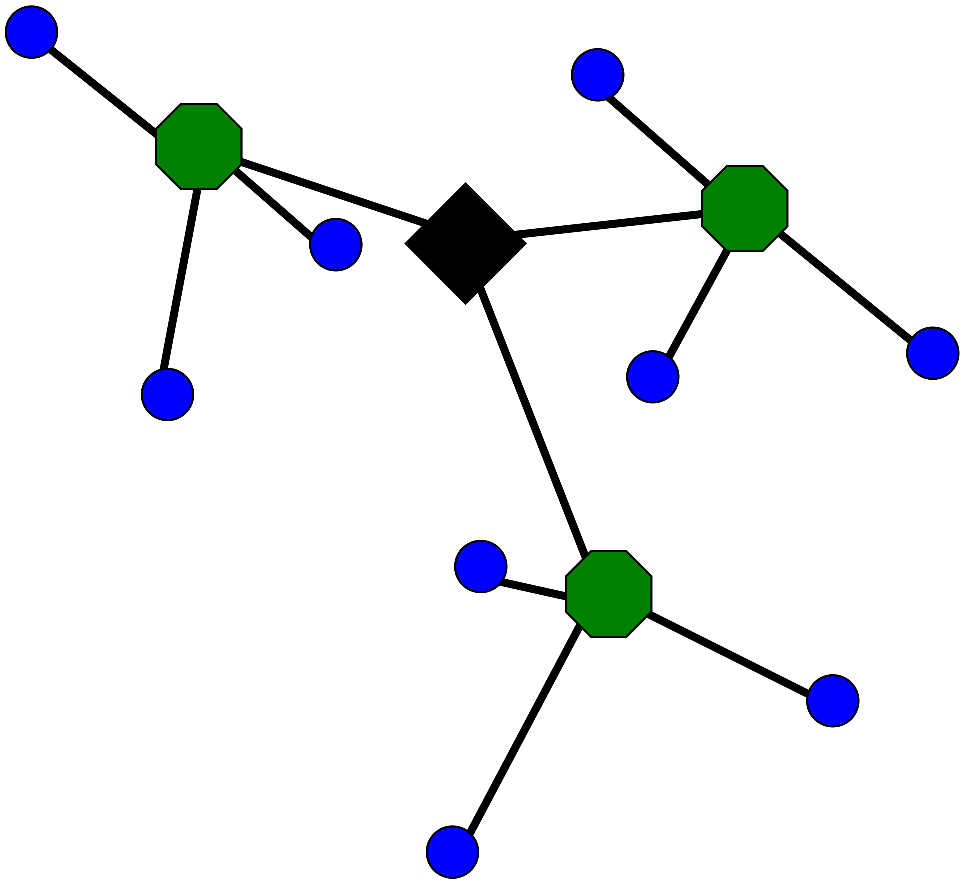
Each vertex in the tree represents a visual word.



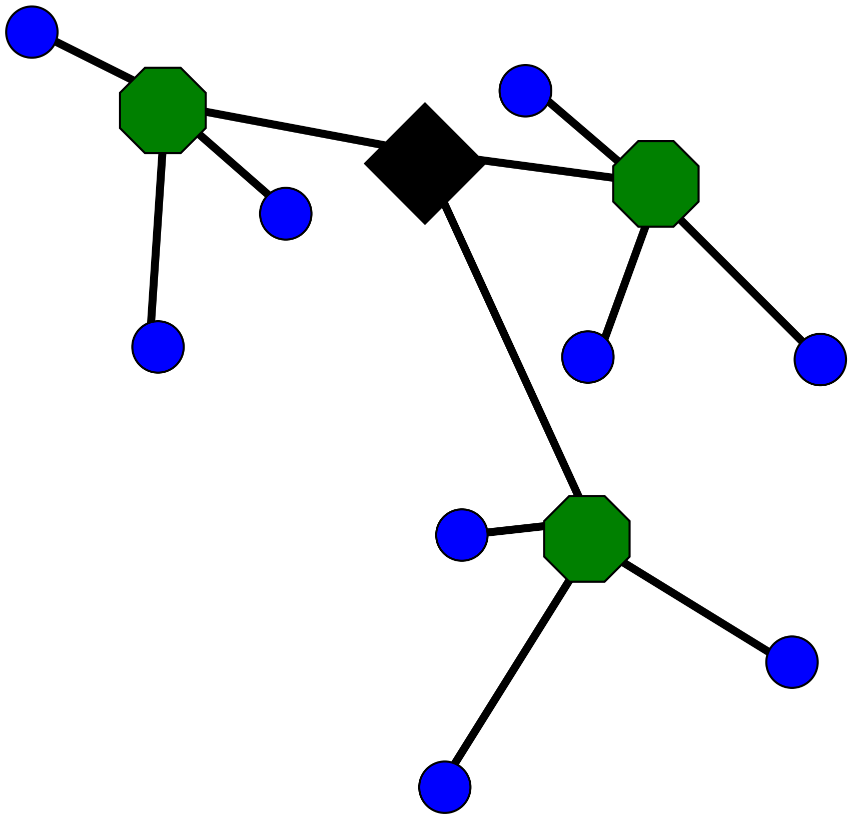
Each vertex in the tree represents a visual word.



