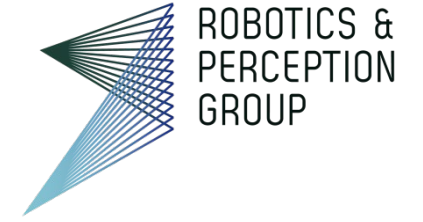




**University of
Zurich** ^{UZH}



Vision Algorithms for Mobile Robotics

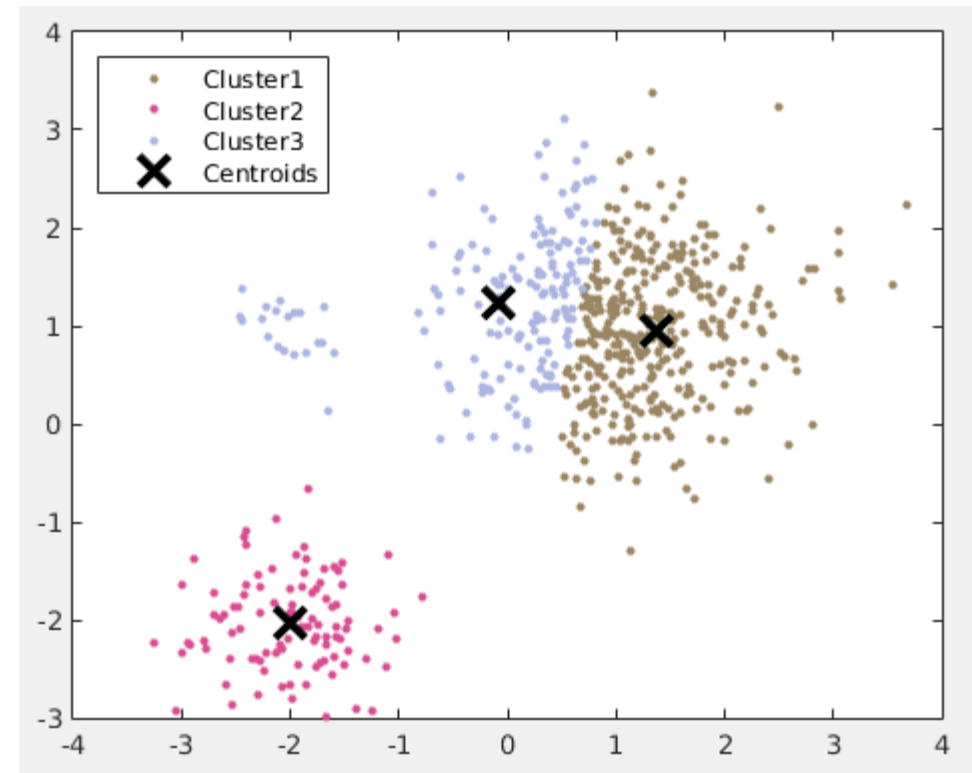
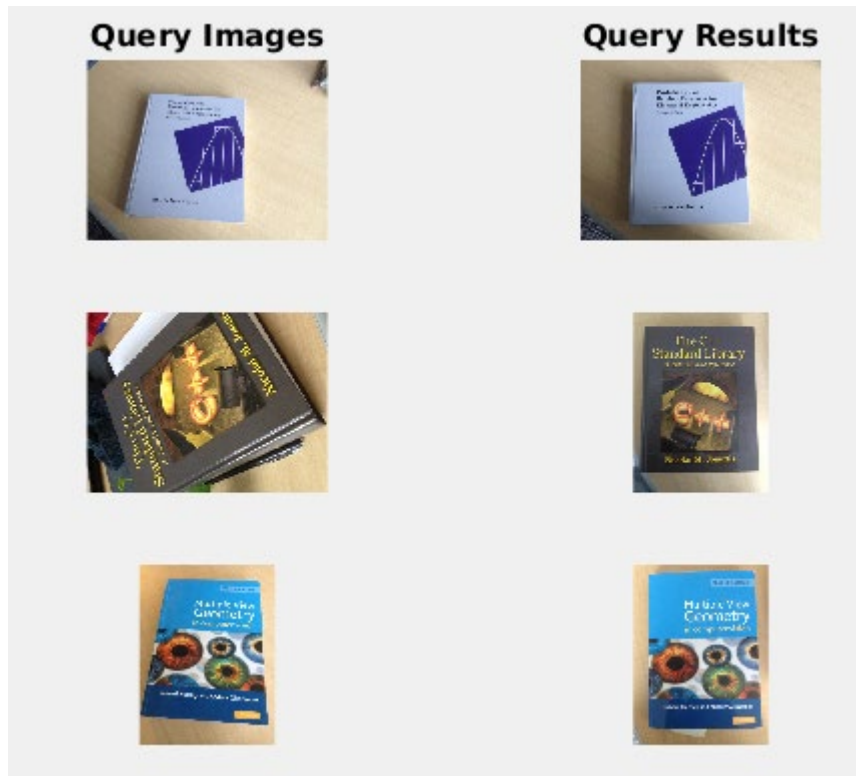
Lecture 12a Place Recognition

Davide Scaramuzza

<http://rpg.ifi.uzh.ch>

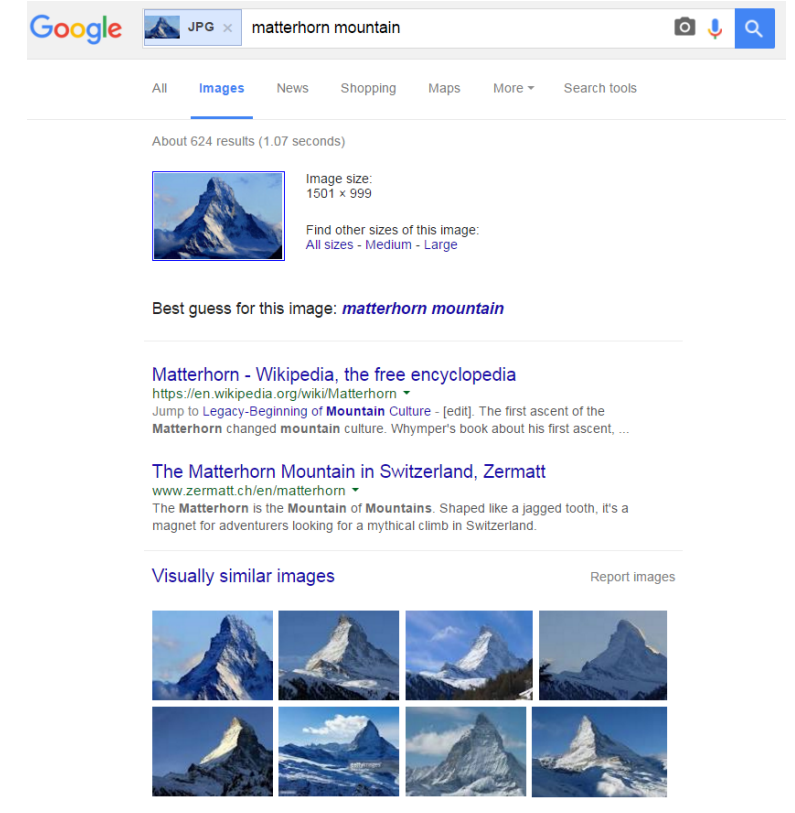
Deep Learning Tutorial Today

- **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

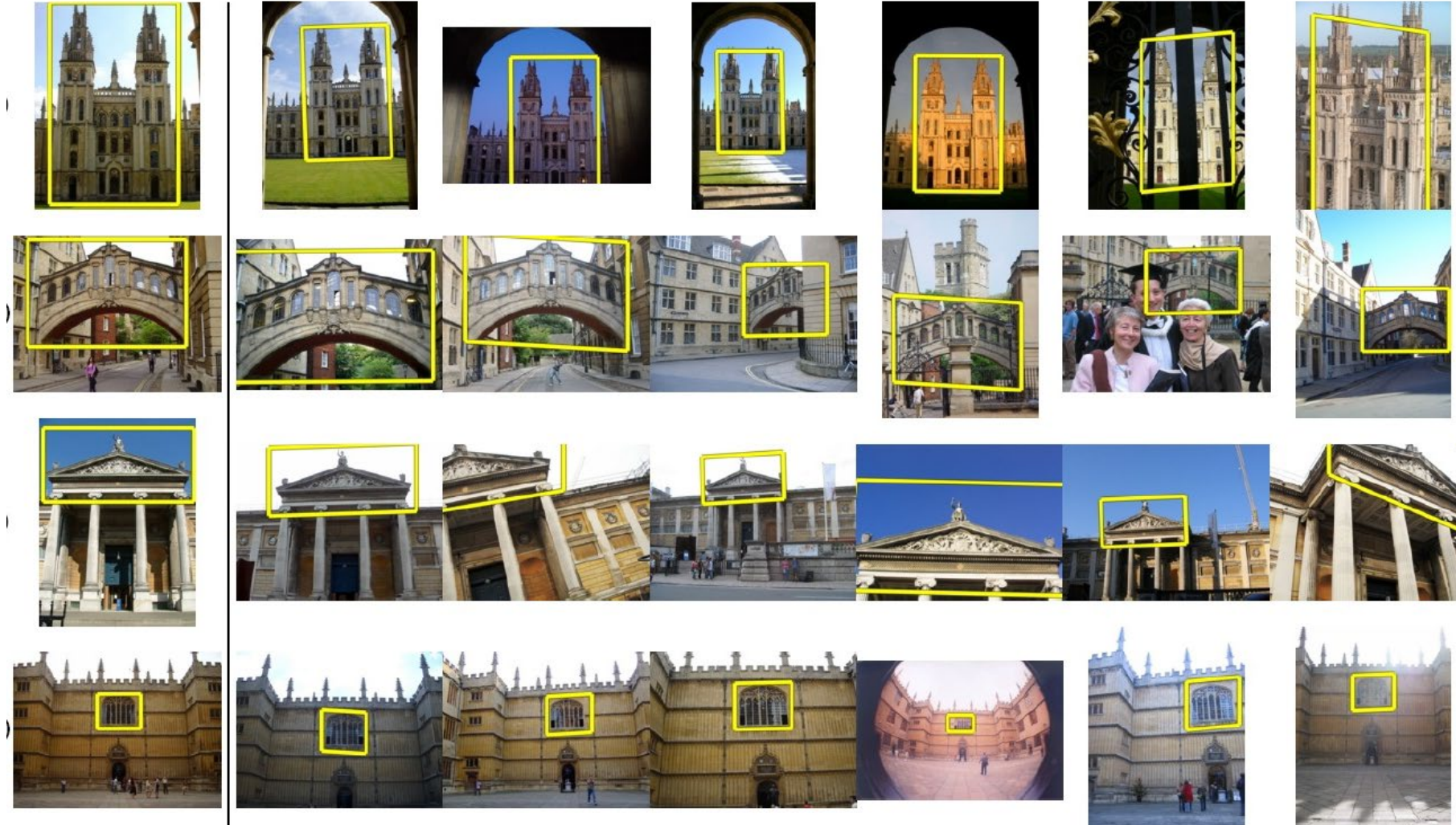
- **Robotics:**
 - Has the robot been to this place before?
 - Which images were taken around the same location?
- **Image retrieval:**
 - Have I seen this image before?
 - Which images in my database look similar to it?
E.g., Google Reverse Image Search



