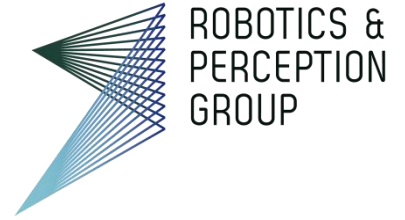




University of
Zurich^{UZH}

ETH zürich

Institute of Informatics – Institute of Neuroinformatics



Lecture 12

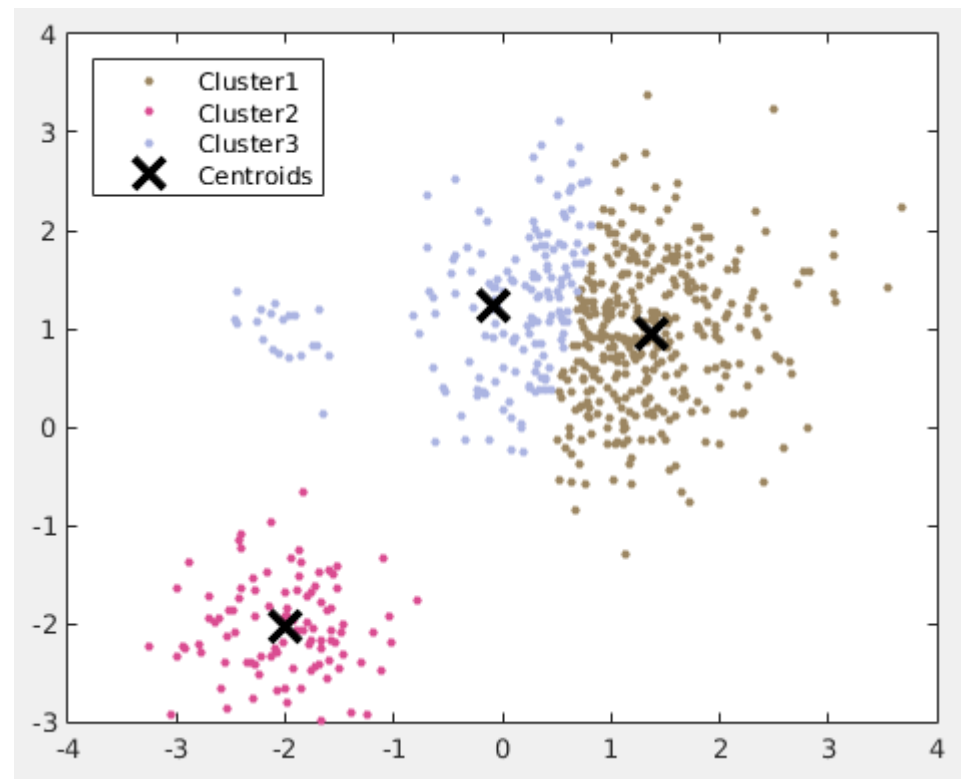
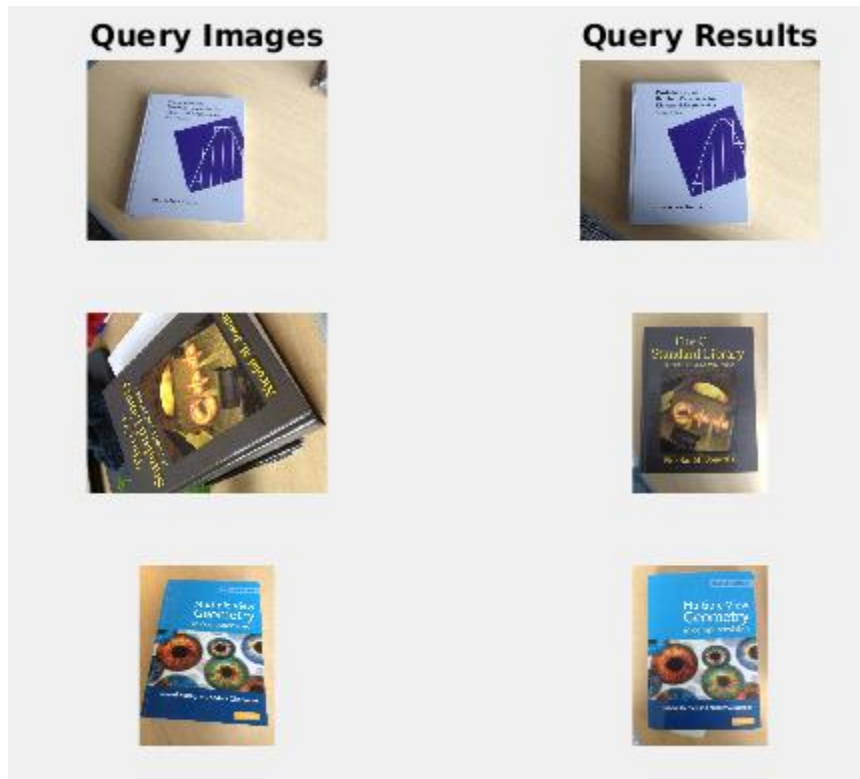
Recognition

Davide Scaramuzza

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

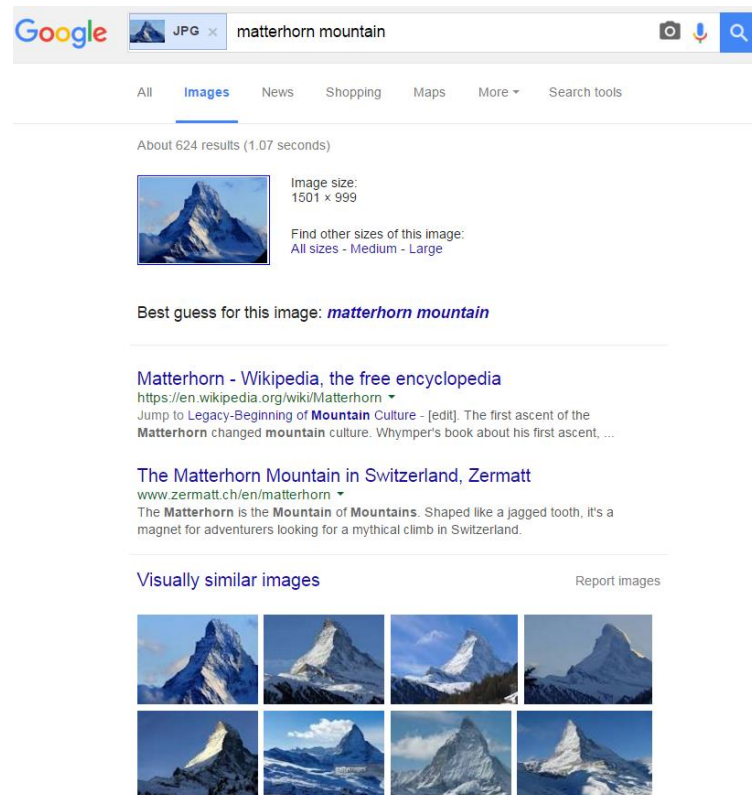
Lab exercise today replaced by Deep Learning Tutorial by Antonio Loquercio

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



Place Recognition

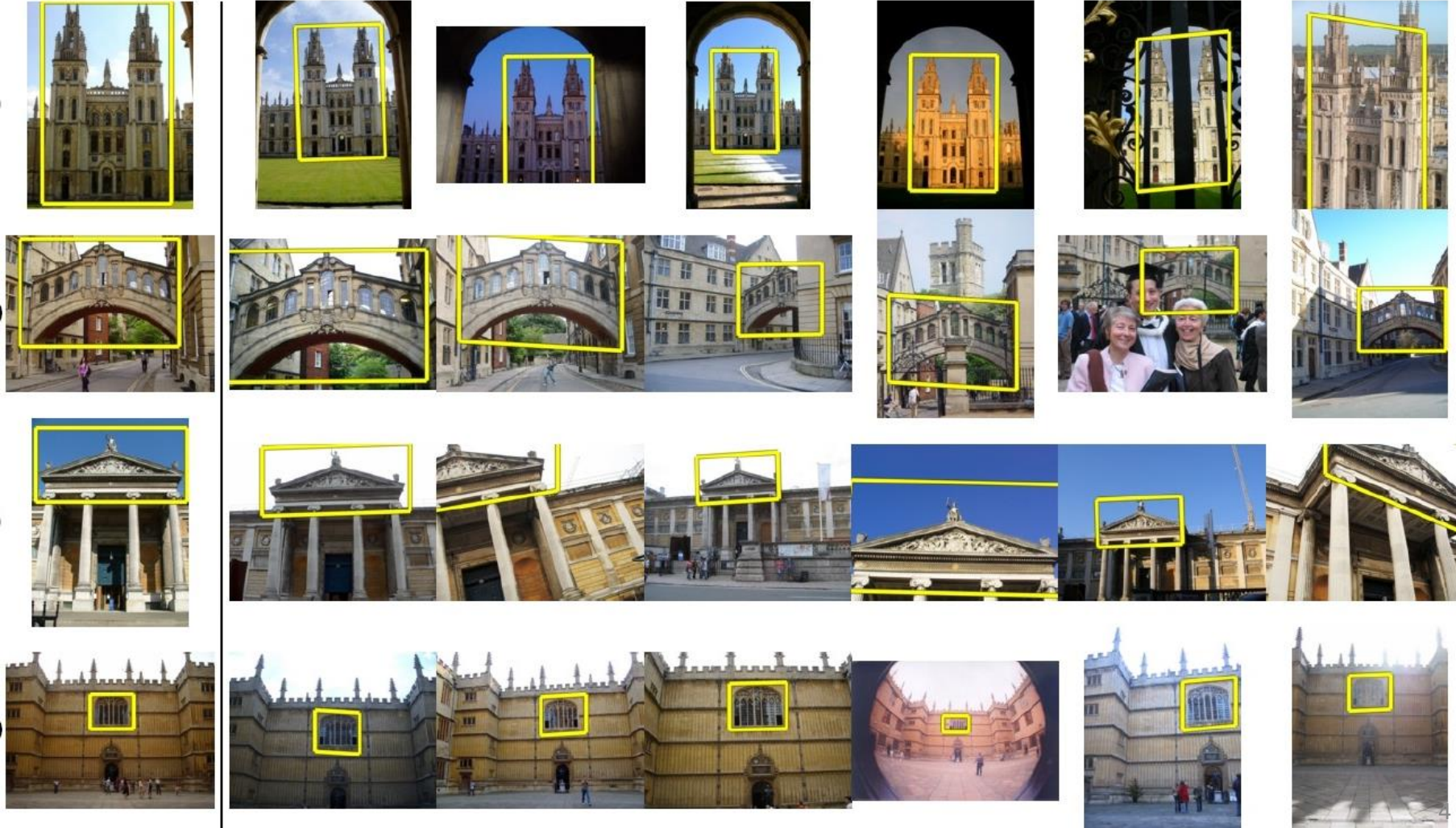
- **Robotics:** Has the robot been to this place before? Which images were taken around the same location?
- **Image retrieval:** Have I seen this image before? Which images in my database look similar to it? E.g., Google Reverse Image Search