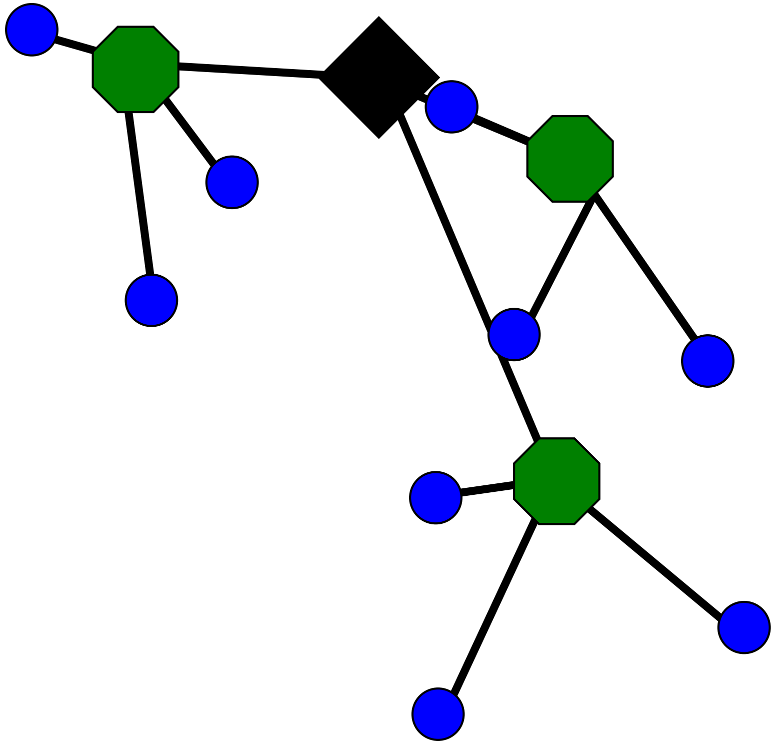
Each vertex in the tree represents a visual word.



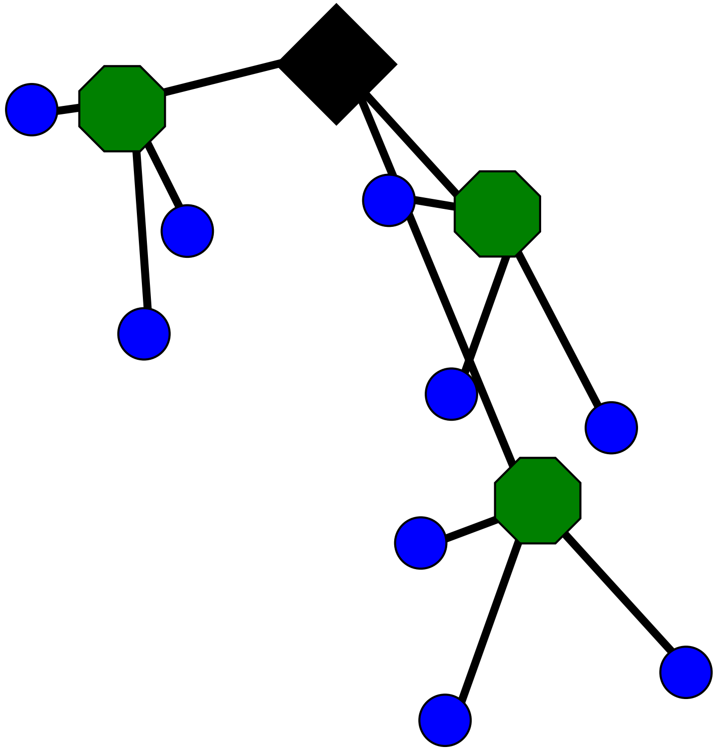
Each vertex in the tree represents a visual word.