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 N images
- **Complexity:** $O(NF^2)$ feature comparisons (assumes each image has F features)
 - Example:
 - assume 1,000 SIFT features per image $\rightarrow F = 1,000$
 - assume $N = 100,000,000$
 - $\rightarrow NF^2 = 100,000,000,000,000$ feature comparisons!
 - If we assume 10 microseconds per feature comparison \rightarrow 1 image query would take **32 years!**

Solution: Use an index file! Complexity reduces to $O(F)$

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](#).¹⁰

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					
A L P H A B E T I C A L I N D E X	Alphabetical Index				
	A Aperture, 103 Arm, Follower, 59 Assembly, Bolt, 83-94 Assembly, Follower Rod, 60-64 Assembly, Operating Rod, 65-69 Assembly, Stock, 51-53 Assembly, Trigger, 26-27	Ejector, Clip, 9 Exterior View, 2 Extractor, 89 Extractor Spring, 93 Extractor Spring Plunger, 91	I J K Knob, Windage, Rear Sight, 106-110	Q R Rear Hand Guard, 38-39 Rear Hand Guard Band, 32 Rear Sight Base, 104 Rear Sight Cover, 105 Rear Sight Elevating Pinion, 111-119 Rear Sight Group, 101-120 Rear Sight Windage Knob, 106-110 Receiver, 77-80 Rifle, U.S. CAL. .30, M1, 1-6	T Trigger, 26-27 Trigger Guard, 10-13 Trigger Housing, 17-18 Trigger Housing Assembly, 7-28 Trigger Pin, 21
	B Band, Lower, 31 Band, Rear Hand Guard, 32 Barrel, 70-71 Base, Rear Sight, 104 Bolt, 85-87 Bolt Assembly, 83-94 Bullet Guide, 75 Butt Plate, 43-44 Butt Plate Cap, 33 Butt Plate Cap Pin, 40 Butt Plate Long Screw, 46 Butt Plate Plunger, 45 Butt Plate Plunger Spring, 50 Butt Swivel, 54	F Ferrule, Front Hand Guard, 34 Ferrule, Stock, 35 Firing Pin, 90 Follower, 74 Follower Arm, 59 Follower Group, 57-82 Follower Rod Assembly, 60-64 Follower Rod Pin, 76 Follower Slide, 81 Front Hand Guard, 36-37 Front Hand Guard Ferrule, 34 Front Hand Guard Spacer, 49 Front Sight, 131 FSN Crossover, 136-138	L Latch Group, 95-100 Latch, Clip, 97 Lock, Gas Cylinder, 124 Longitudinal Section, 3 Lower Band, 31 Lower Band Pin (Spring Pin), 41-42	S Safety, 23 Screw, Butt Plate Long, 46 Screw, Cap, Socket Head, Hexagon, 125 Screw, Gas Cylinder, w/Valve Assembly, 126-129 Screw, Stacking Swivel, 130 Screw, Stock Ferrule, 47 Screw, Wood Slotted Oval Head, Ninety Degree, 48 Sear, 24 Sear Pin, 20 Sectionalized Views, 4-5 Sight, Front, 131 Slide, Follower, 81 Sling Swivel, 55 Spacer, Front Hand Guard, 49 Spring, Butt Plate Plunger, 50 Spring, Clip Latch, 99 Spring, Ejector (Cartridge), 92 Spring, Extractor, 93 Spring, Hammer, 25 Spring, Operating Rod, 82 Stacking Swivel, 132 Stacking Swivel Screw, 130 Stock, 51-53 Stock Assembly & Handguard, 29-56 Stock Ferrule, 35 Stock Ferrule Screw, 47 Swivel, Butt, 54 Swivel, Sling, 55 Swivel, Stacking, 132	W Windage Knob, Rear Sight, 106-110 Wood Slotted Screw, Oval Head, Ninety Degree, 48
	C Cap Screw, Hexagon Socket Head, 125 Cap, Butt Plate, 33 Catch, Operating Rod, 72-73 Clip Ejector, 9 Clip Latch, 97 Clip Latch Pin, 98 Clip Latch Spring, 99 Cover, Rear Sight, 105 Cylinder, Gas, 123	G Gas Cylinder, 123 Gas Cylinder Group, 121-132 Gas Cylinder Lock, 124 Gas Cylinder Screw w/Valve Assembly, 126-129 Geometric Symbols, 134 Guard, Hand, Front, 36-37 Guard, Hand, Rear, 38-39 Guard, Trigger, 10-13 Guide, Bullet, 75	M Manufacturer Markings, 135	X	
	D	H Hammer, 14-15 Hammer Pin, 19 Hammer Spring, 25 Hammer Spring Housing, 16 Hammer Spring Plunger, 22 Housing, Hammer Spring, 16 Housing, Trigger, 17-18	N	Y	
	E Ejector (Cartridge), 88 Ejector Spring (Cartridge), 92		O Operating Rod Assembly, 65-69 Operating Rod Catch, 72-73 Operating Rod Spring, 82	Z	
			P Parts Crossover, 136-138 Parts Markings, 135 Pin, Butt Plate Cap, 40 Pin, Clip Latch, 98 Pin, Firing, 90 Pin, Follower Rod, 76 Pin, Hammer, 19 Pin, Lower Band (Spring Pin), 41-42 Pin, Sear, 20 Pin, Trigger, 21 Pinion, Rear Sight Elevating, 111-119 Plate, Butt, 43-44 Plunger, Butt Plate, 45 Plunger, Extractor Spring, 91 Plunger, Hammer Spring, 22		

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

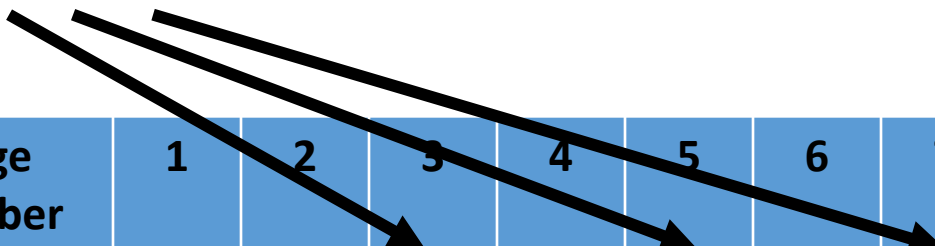
- The solution is to use a **voting array** that has as many cells as the number of pages in the document
- We first set all the cell values to 0
- Then, we add 1 to each cell corresponding to the page numbers where the words of the query text appear according to the index file

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}



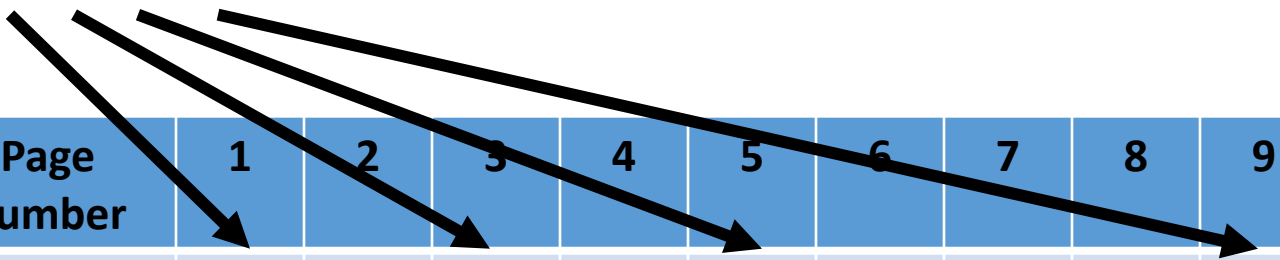
Page number	1	2	3	4	5	6	7	8	9	10
Cell value	0	0	1	0	1	0	1	0	0	0

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}

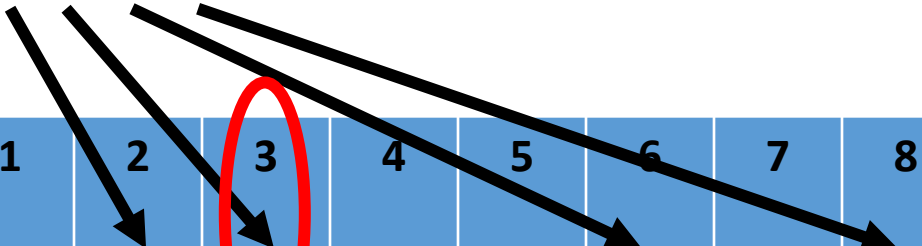
Page number	1	2	3	4	5	6	7	8	9	10
Cell value	1	0	2	0	2	0	1	0	1	0



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)

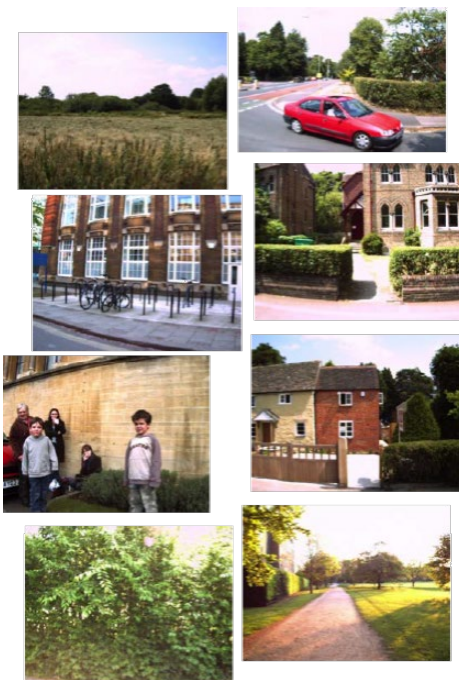
- Can a SIFT descriptor be used as a visual word? And a BRISK descriptor?
 - SIFT $\rightarrow 128 \times 4 \text{ bytes float} = 512 \text{ bytes} = 4096 \text{ bits} = 2^{4096}$ possible SIFT descriptors!
 - BRISK-128 $\rightarrow 128 \text{ bits} = 2^{128}$ possible BRISK descriptors!
 - **Usually, 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