Place Recognition/Image Retrieval

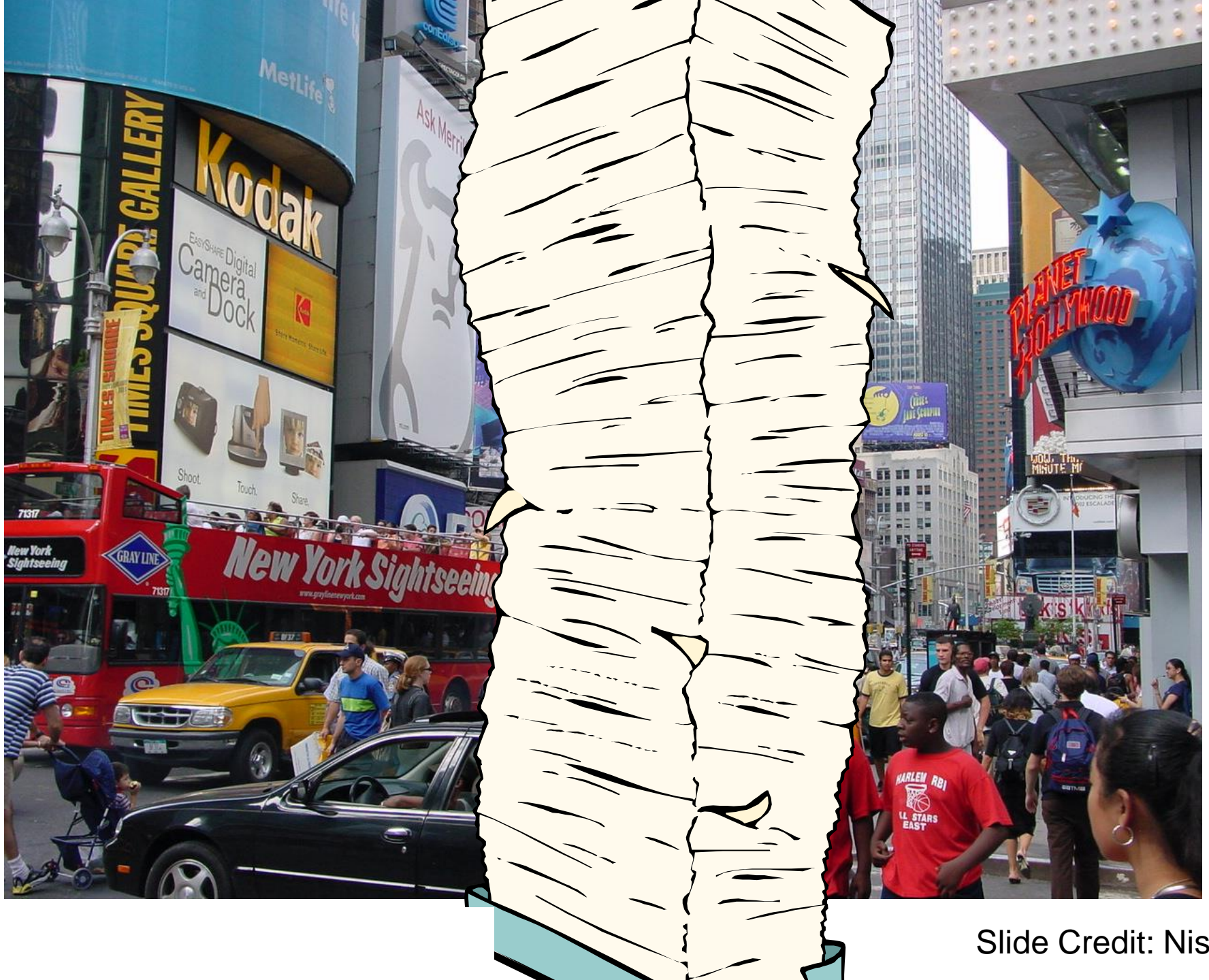
Query image

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



(C) L.W. Wildervanck

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 N images
- **Complexity:** NM^2 feature comparisons (assumes each image has M features)
 - **Example:**
 - assume 1,000 SIFT features per image $\rightarrow M = 1,000$
 - assume $N = 100,000,000$
 - $\rightarrow NM^2 = 100,000,000,000,000$ feature comparisons!
 - If we assume 0.1 ms per feature comparison \rightarrow 1 image query would take **317 years!**

Solution: Use an inverted file index!

Complexity reduces to $O(M)$

[“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)

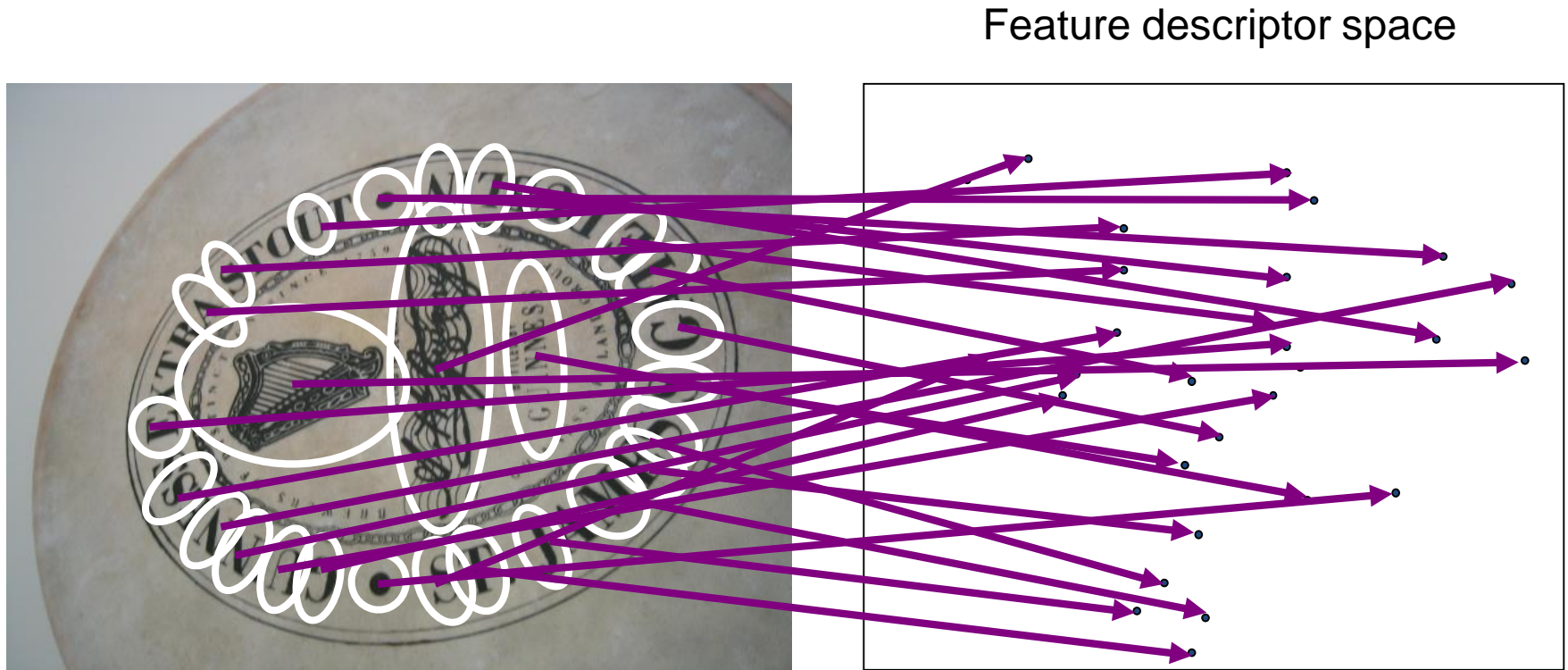
Index		
"Along I-75," From Detroit to Florida; <i>inside back cover</i>	Butterfly Center, McGuire; 134	Driving Lanes; 85
"Drive I-95," From Boston to Florida; <i>inside back cover</i>	CAA (see AAA)	Duval County; 163
1929 Spanish Trail Roadway; 101-102,104	CCC, The; 111,113,115,135,142	Eau Gallie; 175
511 Traffic Information; 83	Ca d'Zan; 147	Edison, Thomas; 152
A1A (Barrier Isl) - I-95 Access; 86	Caloosahatchee River; 152	Eglin AFB; 116-118
AAA (and CAA); 83	Name; 150	Eight Reale; 176
AAA National Office; 88	Canaveral Natnl Seashore; 173	Ellenton; 144-145
Abbreviations,	Cannon Creek Airpark; 130	Emanuel Point Wreck; 120
Colored 25 mile Maps; cover	Canopy Road; 106,169	Emergency Callboxes; 63
Exit Services; 196	Cape Canaveral; 174	Epiphytes; 142,148,157,159
Travelogue; 85	Castillo San Marcos; 169	Escambia Bay; 119
Africa; 177	Cave Diving; 131	Bridge (I-10); 119
Agricultural Inspection Stns; 126	Cayo Costa, Name; 150	County; 120
Ah-Tah-Thi-Ki Museum; 160	Celebration; 93	Estero; 153
Air Conditioning, First; 112	Charlotte County; 149	Everglade,90,95,139-140,154-160
Alabama; 124	Charlotte Harbor; 150	Draining of; 156,181
Alachua; 132	Chautauqua; 116	Wildlife MA; 160
County; 131	Chipley; 114	Wonder Gardens; 154
Alafia River; 143	Name; 115	Falling Waters SP; 115
Alapaha, Name; 126	Choctawhatchee, Name; 115	Fantasy of Flight; 95
Alfred B Maclay Gardens; 106	Circus Museum, Ringling; 147	Fayer Dykes SP; 171
Alligator Alley; 154-155	Citrus; 88,97,130,136,140,180	Fires, Forest; 166
Alligator Farm, St Augustine; 169	CityPlace, W Palm Beach; 180	Fires, Prescribed ; 148
Alligator Hole (definition); 157	City Maps,	Fisherman's Village; 151
Alligator, Buddy; 155	Fl Lauderdale Expwys; 194-195	Flagler County; 171
Alligators; 100,135,138,147,156	Jacksonville; 163	Flagler, Henry; 97,165,167,171
Anastasia Island; 170	Kissimmee Expwys; 192-193	Florida Aquarium; 186
Anhaica; 108-109,146	Miami Expressways; 194-195	Florida,
Apalachicola River; 112	Orlando Expressways; 192-193	12,000 years ago; 187
Appleton Mus of Art; 136	Pensacola; 26	Cavern SP; 114
Aquifer; 102	Tallahassee; 191	Map of all Expressways; 2-3
Arabian Nights; 94	Tampa-St. Petersburg; 63	Mus of Natural History; 134
Art Museum, Ringling; 147	St. Augustine; 191	National Cemetery ; 141
Aruba Beach Cafe; 183	Civil War; 100,108,127,138,141	Part of Africa; 177
Aucilla River Project; 106	Clearwater Marine Aquarium; 187	Platform; 187
Babcock-Web WMA; 151	Collier County; 154	Sheriff's Boys Camp; 126
Bahia Mar Marina; 184	Collier, Barron; 152	Sports Hall of Fame; 130
Baker County; 99	Colonial Spanish Quarters; 168	Sun 'n Fun Museum; 97
Barefoot Mailmen; 182	Columbia County; 101,128	Supreme Court; 107
Barge Canal; 137	Coquina Building Material; 165	Florida's Turnpike (FTP), 178,189
Bee Line Expy; 80	Corkscrew Swamp, Name; 154	25 mile Strip Maps; 66
Belz Outlet Mall; 89	Cowboys; 95	Administration; 189
Bernard Castro; 136	Crab Trap II; 144	Coin System; 190
Big "I"; 165	Cracker, Florida; 88,95,132	Exit Services; 189
Big Cypress; 155,158	Crostown Expy; 11,35,98,143	HEFT; 76,161,190
Big Foot Monster; 105	Cuban Bread; 184	History; 189
Billie Swamp Safari; 160	Dade Battlefield; 140	Names; 189
Blackwater River SP; 117	Dade, Maj. Francis; 139-140,161	Service Plazas; 190
Blue Angels	Dania Beach Hurricane; 184	Spur SR91; 76
A4-C Skyhawk; 117	Daniel Boone, Florida Walk; 117	Ticket System; 190
Atrium; 121	Daytona Beach; 172-173	Toll Plazas; 190
Blue Springs SP; 87	De Land; 87	Ford, Henry; 152
Blue Star Memorial Highway; 125	De Soto, Hernando,	Fort Barrancas; 122
Boca Ciega; 189	Anhaica; 108-109,146	Buried Alive; 123
Boca Grande; 150	County; 149	Fort Caroline; 164
	Explorer; 146	Fort Clinch SP; 161
	Landing; 146	Fort De Soto & Egmont Key; 188
	Napitaca; 103	Fort Lauderdale; 161,182-184