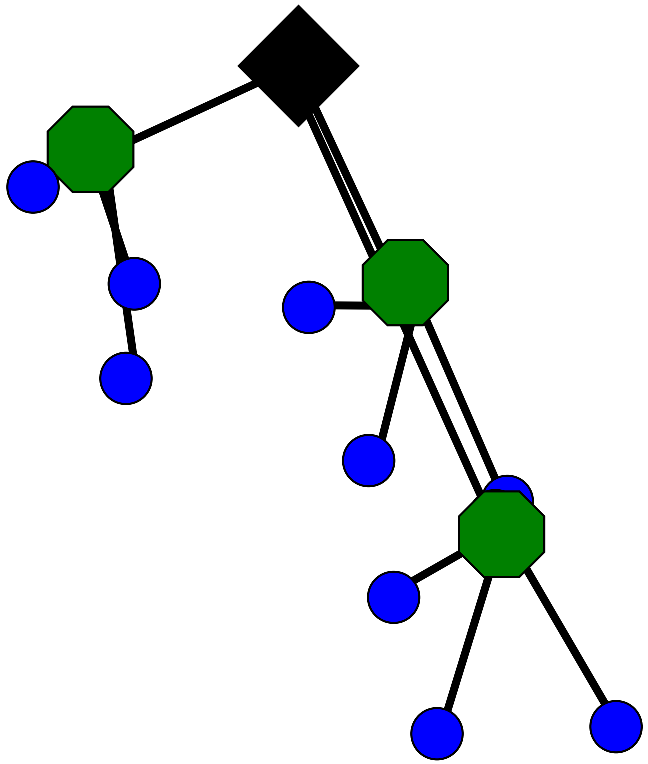
Each vertex in the tree represents a visual word.



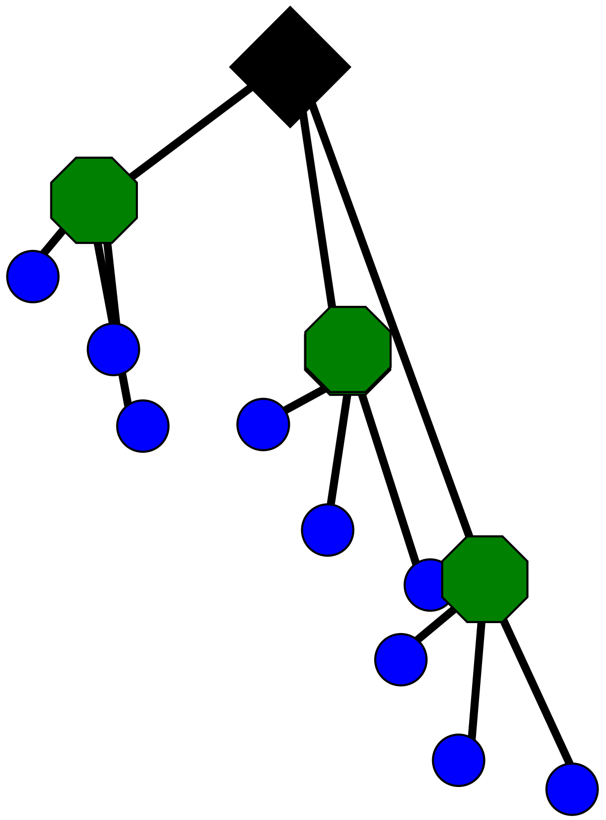
Each vertex in the tree represents a visual word.