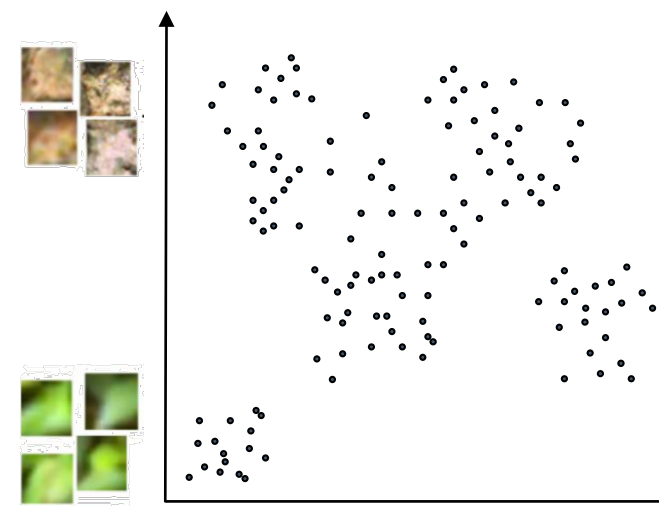
Image database
(e.g., 100 million images)



Feature extraction
(~1,000 features per image)

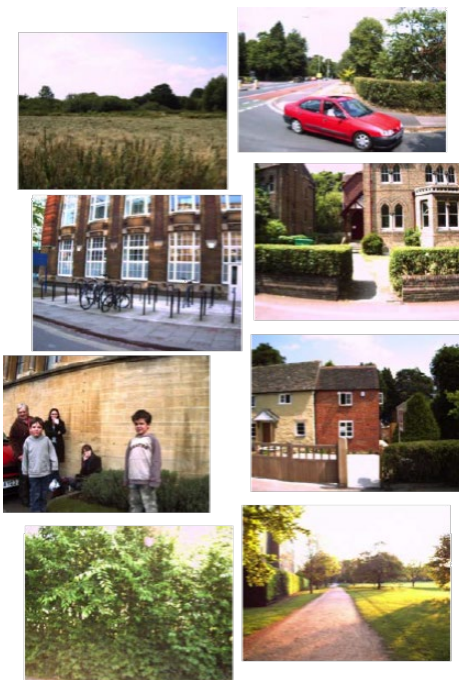


Map all features into the descriptor space
(~100 billion descriptors)



Extracting Visual Words

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



Feature extraction
(~1,000 features per image)

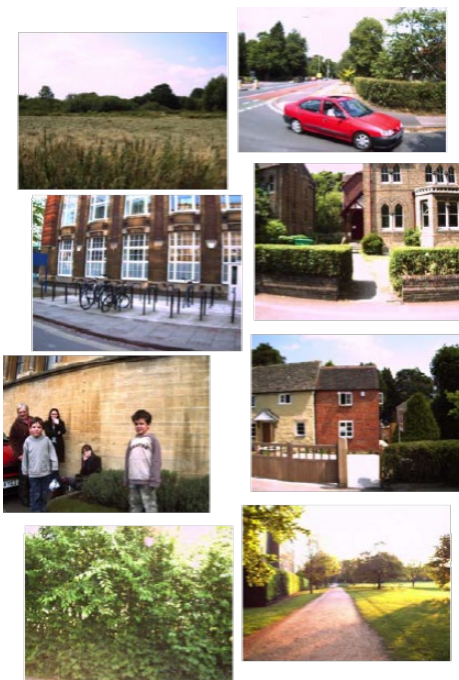


Feature clustering
(~1 million words)

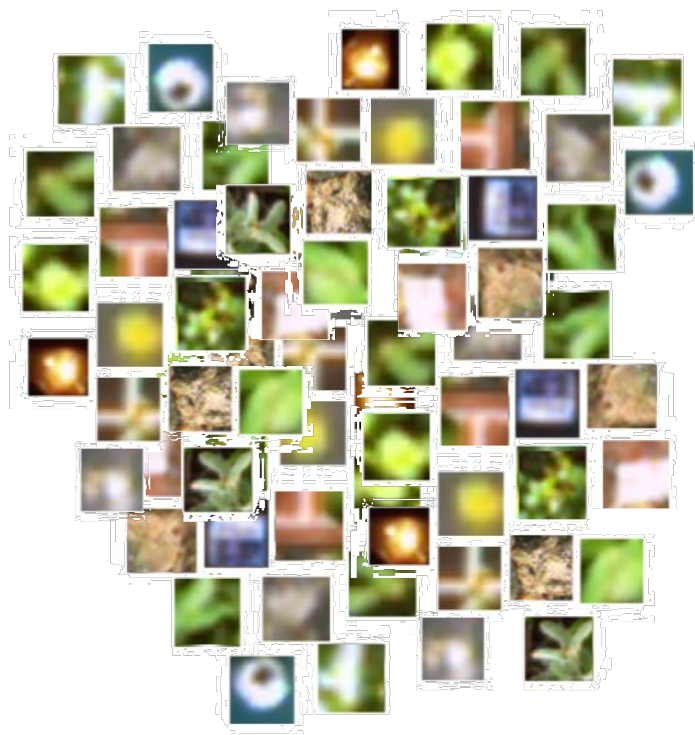


Extracting Visual Words

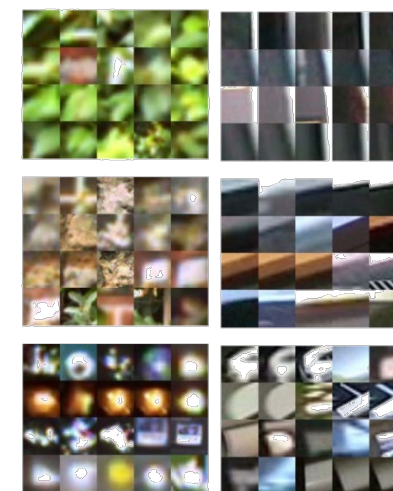
Image database
(e.g., 100 million images)



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**

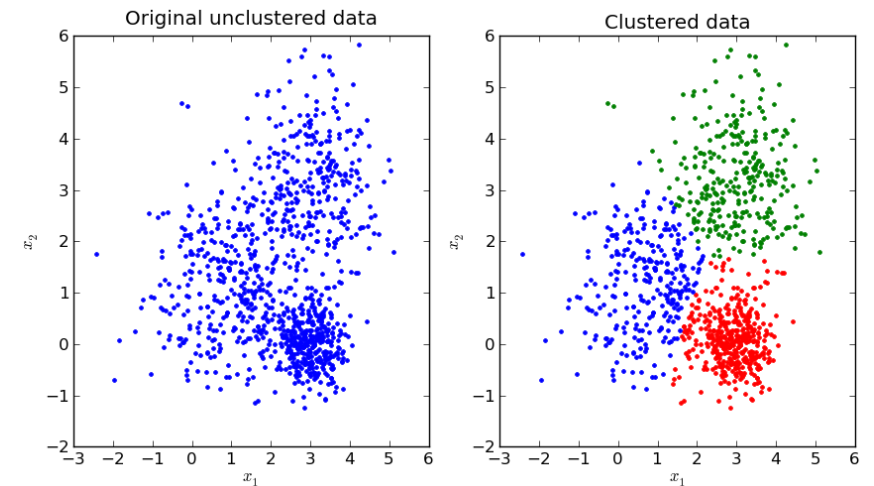
How do we cluster the descriptor space?

- **k-means clustering** is an algorithm to partition n data points into k clusters in which each data point x belongs to the cluster S_i with center m_i
- It minimizes the **squared Euclidean distance** between points x and their nearest cluster centers 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 x_j to the nearest center 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

ALPHABETICAL INDEX	Alphabetical Index	RIFLE, U.S. CAL. .30, M1: DIAGRAMS & PICTURES		
	A Aperture, 103 Arm, Follower, 59 Assembly, Bolt, 83-94 Assembly, Follower Rod, 60-64 Assembly, Operating Rod, 65-69 Assembly, Stock, 51-53 Assembly, Trigger, 26-27	E Ejector, Clip, 9 Exterior View, 2 Extractor, 89 Extractor Spring, 93 Extractor Spring Plunger, 91	I ↓ K Knob, Windage, Rear Sight, 106-110	Q R Rear Hand Guard, 38-39 Rear Hand Guard Band, 32 Rear Sight Base, 104 Rear Sight Cover, 105 Rear Sight Elevating Pinion, 111-119 Rear Sight Group, 101-120 Rear Sight Windage Knob, 106-110 Receiver, 77-80 Rifle, U.S. CAL. .30, M1, 1-6
	B Band, Lower, 31 Band, Rear Hand Guard, 32 Barrel, 70-71 Base, Rear Sight, 104 Bolt, 85-87 Bolt Assembly, 83-94 Bullet Guide, 75 Butt Plate, 43-44 Butt Plate Cap, 33 Butt Plate Cap Pin, 40 Butt Plate Long Screw, 46 Butt Plate Plunger, 45 Butt Plate Plunger Spring, 50 Butt Swivel, 54	F Ferrule, Front Hand Guard, 34 Ferrule, Stock, 35 Firing Pin, 90 Follower, 74 Follower Arm, 59 Follower Group, 57-82 Follower Rod Assembly, 60-64 Follower Rod Pin, 76 Follower Slide, 81 Front Hand Guard, 36-37 Front Hand Guard Ferrule, 34 Front Hand Guard Spacer, 49 Front Sight, 131 FSN Crossover, 136-138	L Latch Group, 95-100 Latch, Clip, 97 Lock, Gas Cylinder, 124 Longitudinal Section, 3 Lower Band, 31 Lower Band Pin (Spring Pin), 41-42	S Safety, 23 Screw, Butt Plate Long, 46 Screw, Cap, Socket Head, Hexagon, 125 Screw, Gas Cylinder, w/Valve Assembly, 126-129 Screw, Stacking Swivel, 130 Screw, Stock Ferrule, 47 Screw, Wood Slotted Oval Head, Ninety Degree, 48 Sear, 24 Sear Pin, 20 Sectionalized Views, 4-5 Sight, Front, 131 Slide, Follower, 81 Sling Swivel, 55 Spacer, Front Hand Guard, 49 Spring, Butt Plate Plunger, 50 Spring, Clip Latch, 99 Spring, Ejector (Cartridge), 92 Spring, Extractor, 93 Spring, Hammer, 25 Spring, Operating Rod, 82 Stacking Swivel, 132 Stock Assembly & Handguard, 29-56 Stock Ferrule, 35 Stock Ferrule Screw, 47 Swivel, Butt, 54 Swivel, Sling, 55 Swivel, Stacking, 132
	C Cap Screw, Hexagon Socket Head, 125 Cap, Butt Plate, 33 Catch, Operating Rod, 72-73 Clip Ejector, 9 Clip Latch, 97 Clip Latch Pin, 98 Clip Latch Spring, 99 Cover, Rear Sight, 105 Cylinder, Gas, 123	G Gas Cylinder, 123 Gas Cylinder Group, 121-132 Gas Cylinder Lock, 124 Gas Cylinder Screw w/Valve Assembly, 126-129 Geometric Symbols, 134 Guard, Hand, Front, 36-37 Guard, Hand, Rear, 38-39 Guard, Trigger, 10-13 Guide, Bullet, 75	M Manufacturer Markings, 135	T Trigger, 26-27 Trigger Guard, 10-13 Trigger Housing, 17-18 Trigger Housing Assembly, 7-28 Trigger Pin, 21
	D E Ejector (Cartridge), 88 Ejector Spring (Cartridge), 92	H Hammer, 14-15 Hammer Pin, 19 Hammer Spring, 25 Hammer Spring Housing, 16 Hammer Spring Plunger, 22 Housing, Hammer Spring, 16 Housing, Trigger, 17-18	N O Operating Rod Assembly, 65-69 Operating Rod Catch, 72-73 Operating Rod Spring, 82	W Windage Knob, Rear Sight, 106-110 Wood Slotted Screw, Oval Head, Ninety Degree, 48