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 **same 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

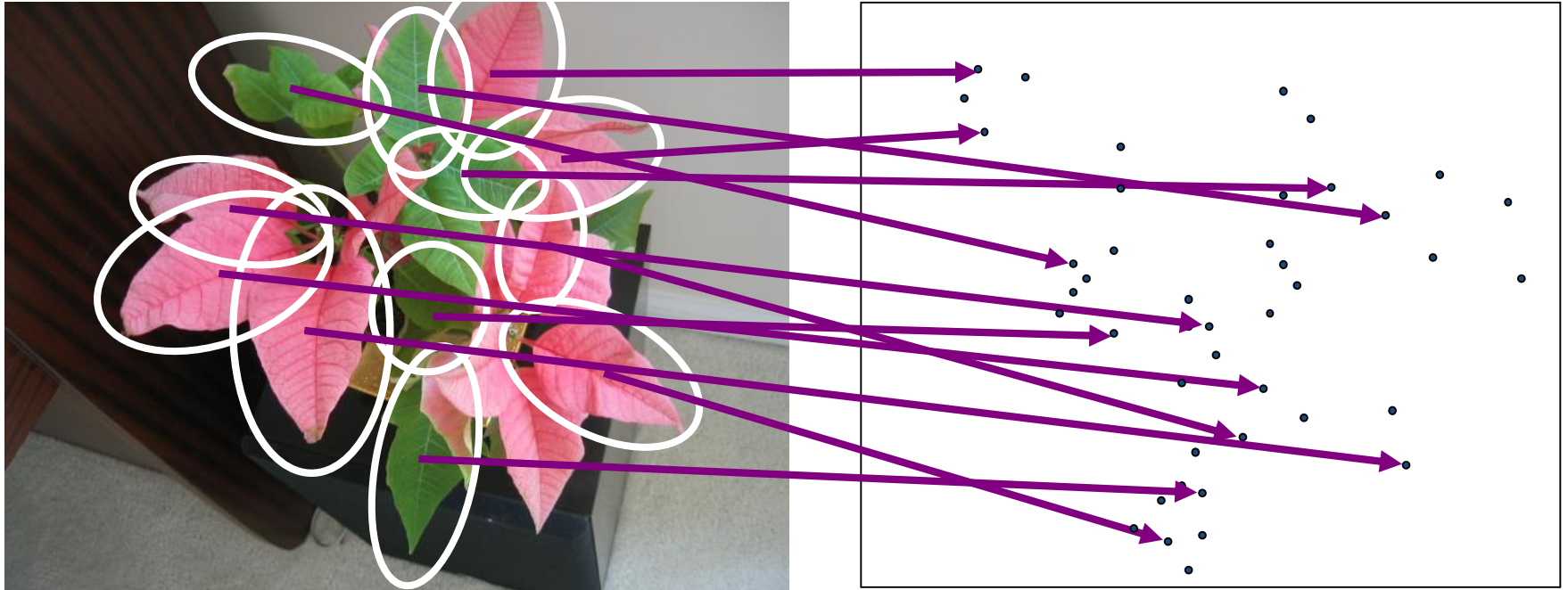
Let's see an example...

Extracting and mapping descriptors into the descriptor space



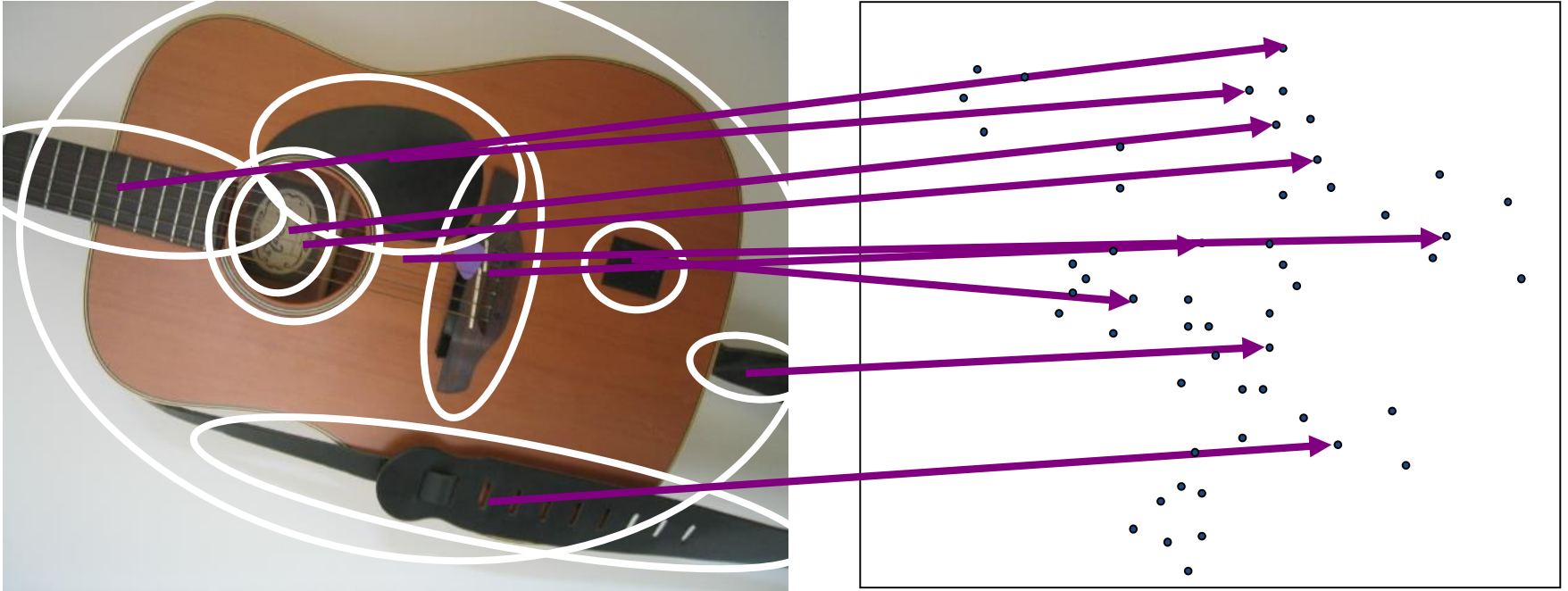
For convenience we assume that the descriptor space has 2 dimensions.
In case of SIFT, it would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