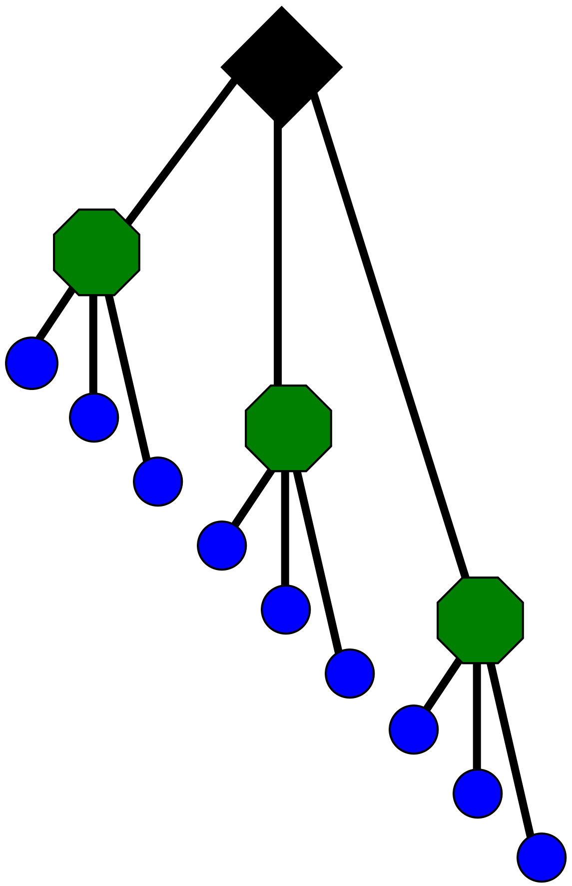
Each vertex in the tree represents a visual word.



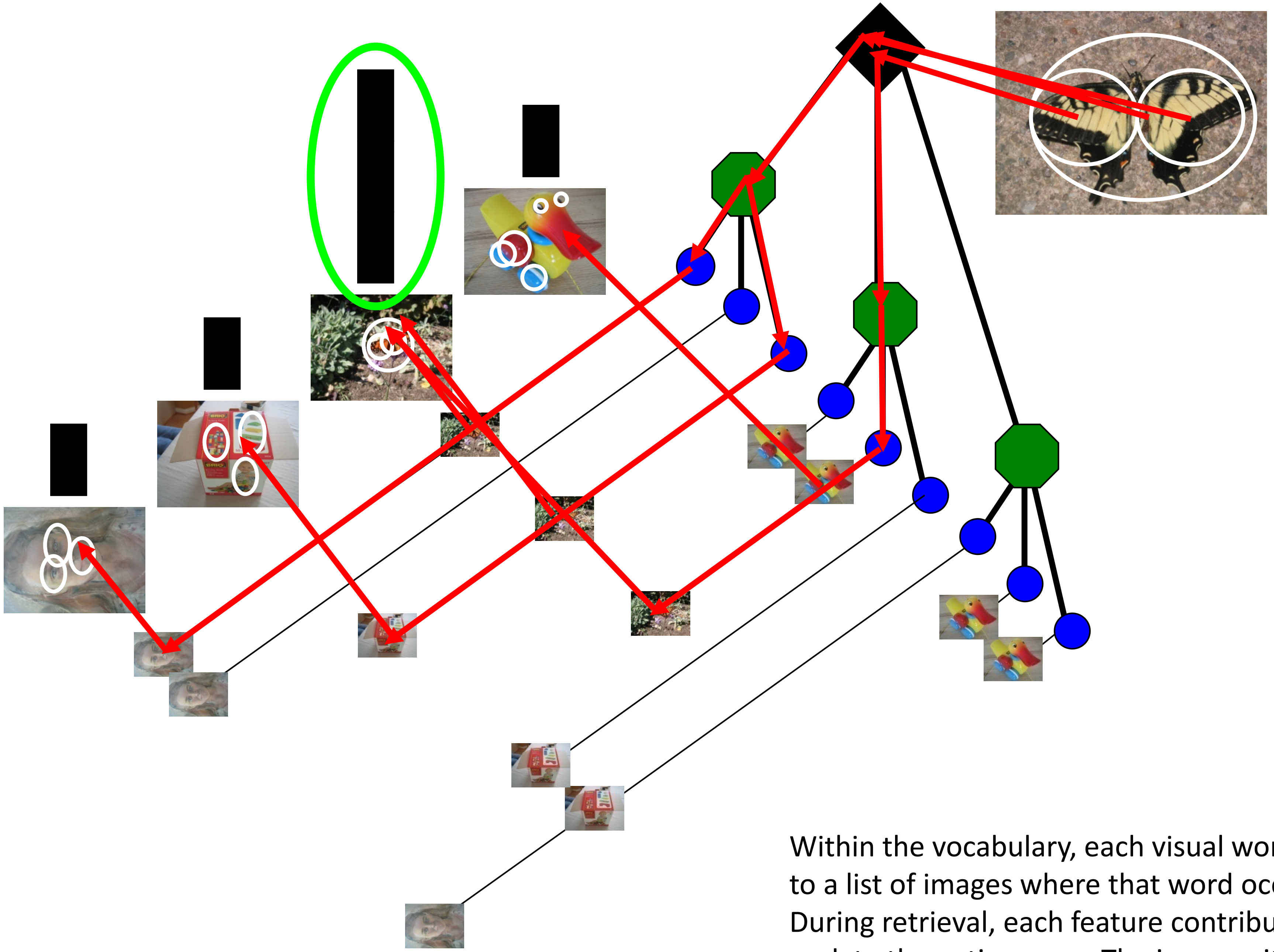
Each vertex in the tree represents a visual word.