Image vocabulary

Visual words List of images in which each word appears

0	→	
...	→	
101	→	
102	→	
103	→	
104	→	
105	→	
...	→	

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

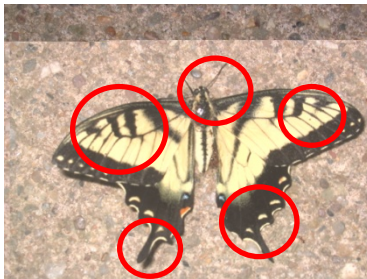
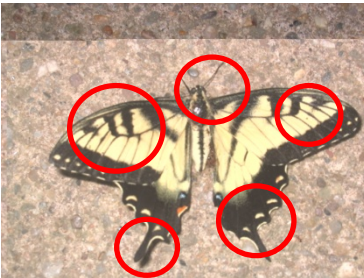


Image Retrieval

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



Voting Array for Q

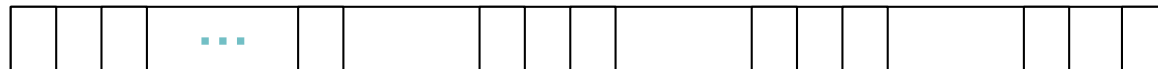


Image Retrieval

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

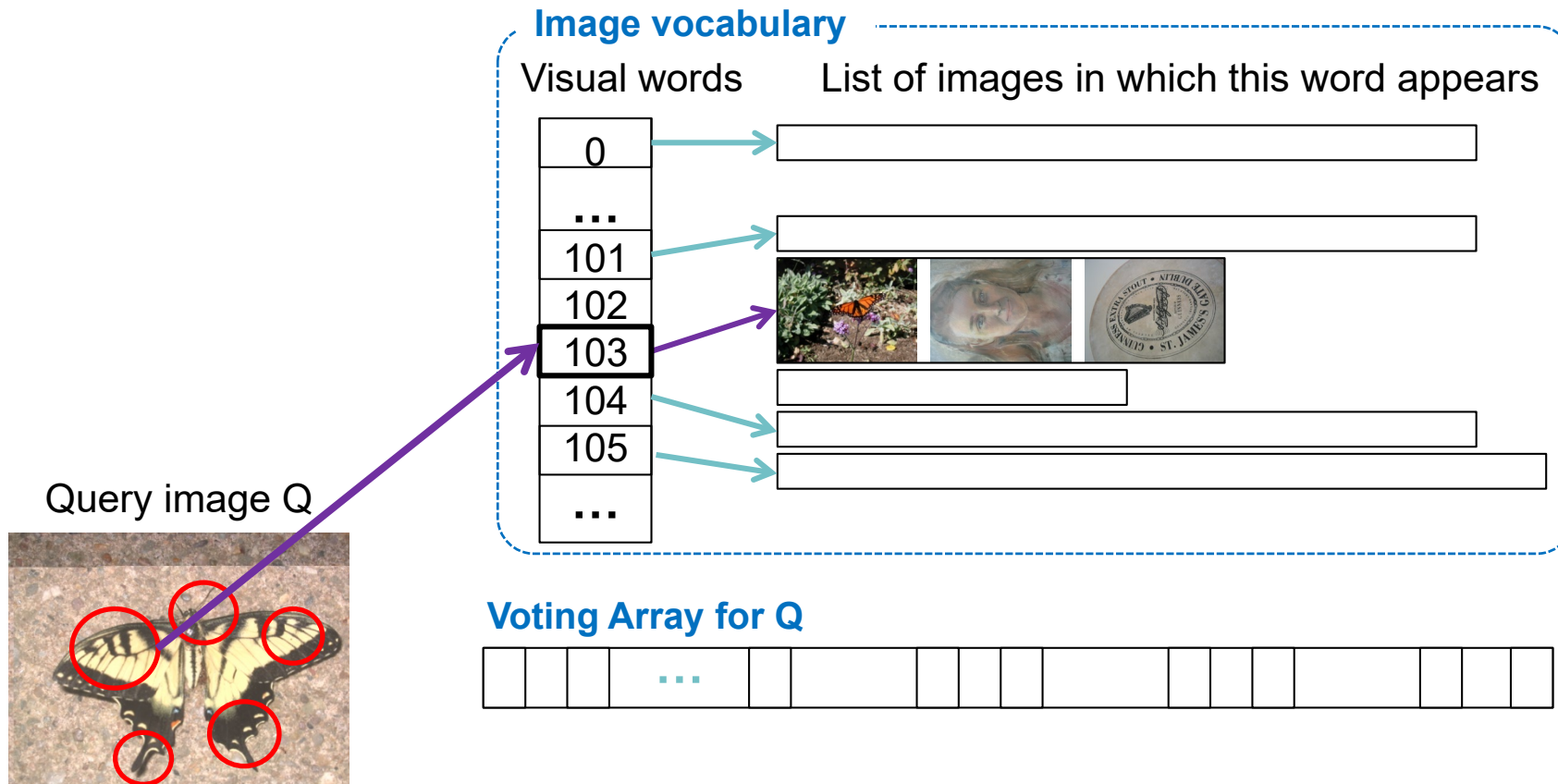
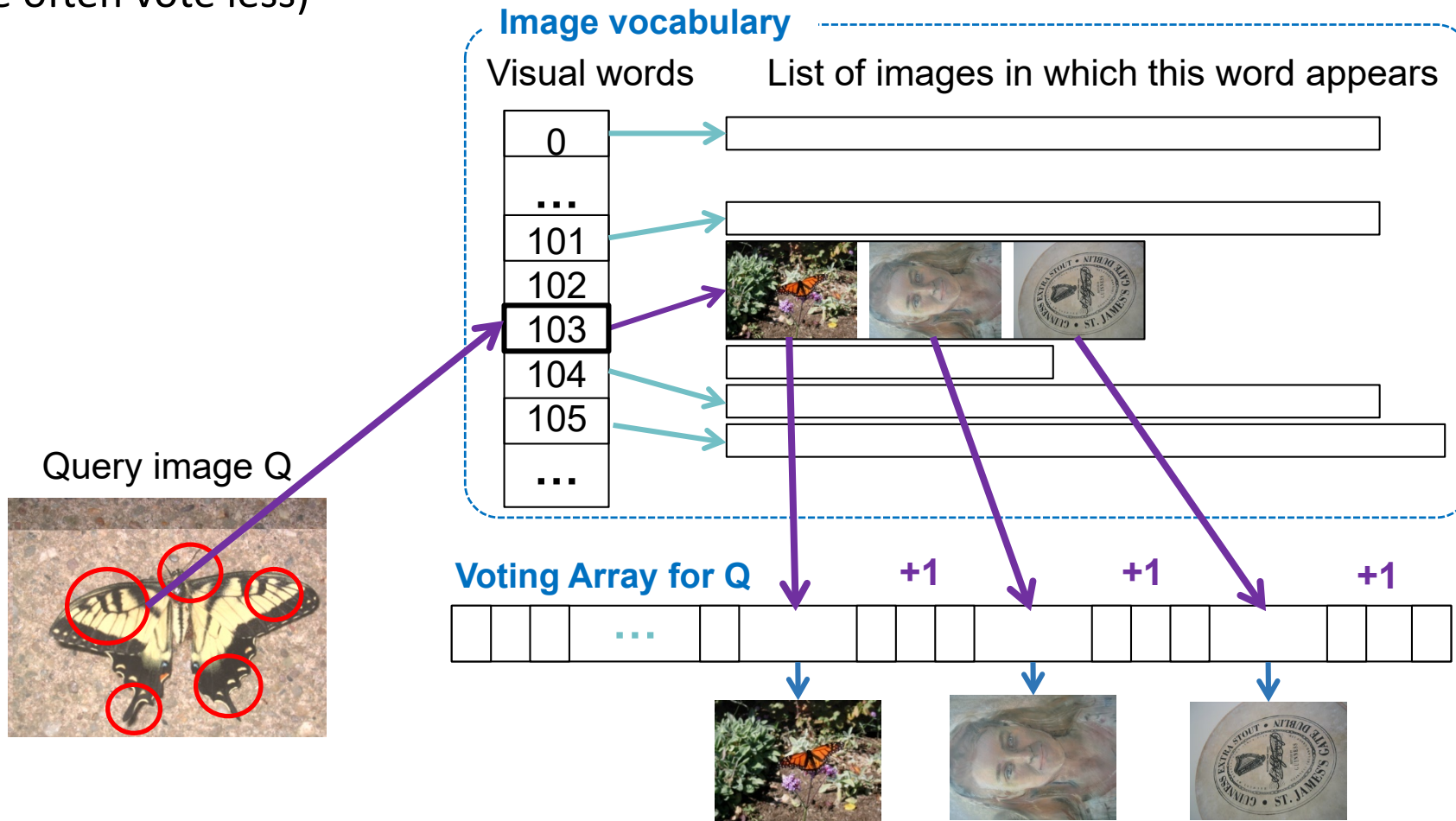


Image Retrieval

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:

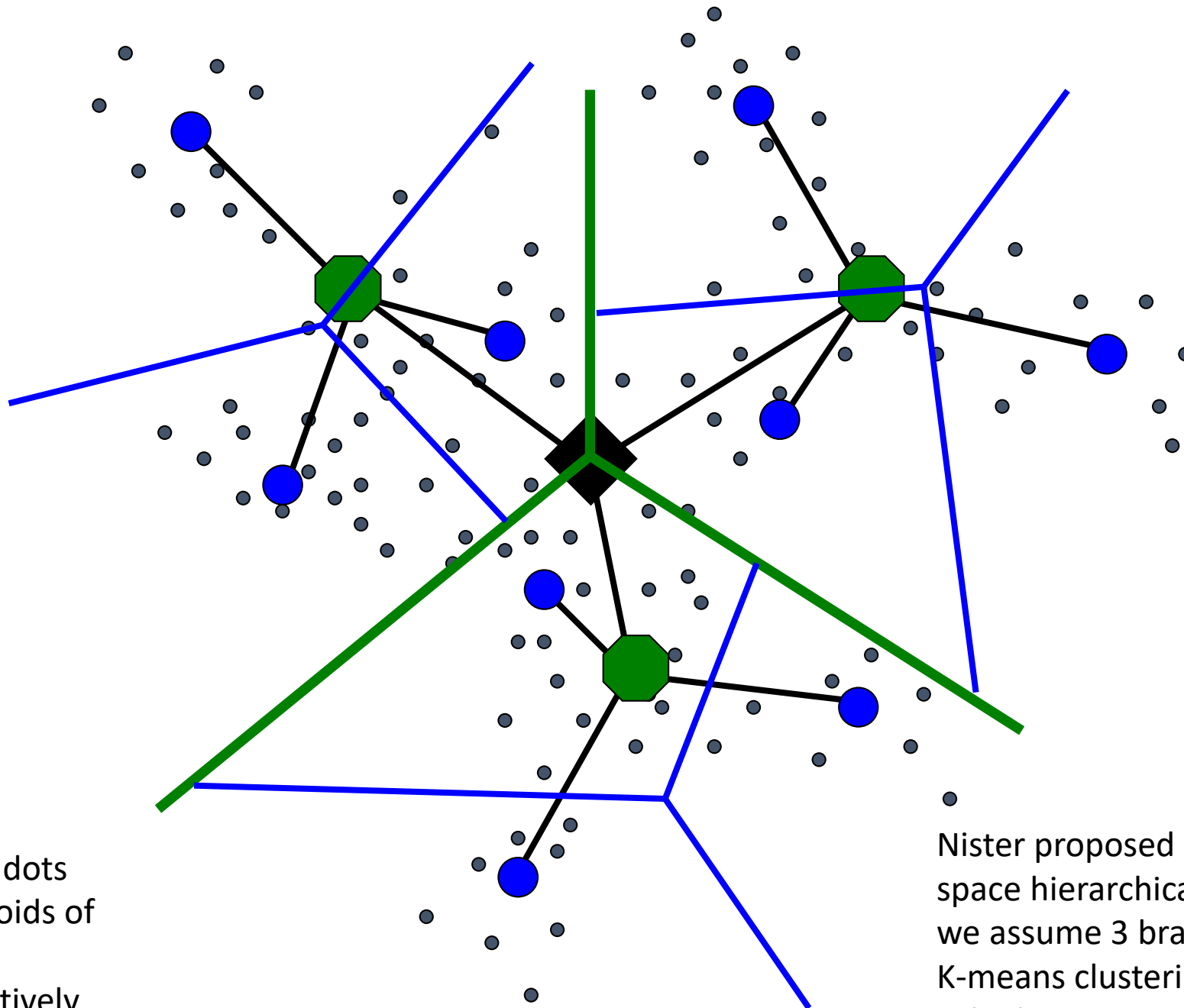
- Example:
 - assume our query image has 1,000 features;
 - assume 1 million visual words → number of feature comparisons would be equal to **1 billion!**
 - If we assume 10 microseconds per feature comparison, then querying one image would take **~3 hours!**
- **How can we make the comparison cheaper?**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

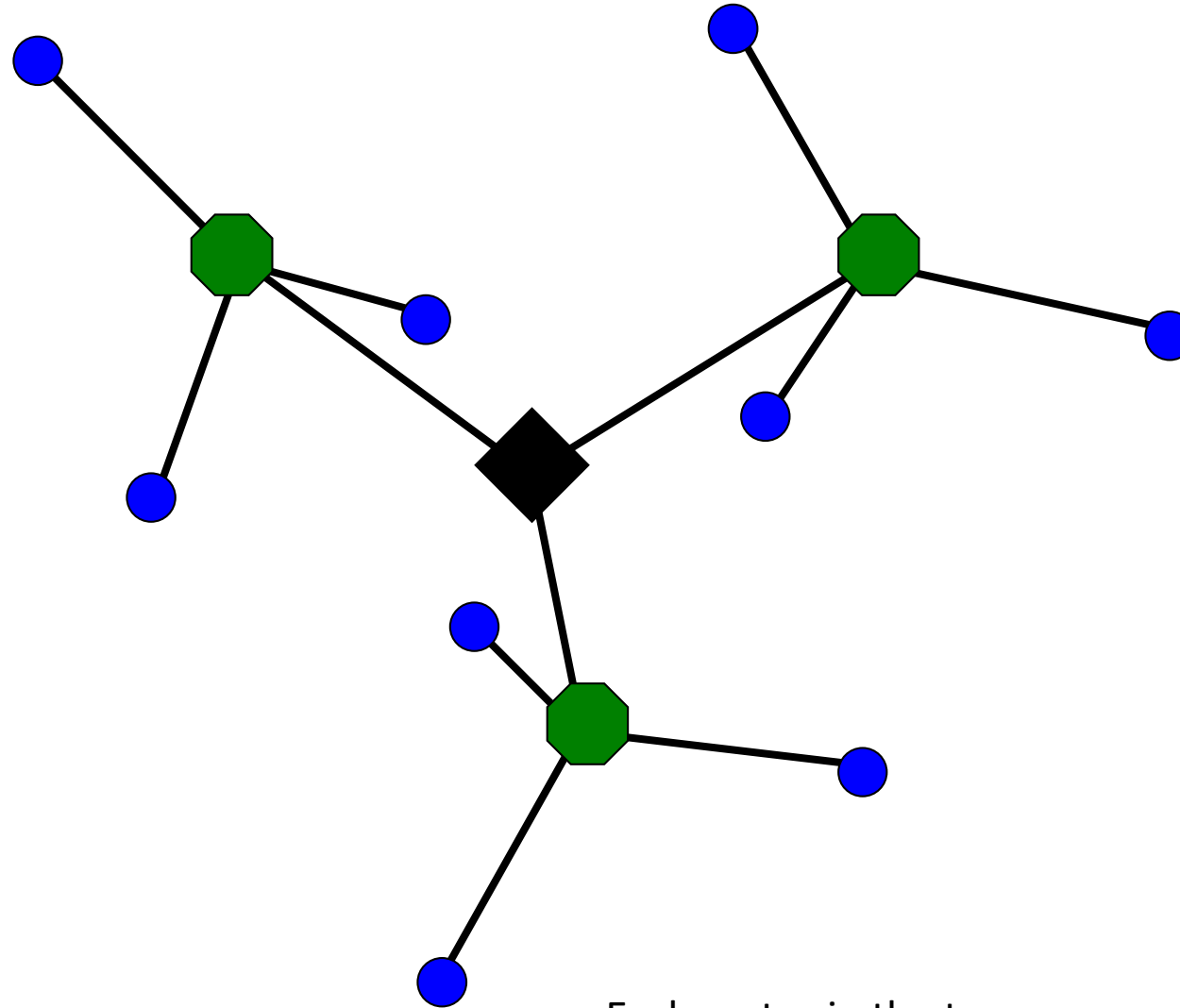


Each dot represents a SIFT feature.
SIFT features have 128 dimensions!
For convenience, here we assume only
2 dimensions 😊

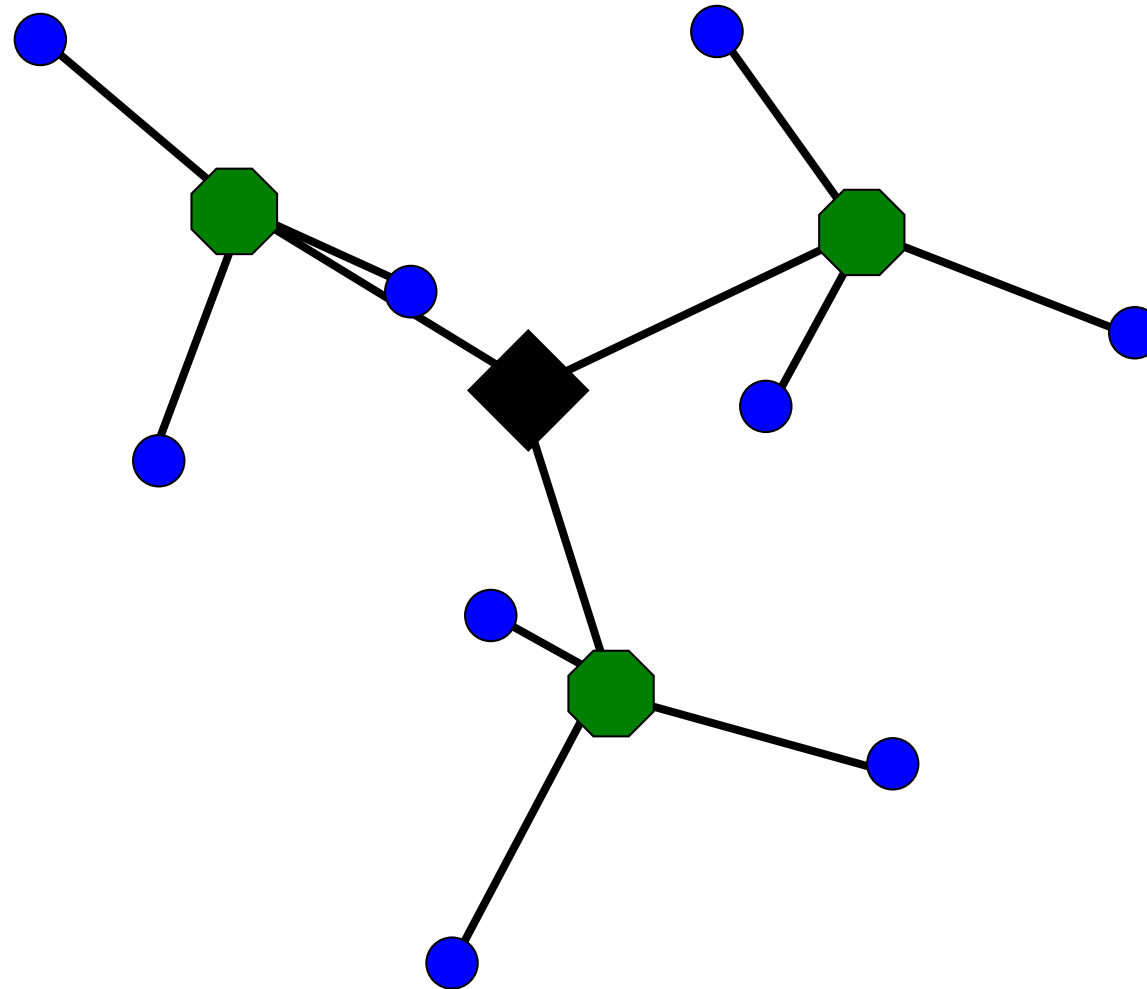


The green and blue dots represent the centroids of level 1 and level 2 sub-clusters, respectively