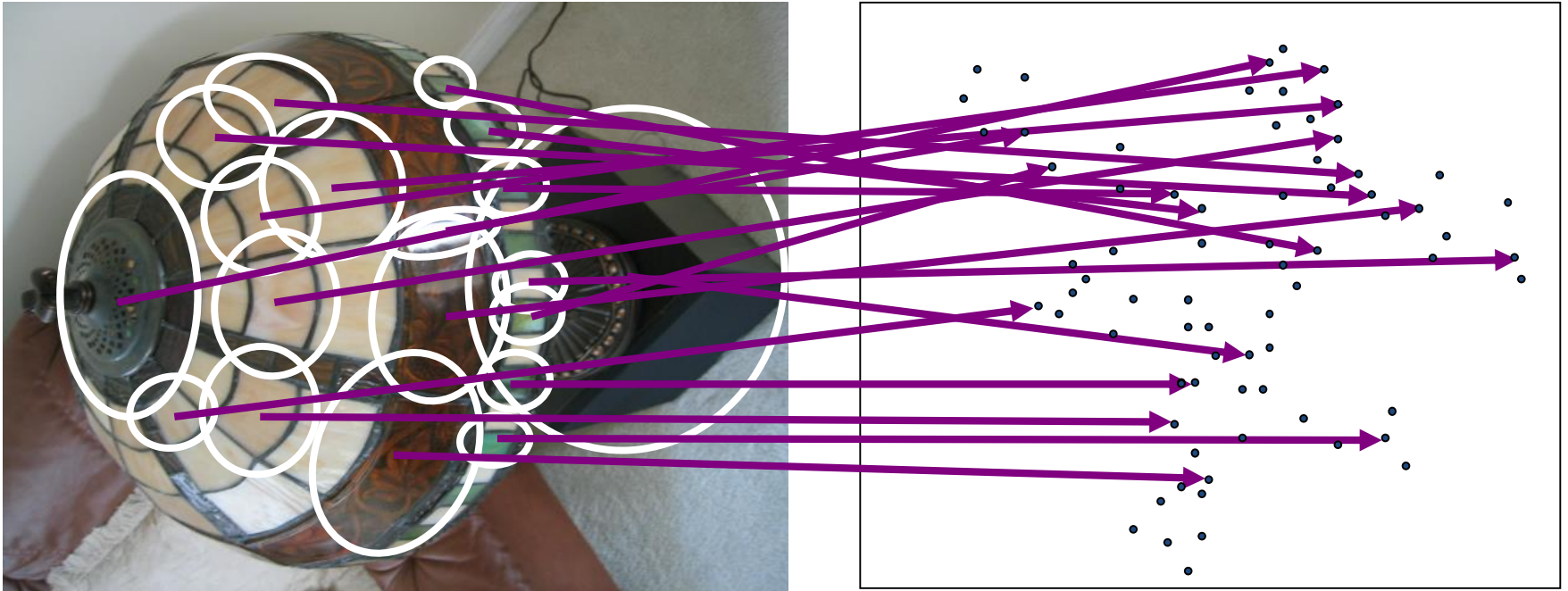
For convenience we assume that the descriptor space has 2 dimensions.
In case of SIFT, it would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



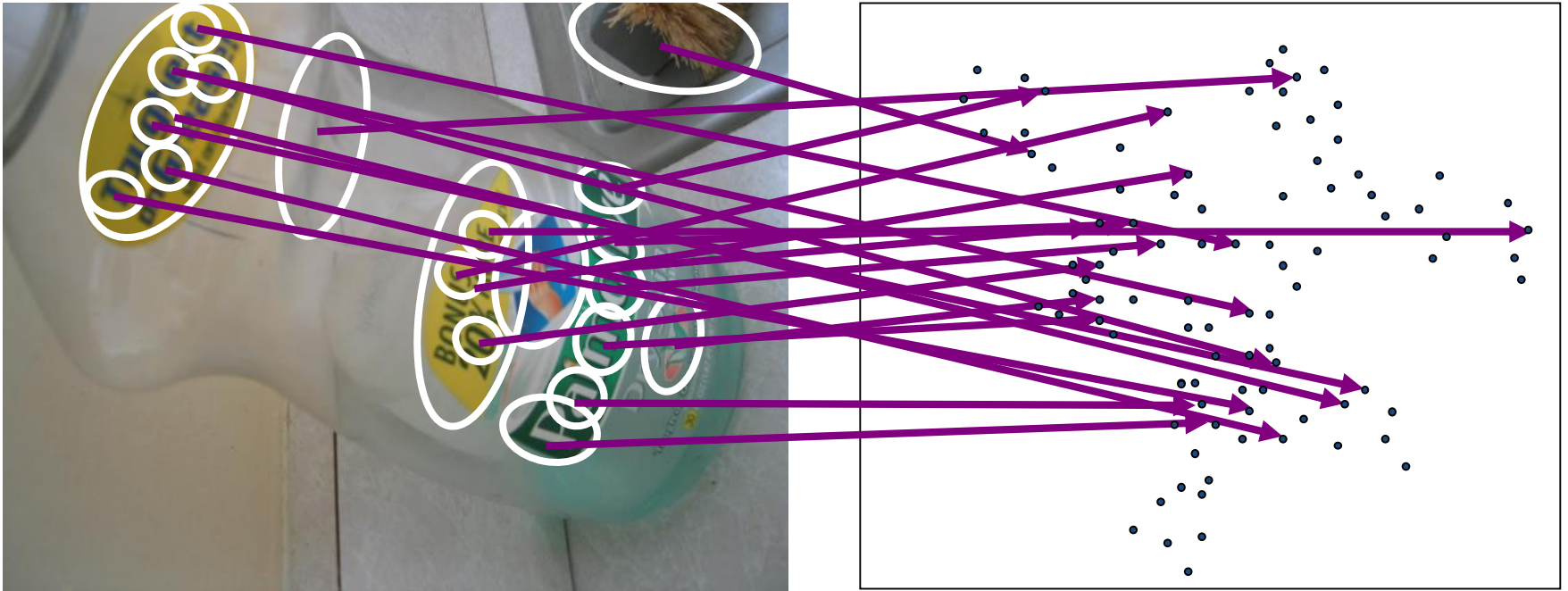
For convenience we assume that the descriptor space has 2 dimensions.
In case of SIFT, it would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space



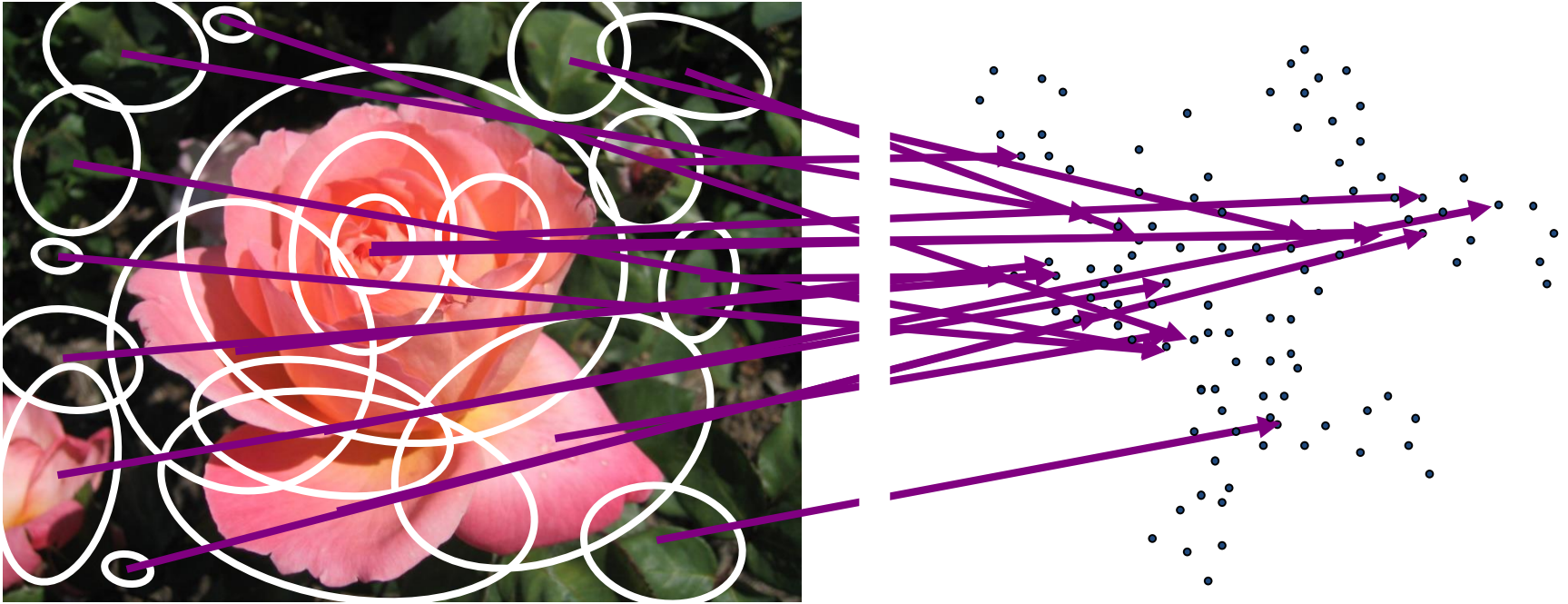
For convenience we assume that the descriptor space has 2 dimensions.
In case of SIFT, it would have 128 dimensions!

Extracting and mapping descriptors into the descriptor space

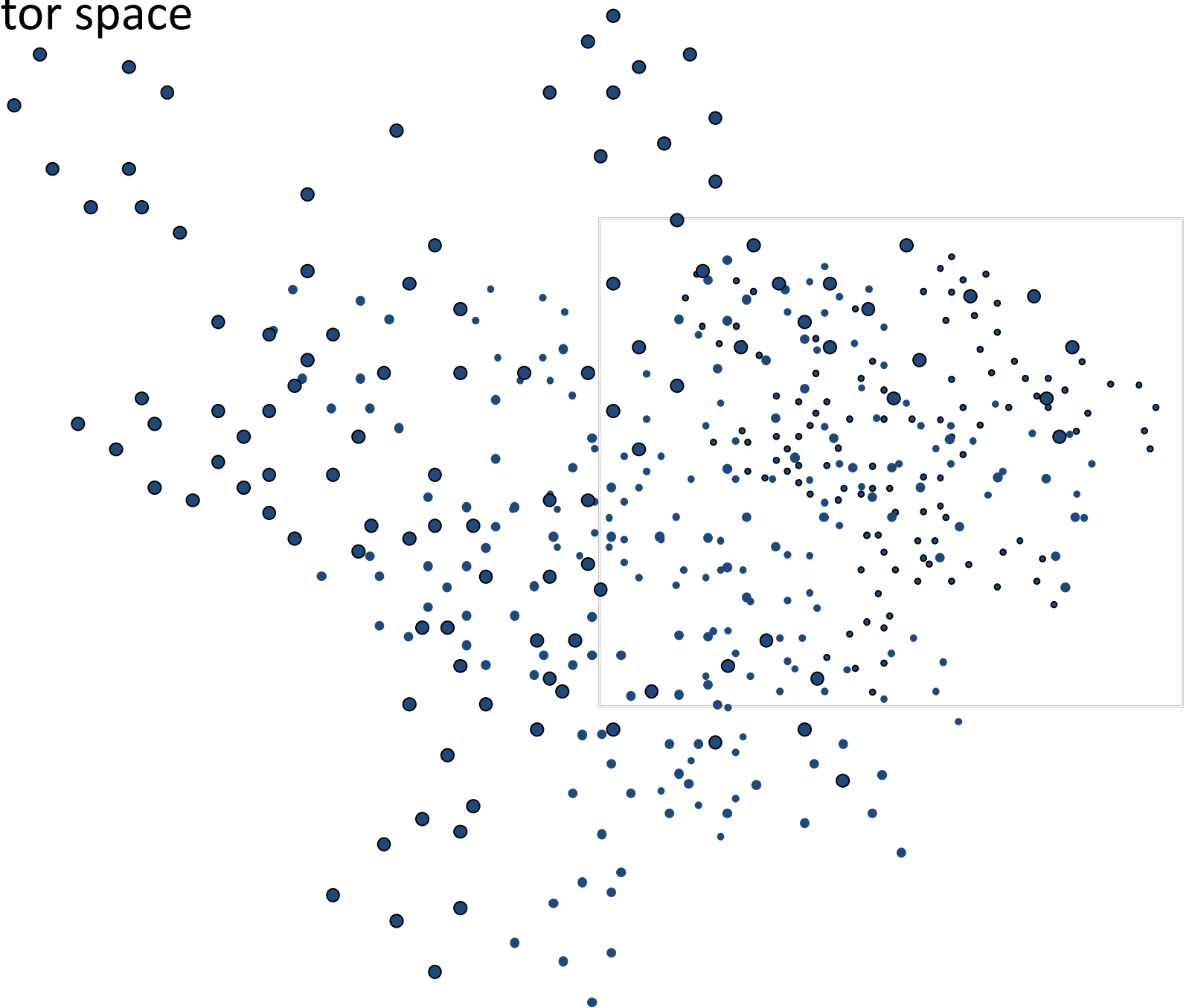


For convenience we assume that the descriptor space has 2 dimensions.
In case of SIFT, it would have 128 dimensions!