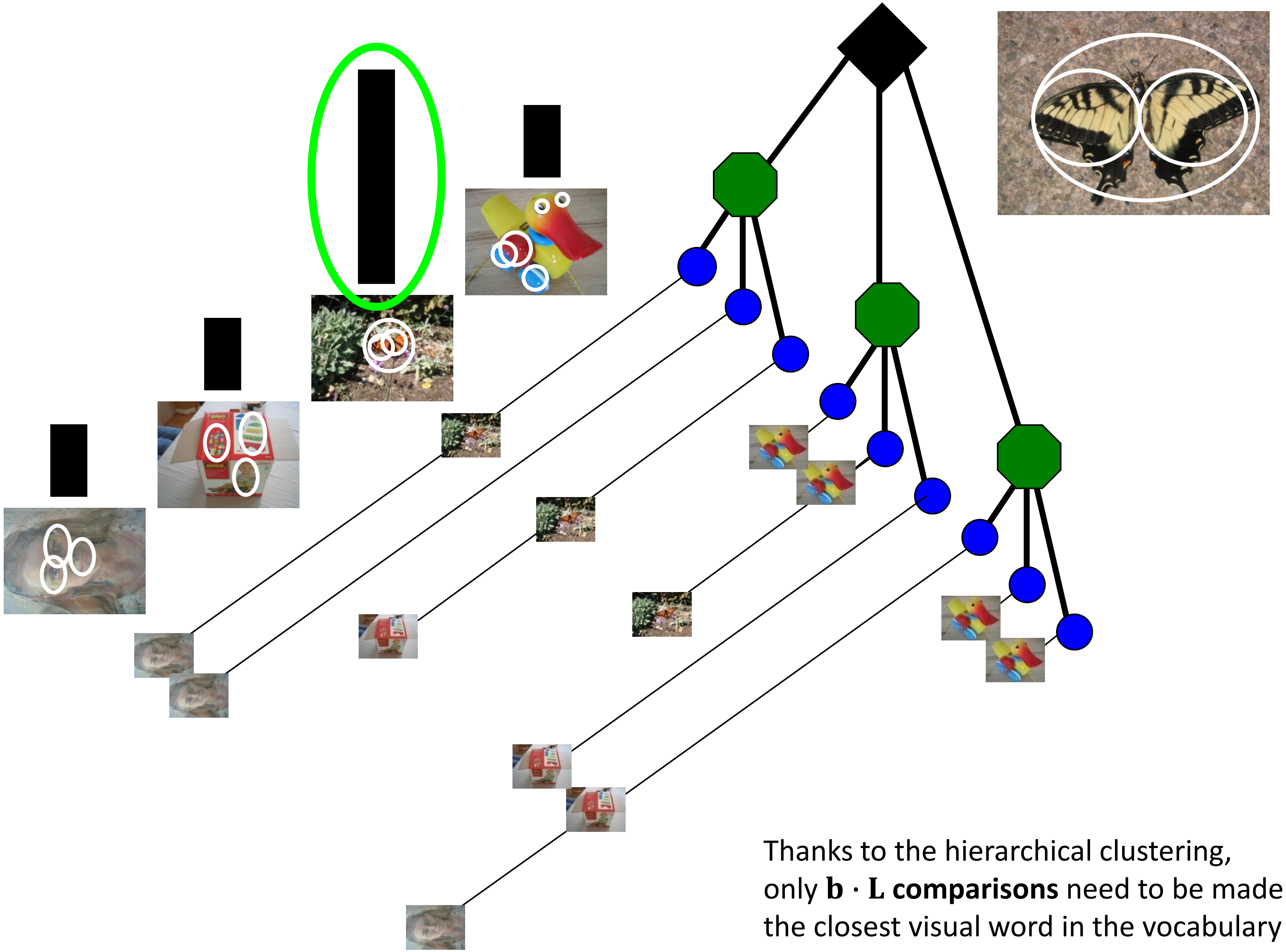
Each vertex in the tree represents a visual word.