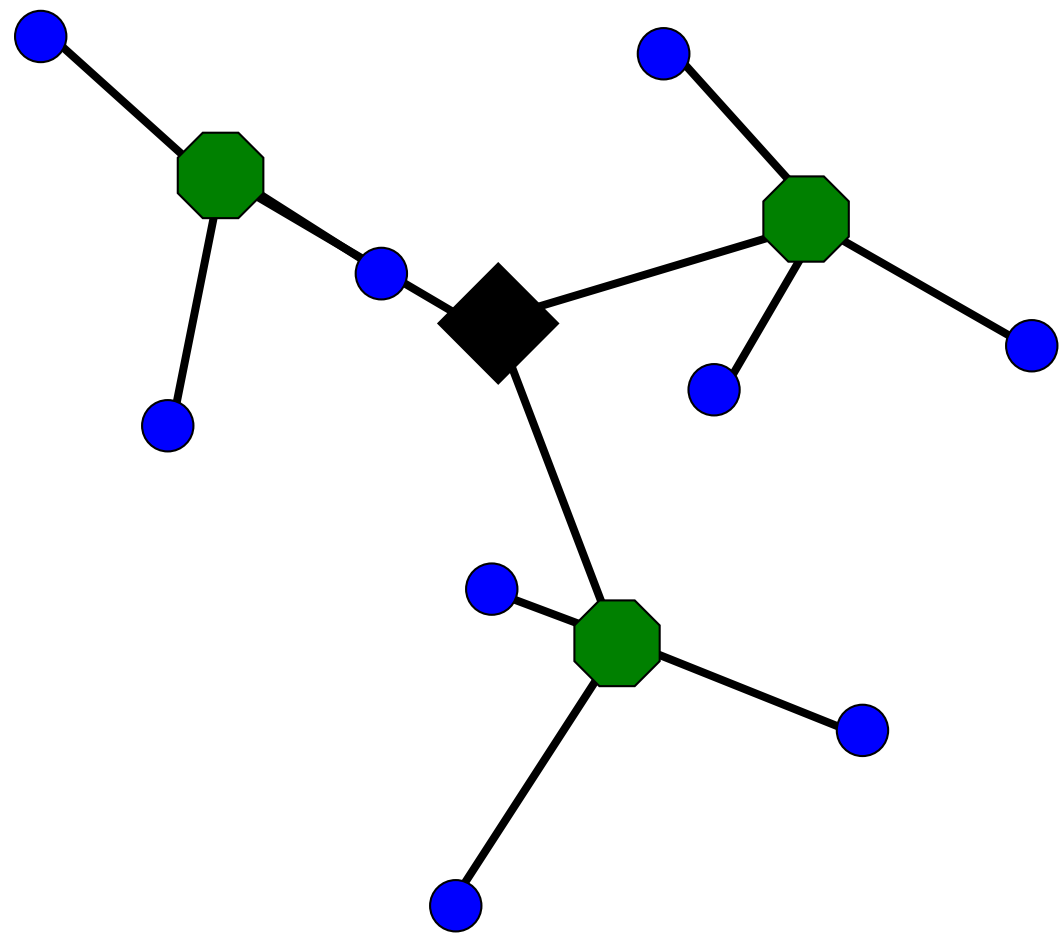
Nister proposed to cluster the feature space hierarchically. For example, here we assume 3 branches and 2 levels. K-means clustering is used to cluster each sub-cluster.



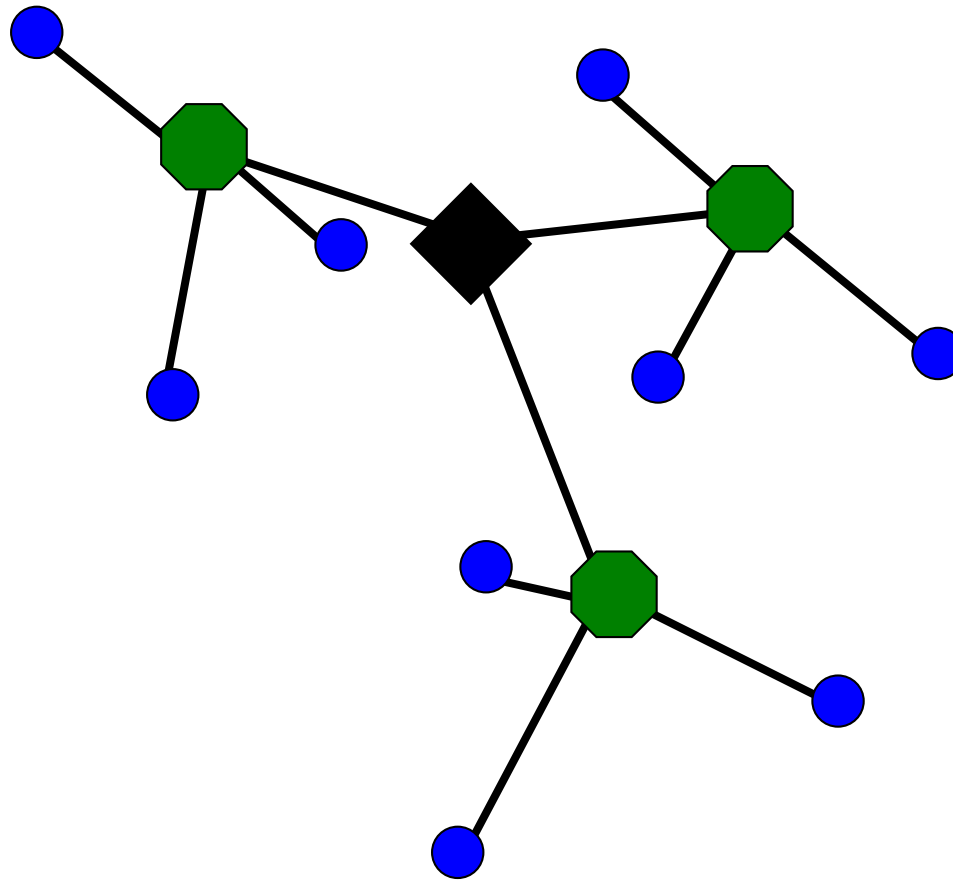
Each vertex in the tree represents a visual word.



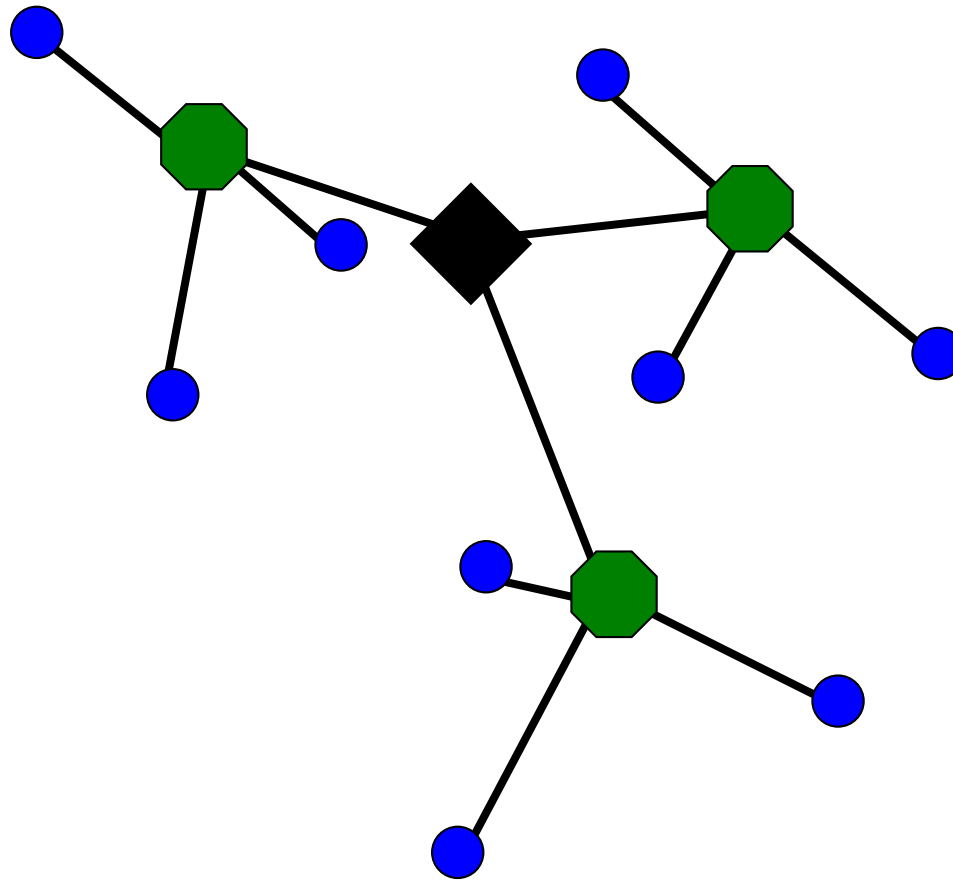
Each vertex in the tree represents a visual word.



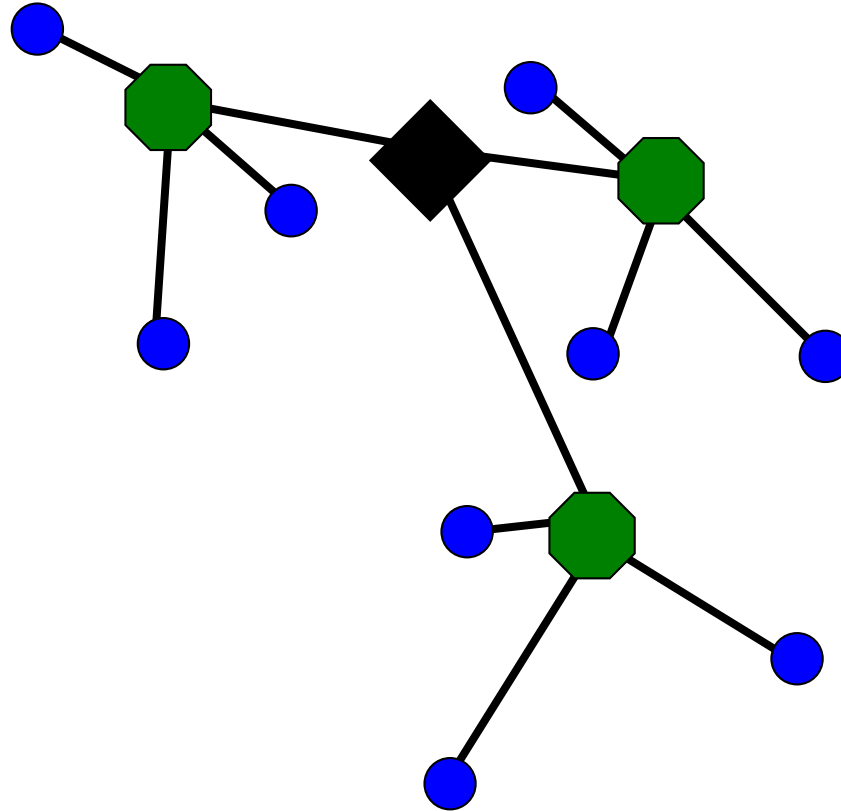
Each vertex in the tree represents a visual word.