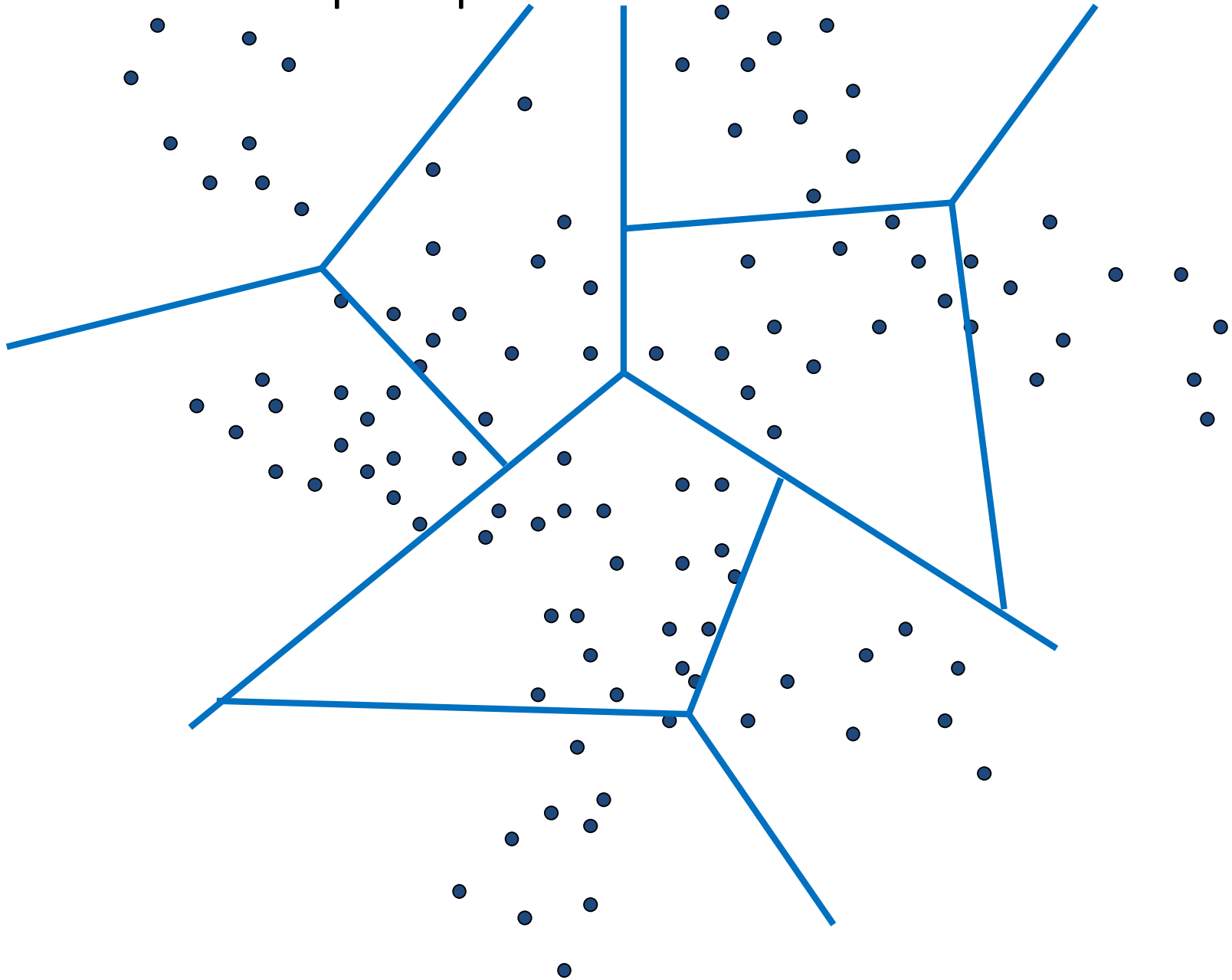
Extracting and mapping descriptors into the descriptor space



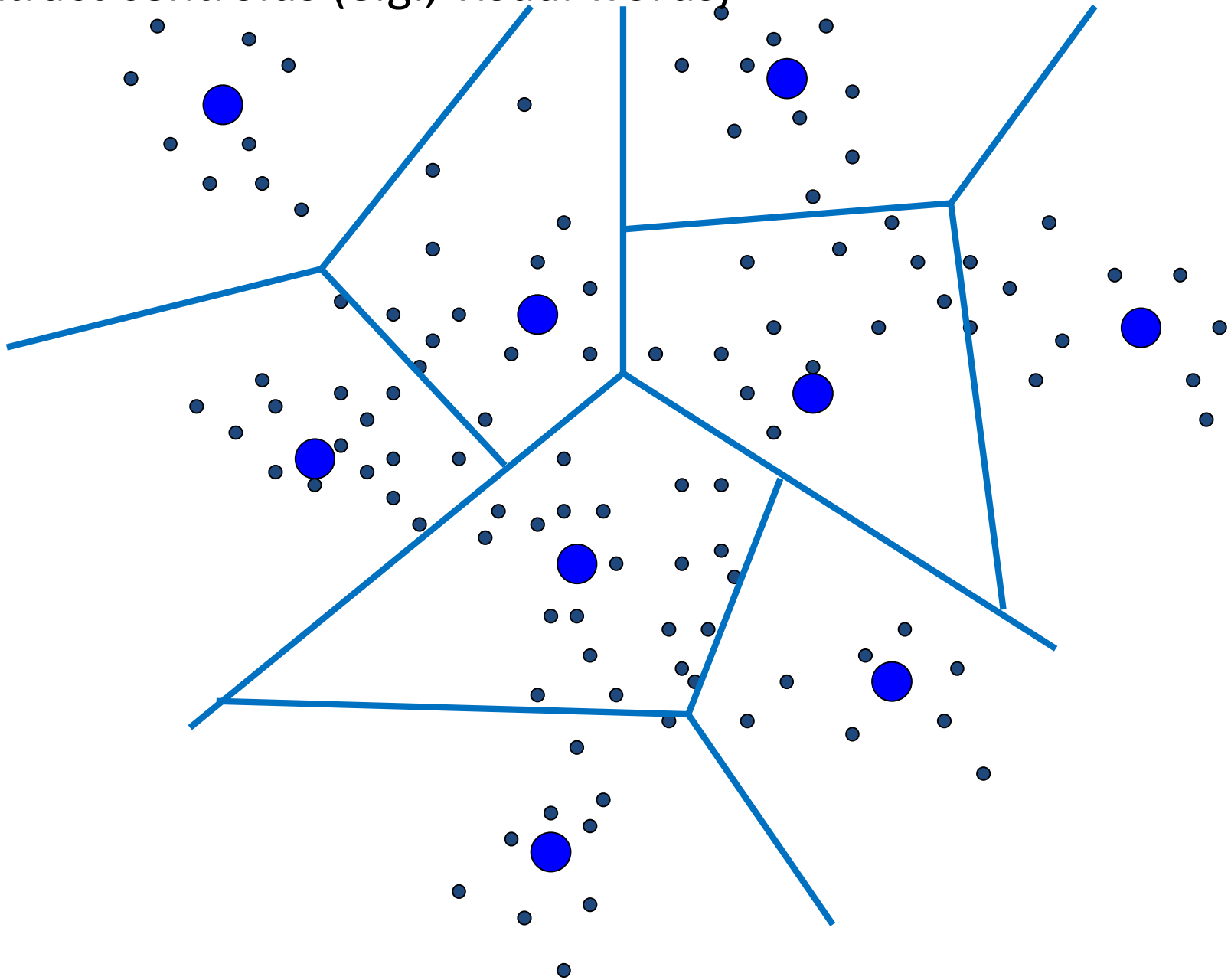
Descriptor space



Cluster the descriptor space into K clusters

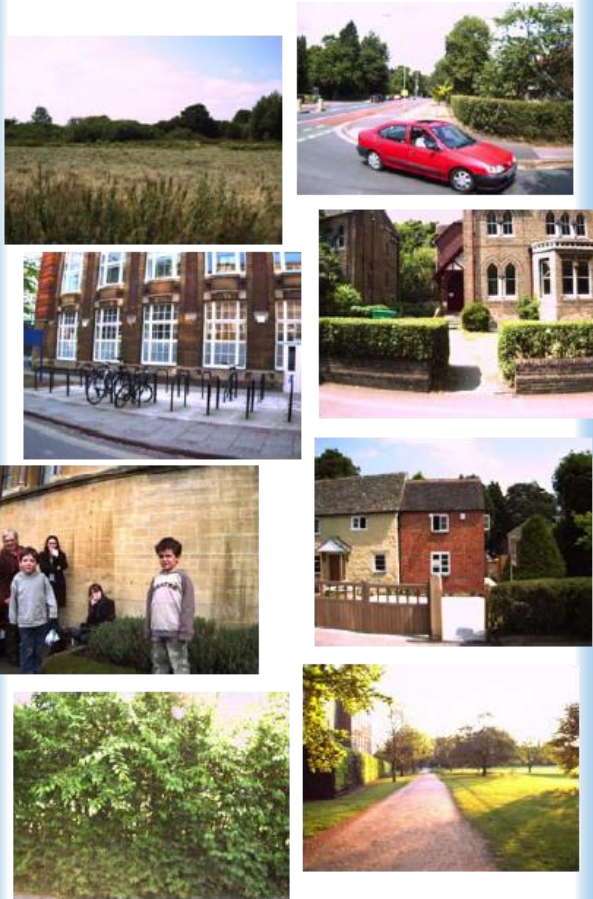


Extract centroids (e.g., visual words)