Each vertex in the tree represents a visual word.



Within the vocabulary, each visual word points to a list of images where that word occurs. During retrieval, each feature contributes to update the voting array. The image with most votes is returned.



Thanks to the hierarchical clustering,
only **b · L comparisons** need to be made to find
the closest visual word in the vocabulary!
b is the **number of branches** while
L is the **number of levels** of the tree.

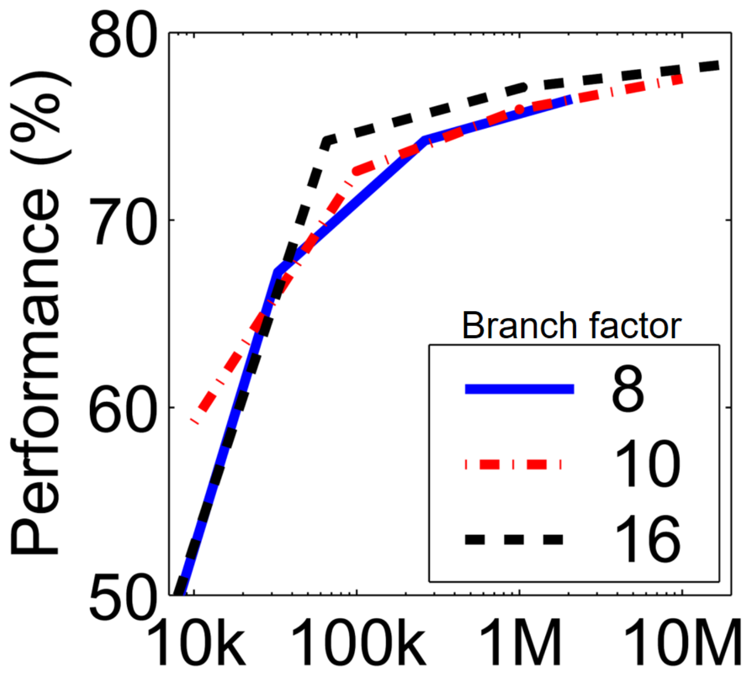
Example

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

- assume a query image with $\mathrm{F} = 1{,}000$ features
- assume a tree structure with $\mathrm{b} = 10$ branches per level and $\mathrm{L} = 6$ levels (i.e., $b^L = 1{,}000{,}000$ visual words)
- Then, the **number of feature comparisons** $= \mathbf{F} \cdot \mathbf{b} \cdot \mathbf{L} = 1{,}000 \cdot 10 \cdot 6 = 60{,}000$ **instead of** $\mathbf{F} \cdot \boldsymbol{b}^{\boldsymbol{L}} = 1\ \text{billion}$ comparisons (which was the case without hierarchical clustering)
- If we assume 10 microseconds per feature comparison
  → 1 image query would take **0.6 seconds**!

How many visual words, branches, and levels?