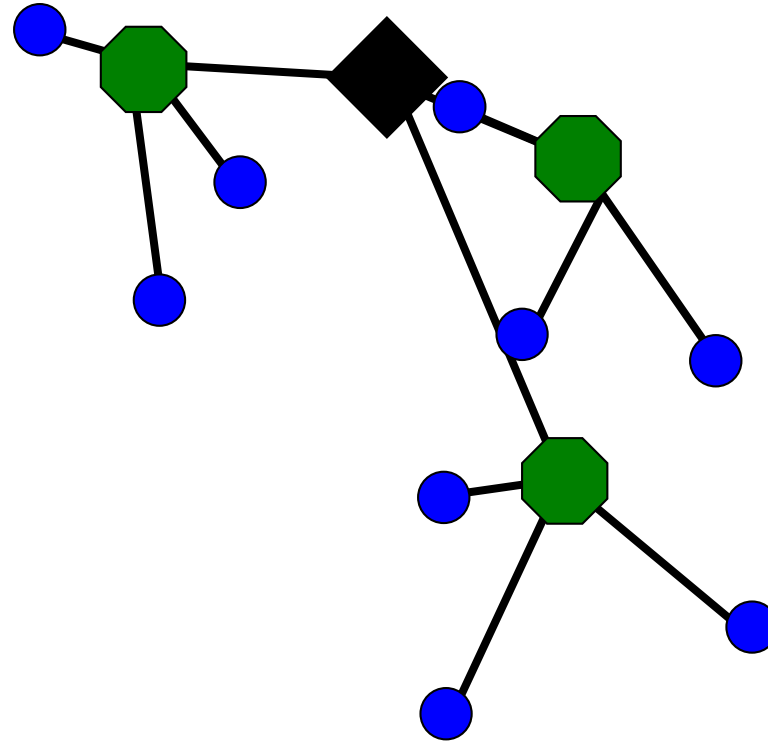
Each vertex in the tree represents a visual word.



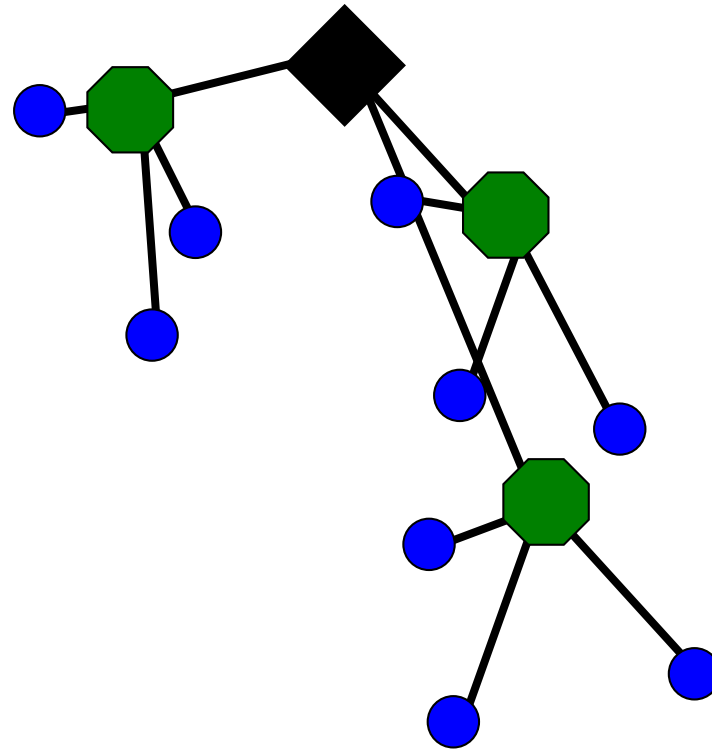
Each vertex in the tree represents a visual word.



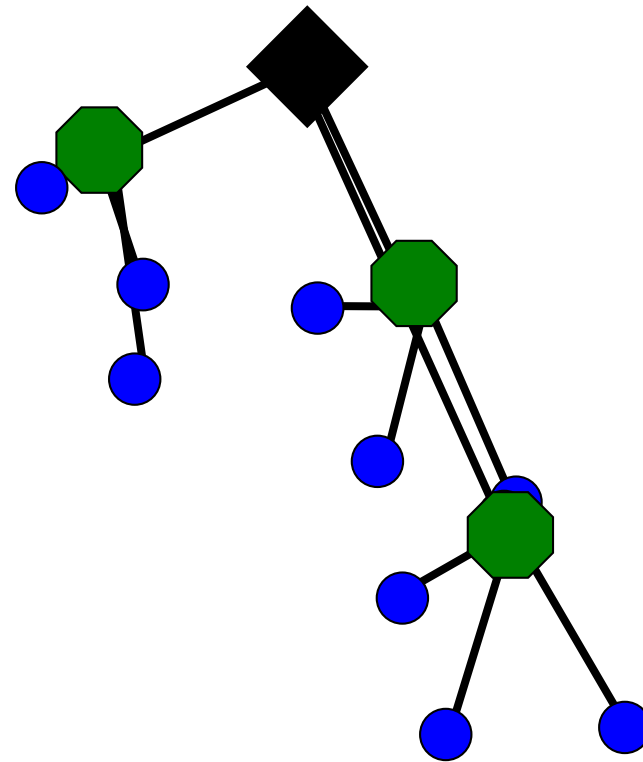
Each vertex in the tree represents a visual word.



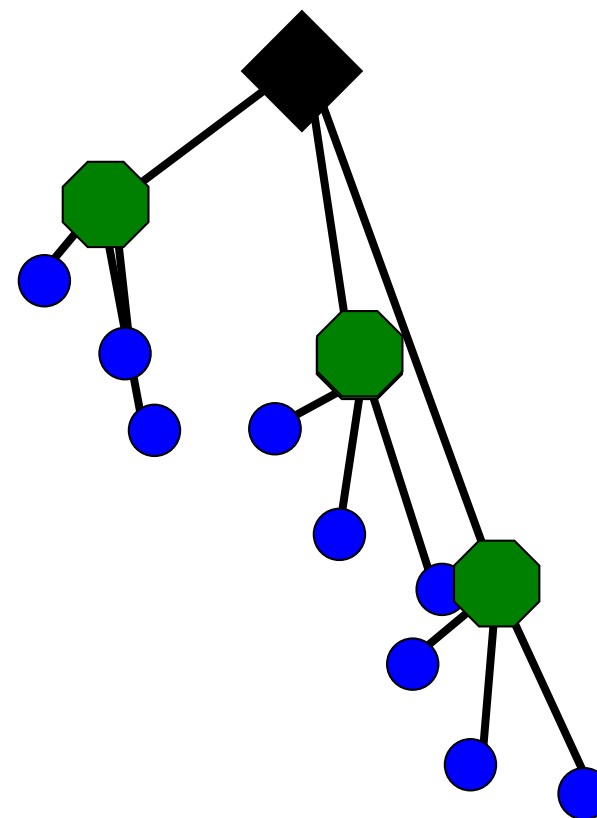
Each vertex in the tree represents a visual word.



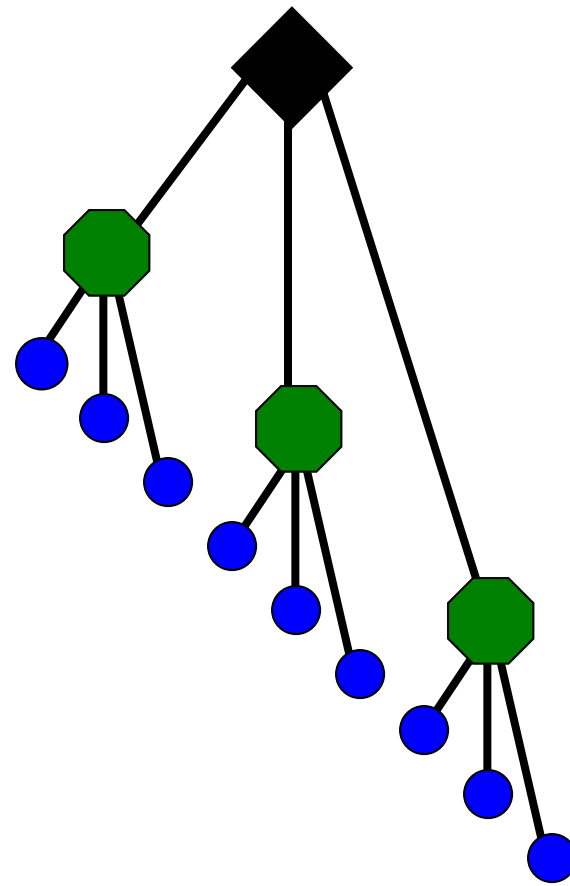
Each vertex in the tree represents a visual word.



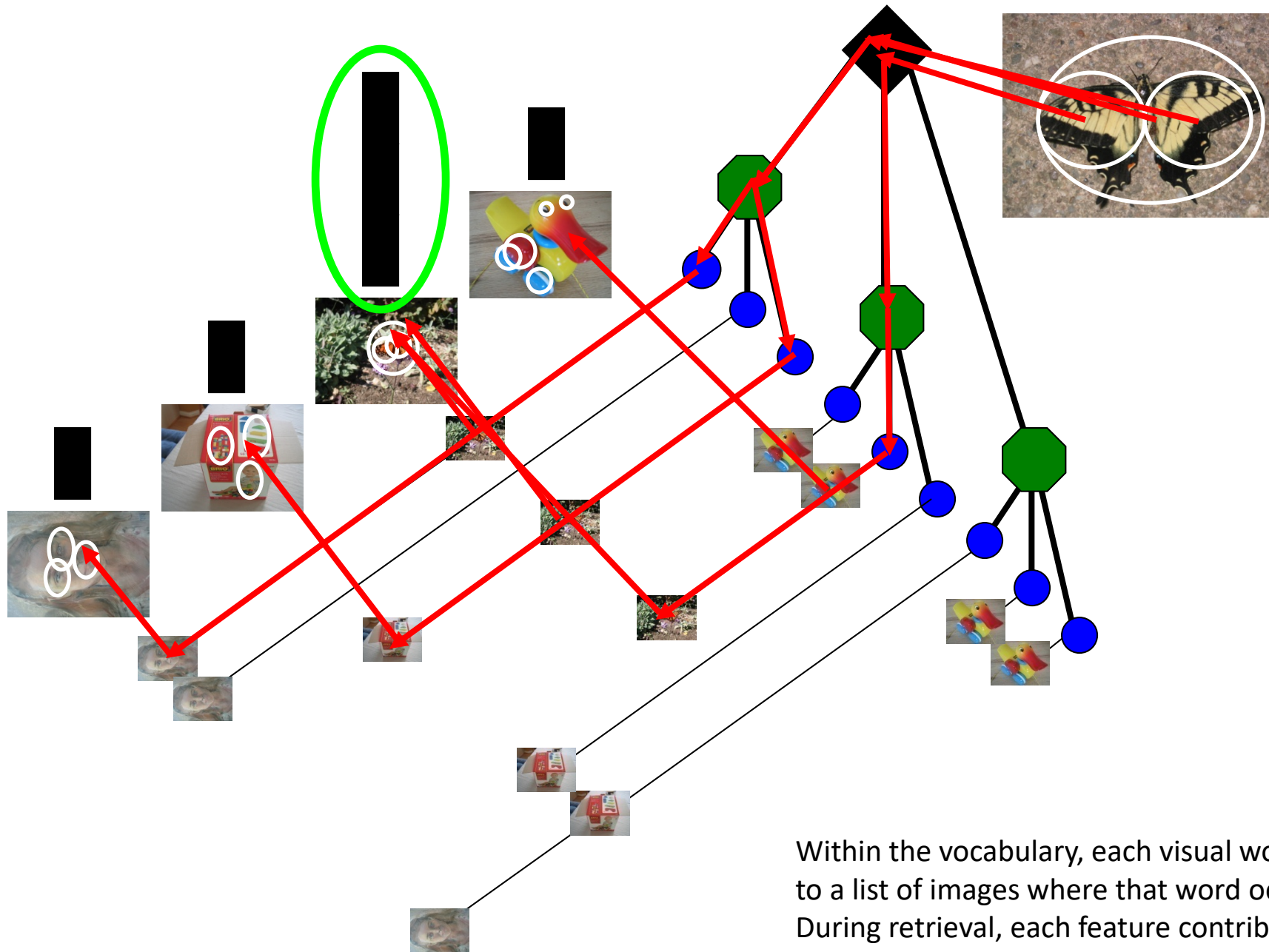
Each vertex in the tree represents a visual word.