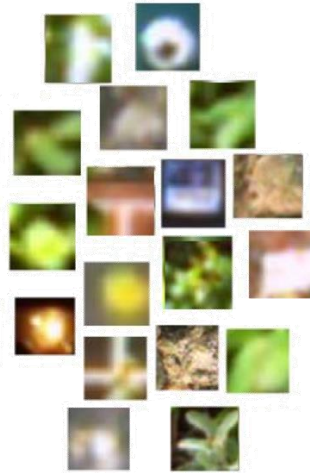


Summary

Image Collection



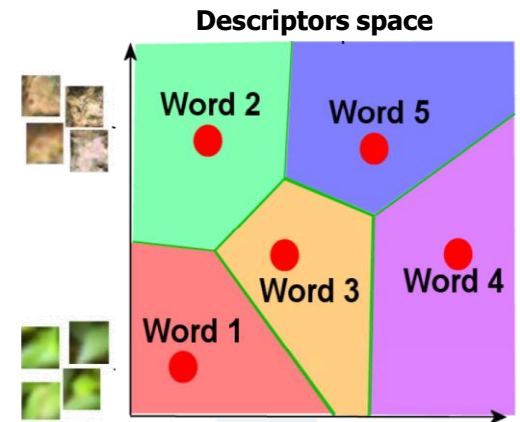
Extract Features



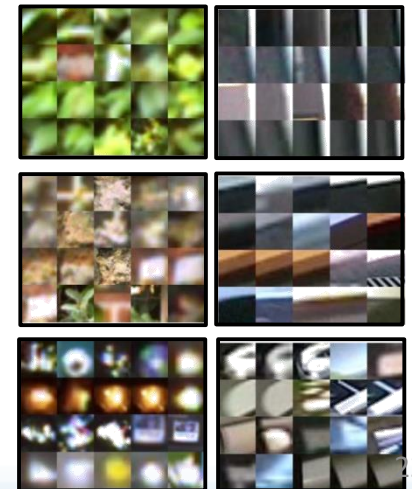
What is a visual word?

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



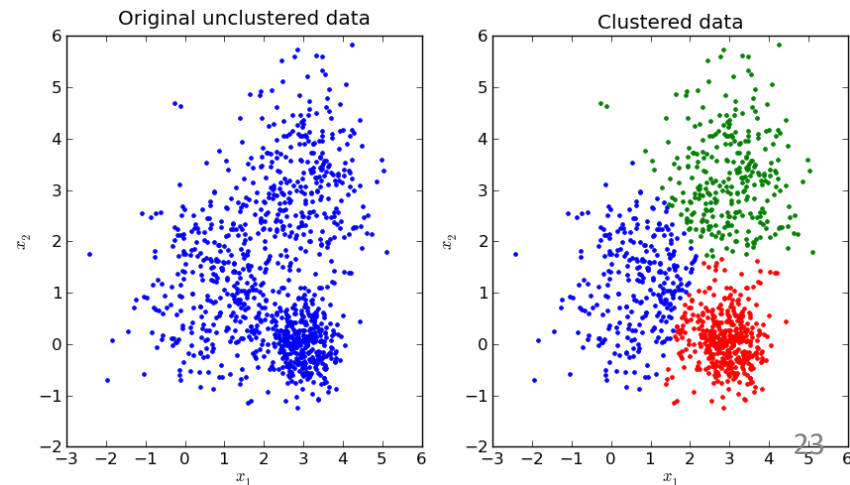
How do we cluster the descriptor space?

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

$$D(X, M) = \sum_{i=1}^k \sum_{\mathbf{x} \in \mathcal{S}_i} (\mathbf{x} - \mathbf{m}_i)^2$$

Algorithm:

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



K-means demo

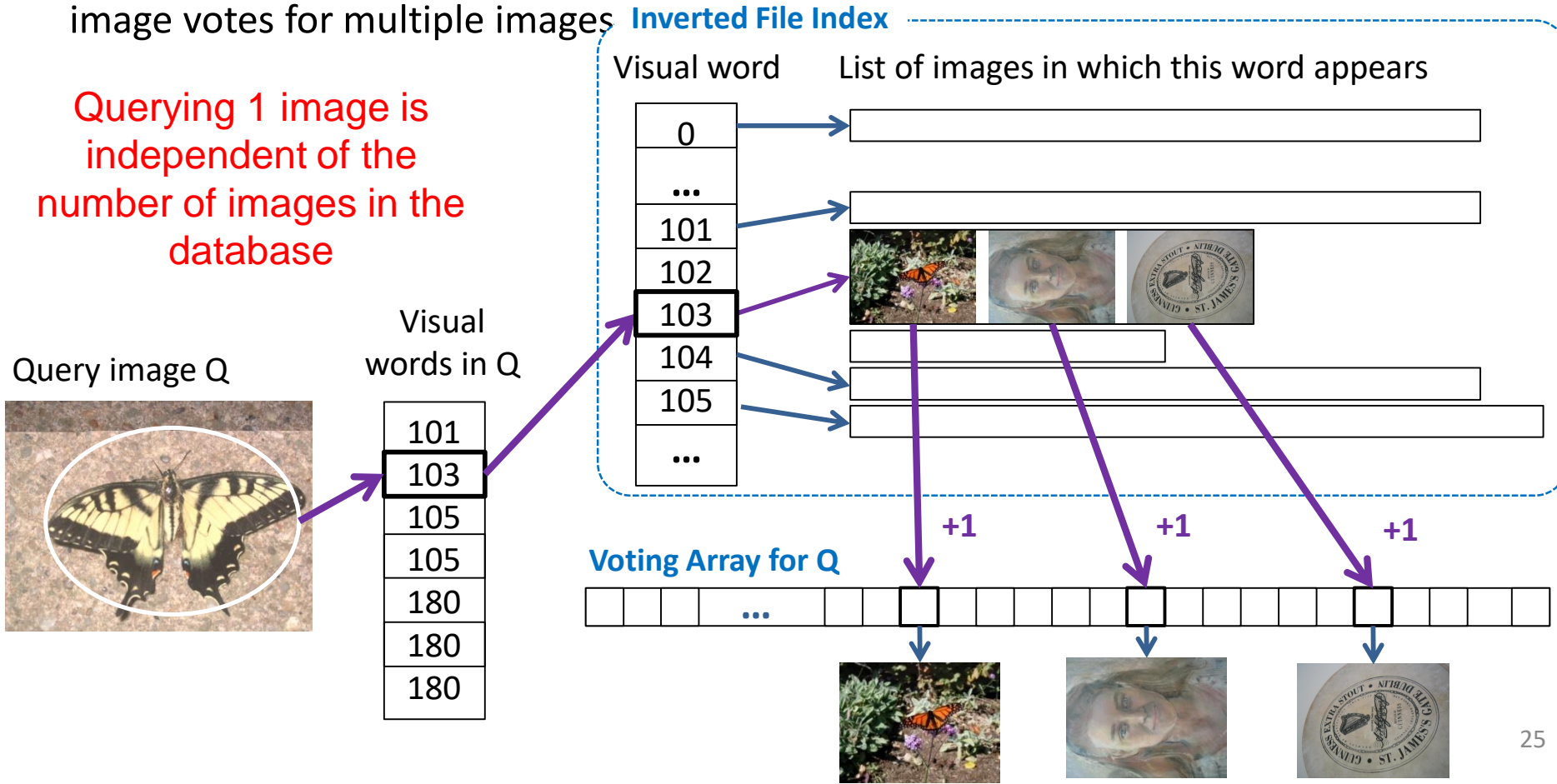


Source: <http://shabal.in/visuals/kmeans/1.html>

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 all 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

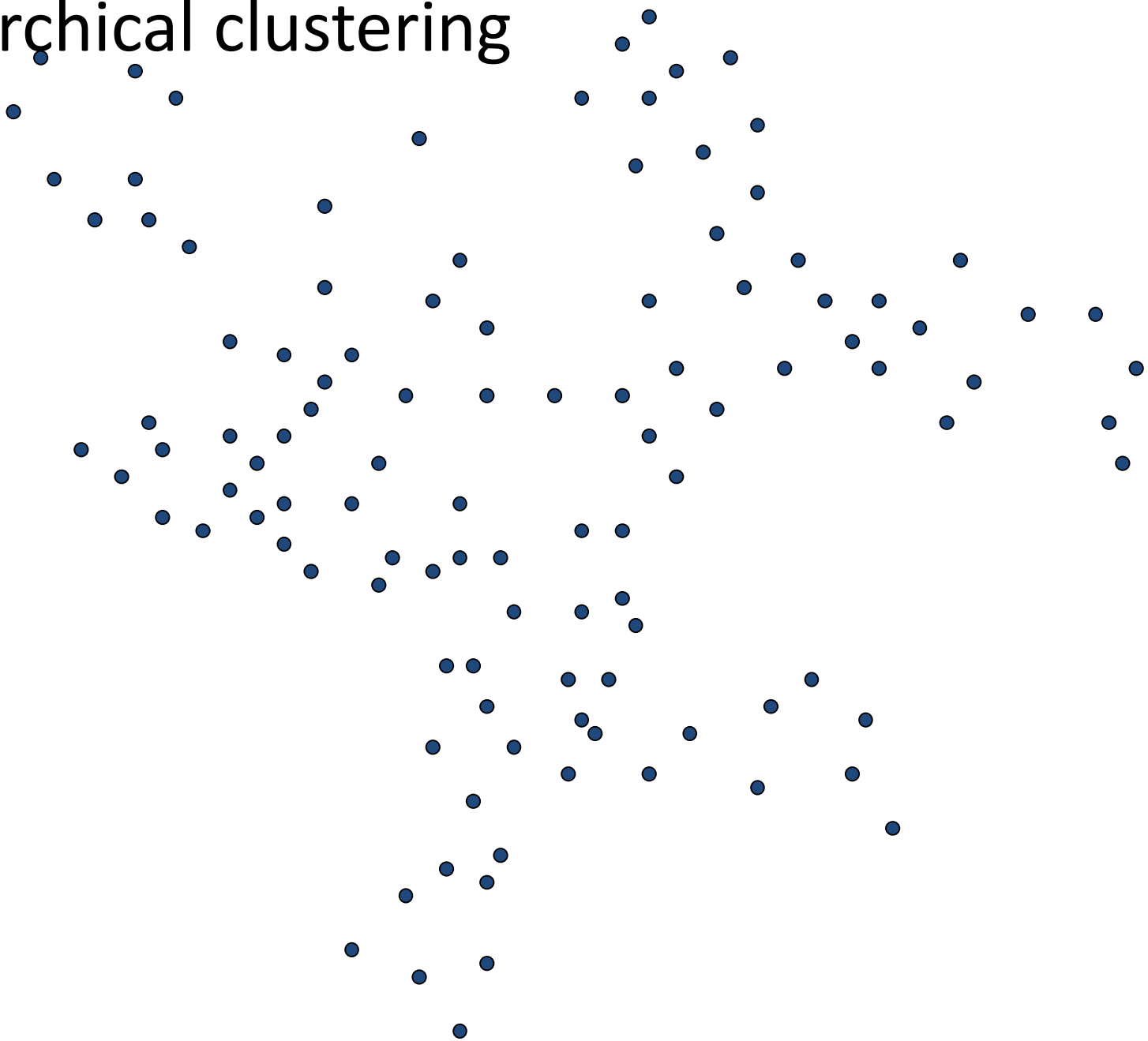
Querying 1 image is independent of the number of images in the database