- **More words is better**, but **1 million words are used in practice**
- Also, **6 branches and 10 depth levels are practically used** (e.g., ORB-SLAM)



Geometric Verification

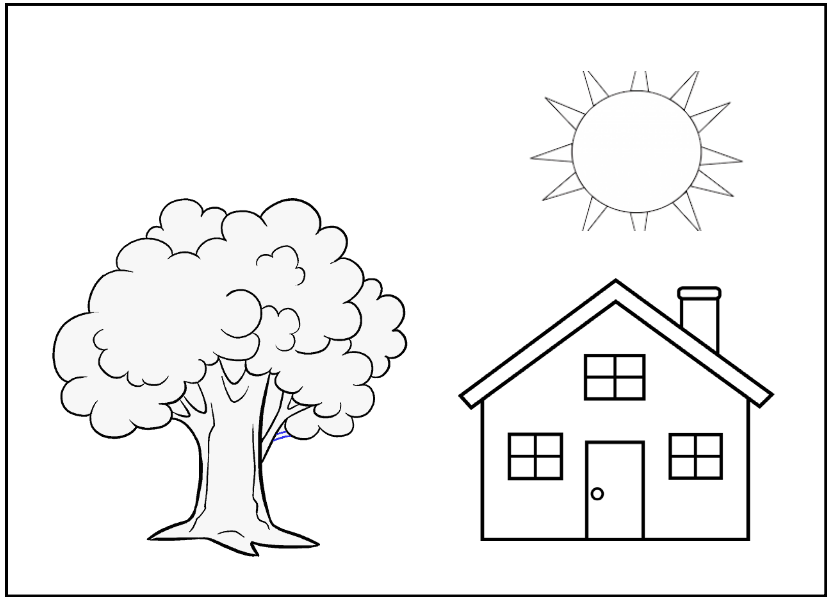
Bag of Words returns a list of images with score above a given threshold. How do we pick the image that was captured **closest** to the query image?

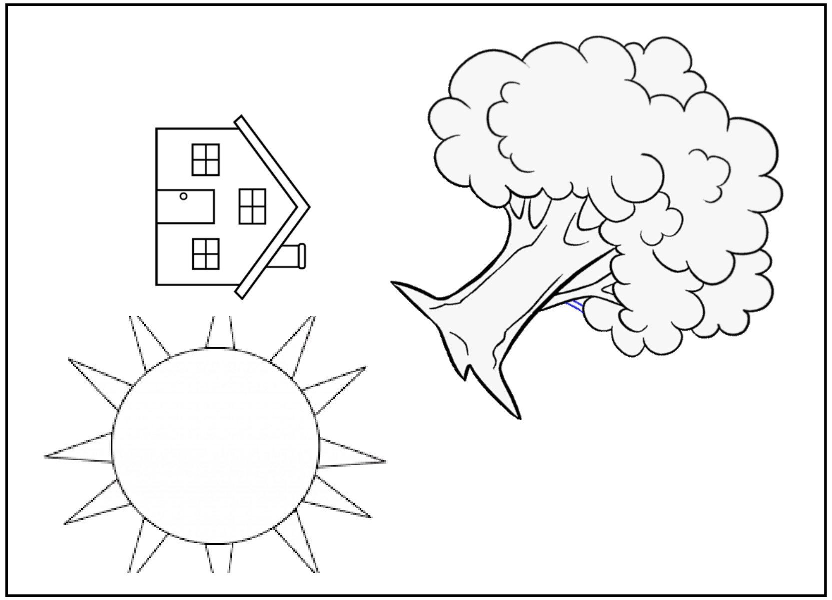
Query image Results with the highest matching score

- Observe that the visual vocabulary discards the spatial relationships between features
- Solution: Test each returned image for geometric consistency against the query image (e.g. using 5- or 8-point RANSAC) and pick the image with **the smallest reprojection error and largest number of inliers**

Curiosity

Imagine to query two images with the same features shuffled around. Will the scores returned by Bag of Words be different?





Open Challenges

When does place recognition fail and how would you address them?

Things to remember

- K-means clustering
- Bag of Words approach
    - What is visual word
    - What is a visual vocabulary
    - How do we query an image

Readings

- Sivic, Zisserman, *Video Google: A Text Retrieval Approach to Object Matching in Videos*, International Conference on Computer Vision (ICCV), 2003. PDF.
- Nister, Stewenius, *Scalable Recognition with a Vocabulary Tree*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2006. PDF.

Understanding Check

Are you able to answer the following questions?

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