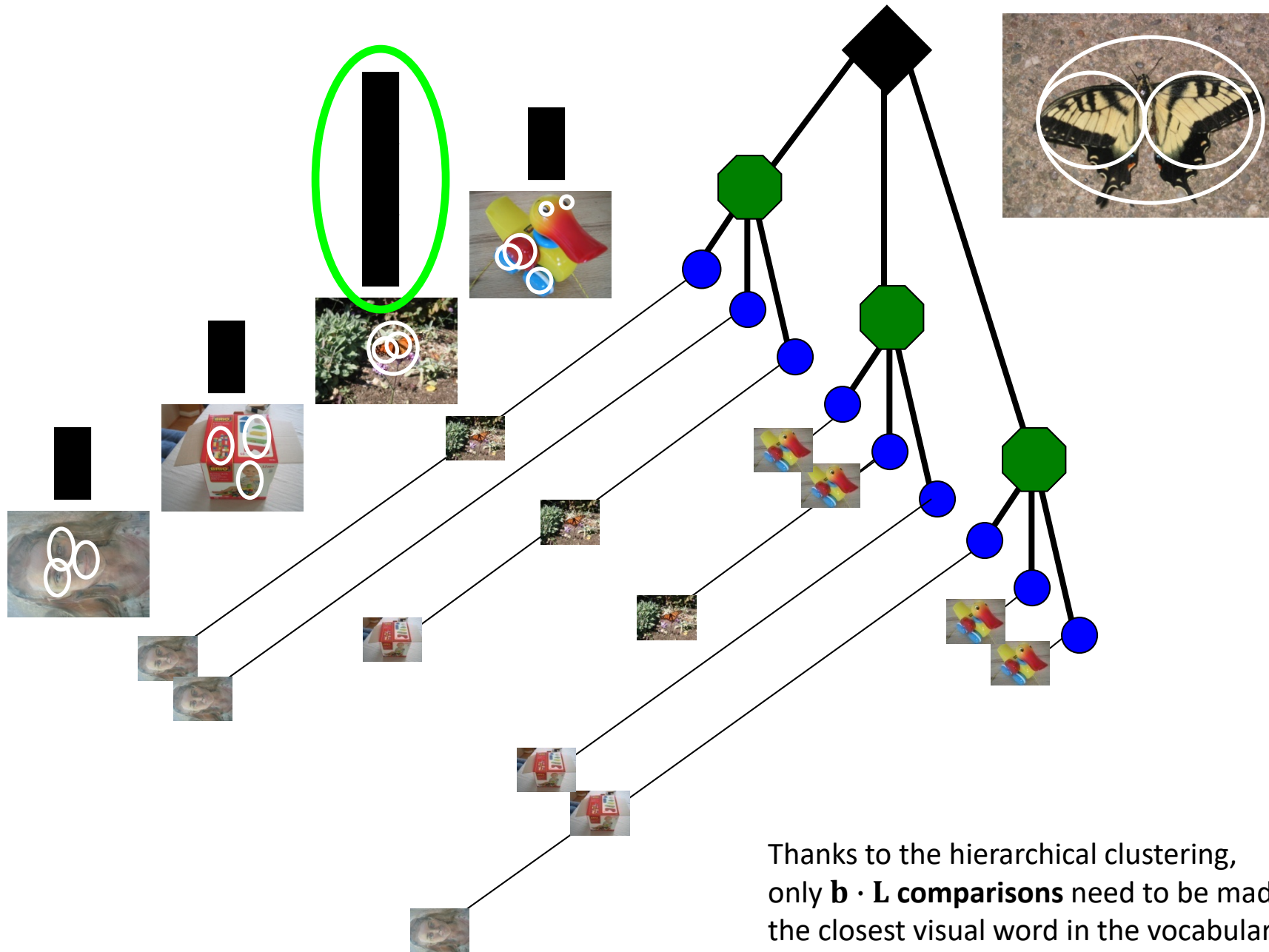
Each vertex in the tree represents a visual word.



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.



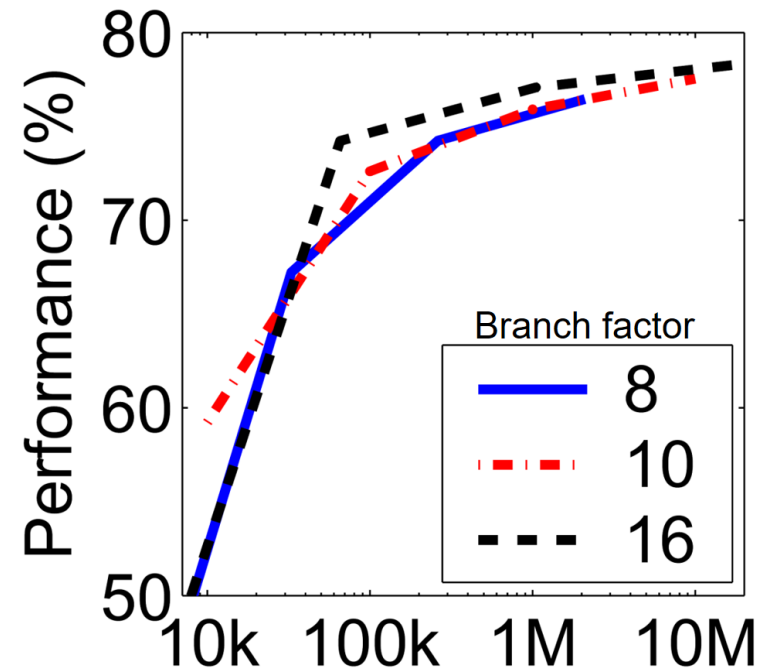
Example

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

- assume a query image with $F = 1,000$ features
- assume a tree structure with $b = 10$ branches per level and $L = 6$ levels (i.e., $b^L = 1,000,000$ visual words)
- Then, the **number of feature comparisons** = $F \cdot b \cdot L = 1,000 \cdot 10 \cdot 6 = 60,000$ **instead of** $F \cdot b^L = 1$ 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)**



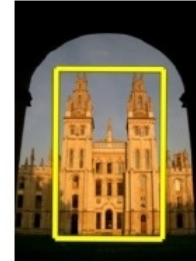
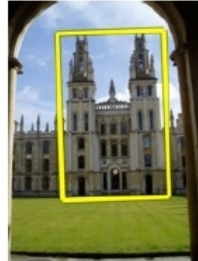
Geometric Verification

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



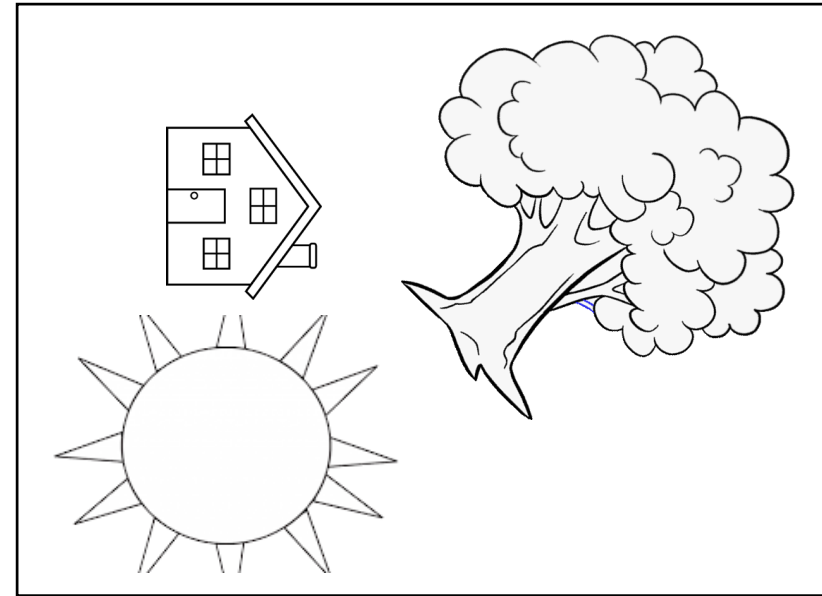
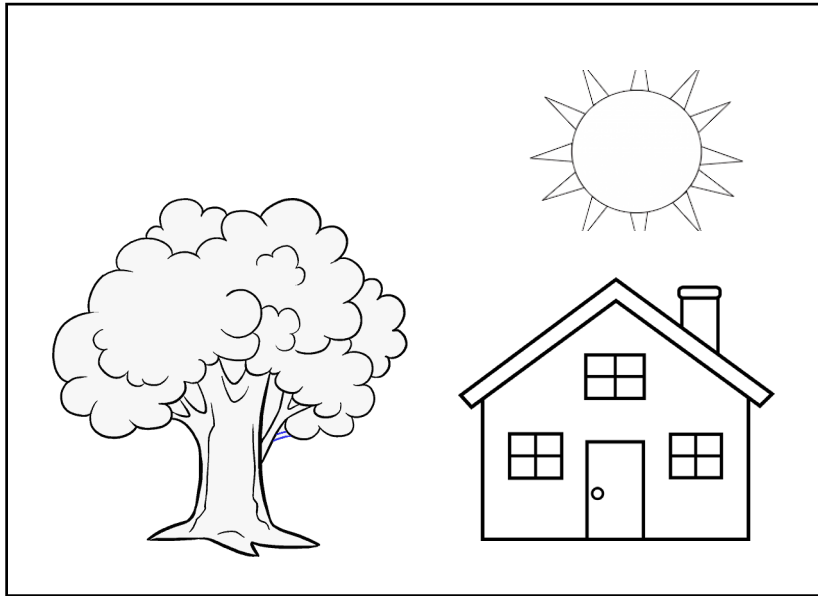
Results with the highest matching score



- 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?