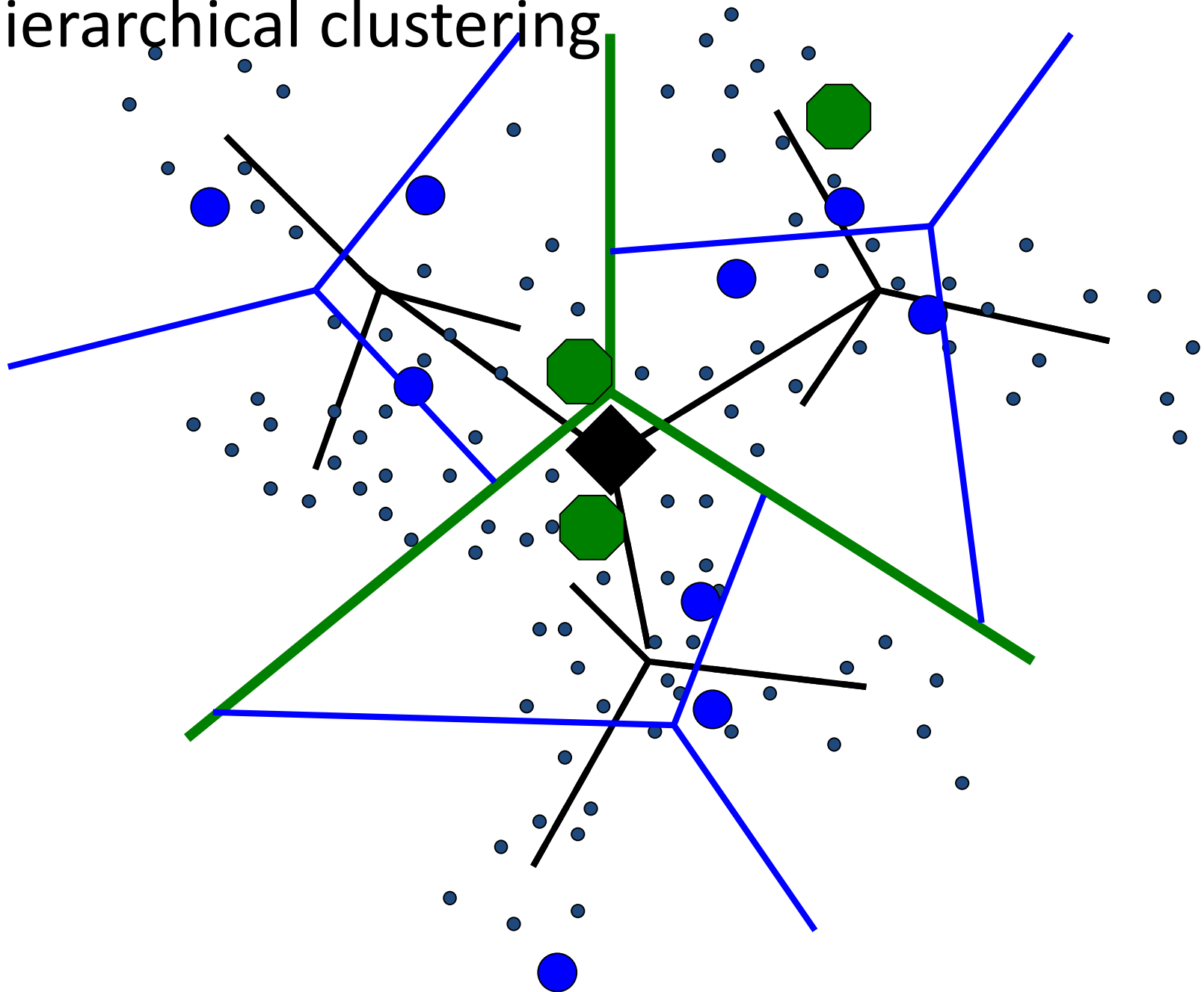
Drawback

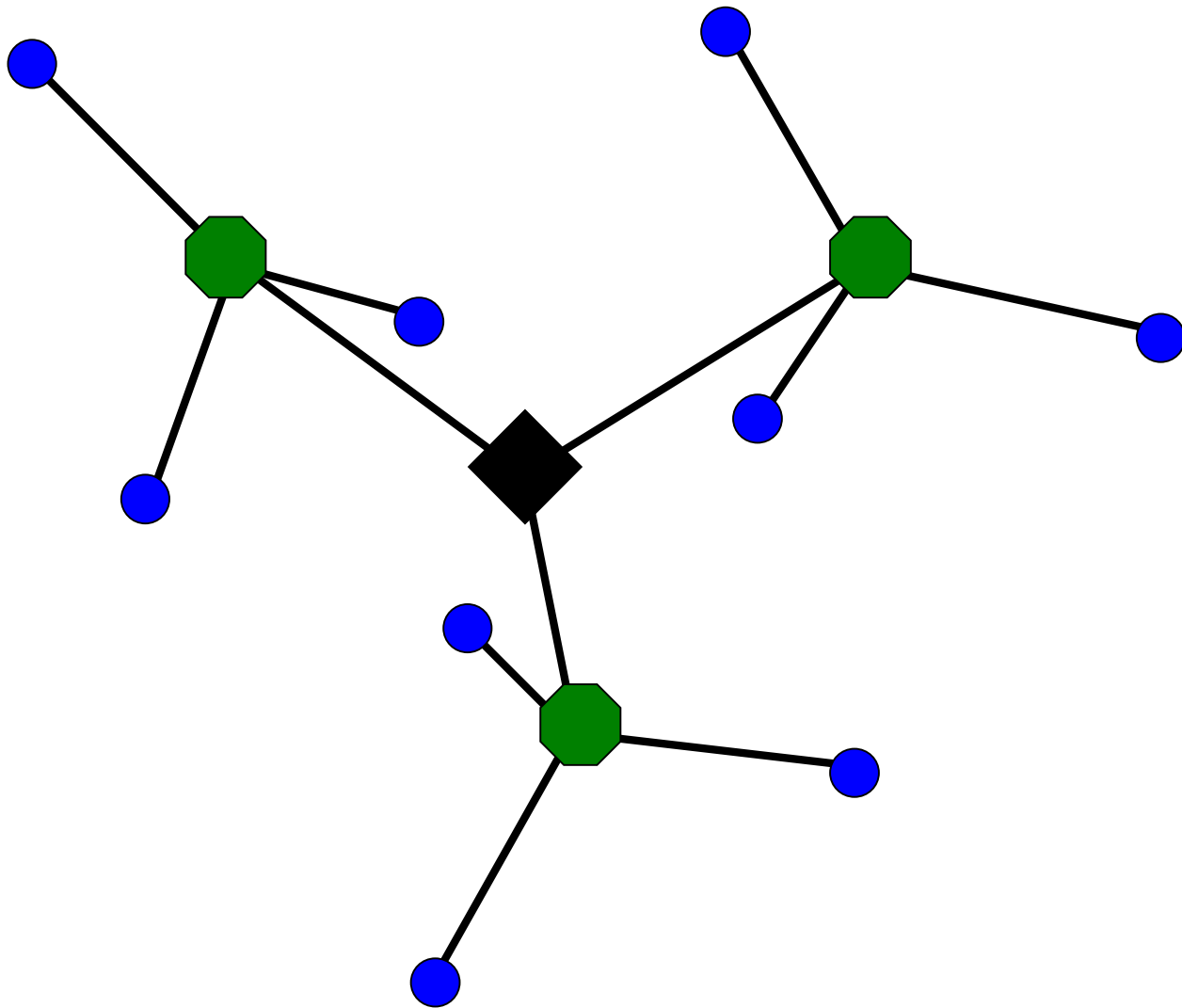
- Every feature in the query image still needs to be compared against all features in the vocabulary:
 - Example:
 - Assume our query image has 1,000 SIFT features $\rightarrow M = 1,000$
 - assume 1,000,000 visual words
 - \rightarrow Number of feature comparisons = 1,000,000,000
 - If we assume 0.1 ms per feature comparison \rightarrow 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]

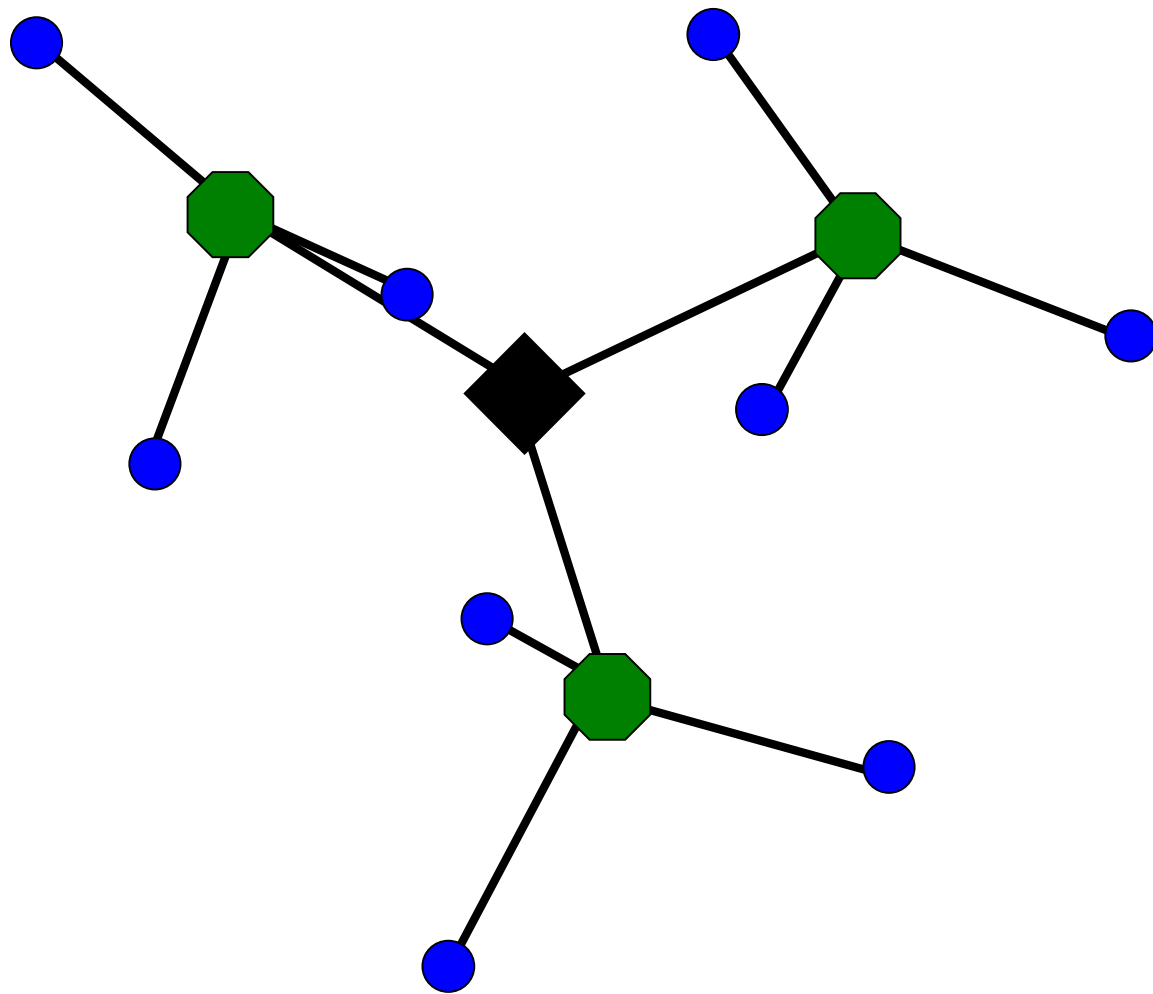
Hierarchical clustering

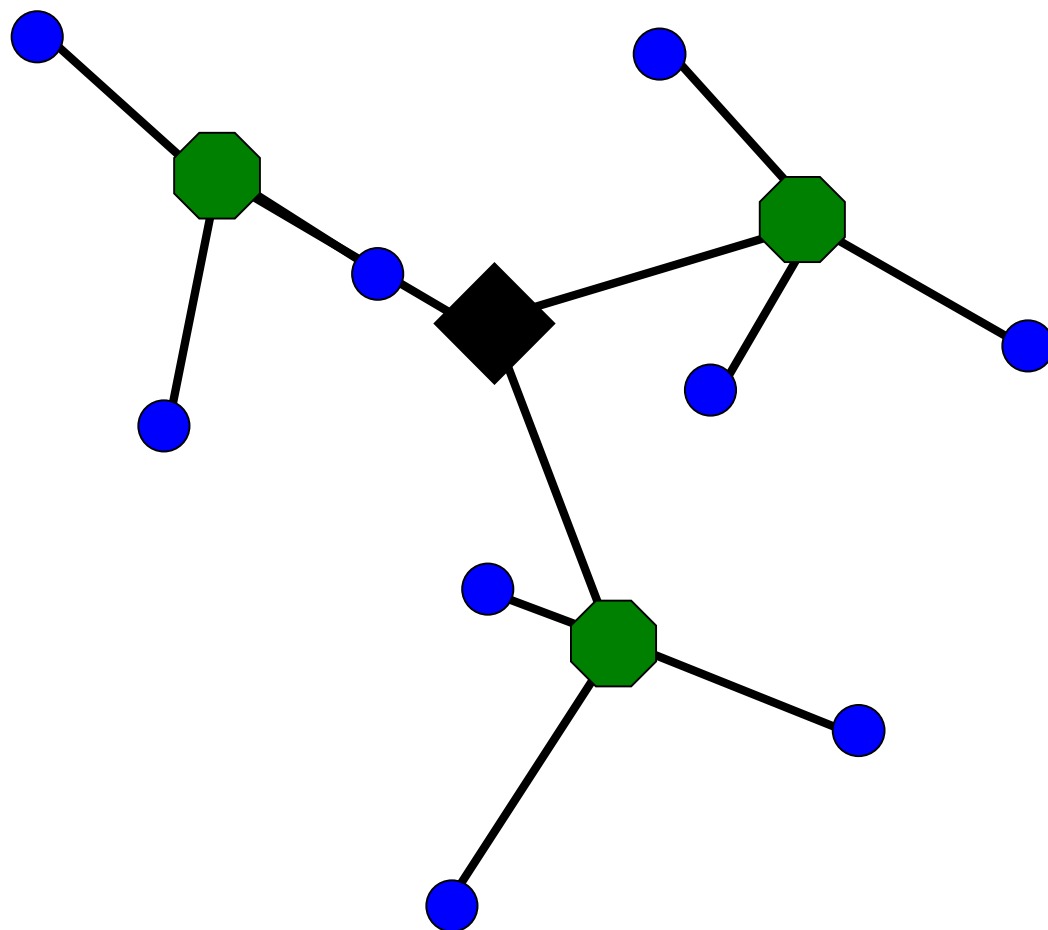


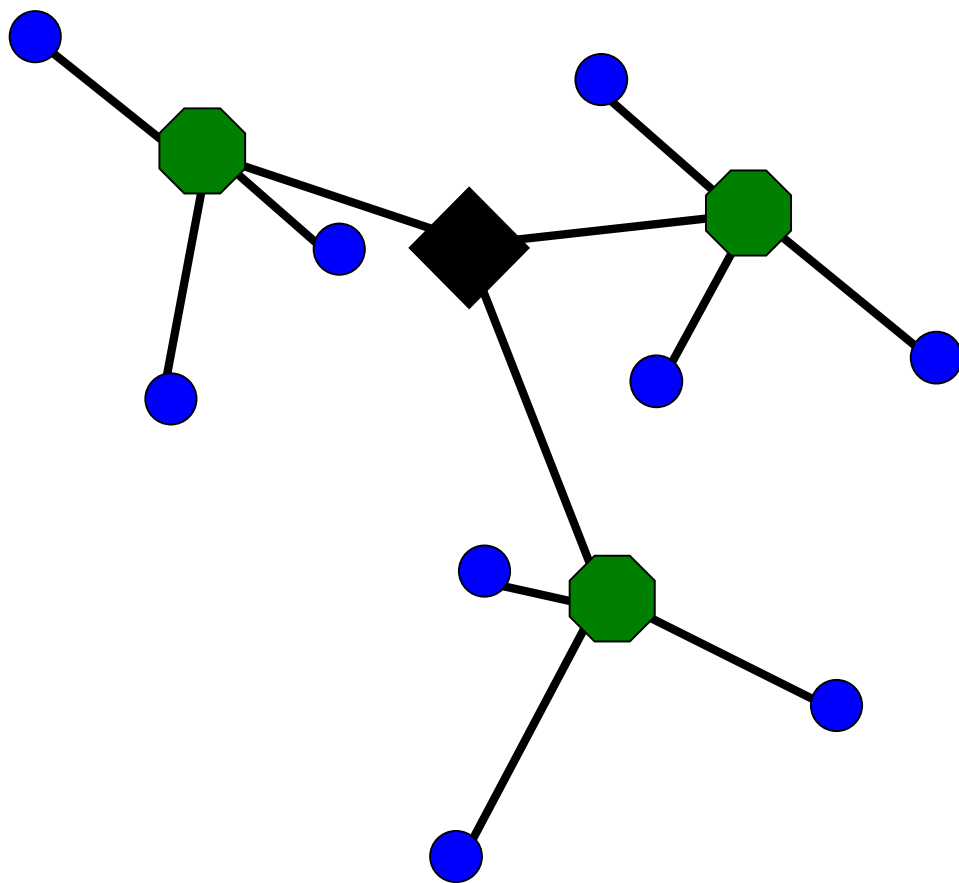
Hierarchical clustering

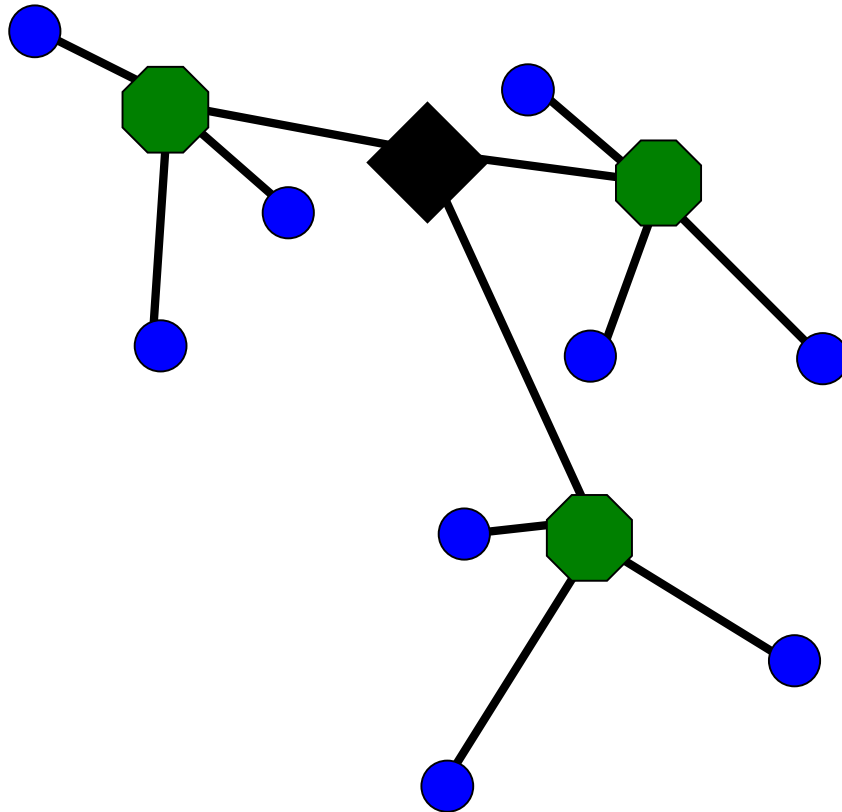


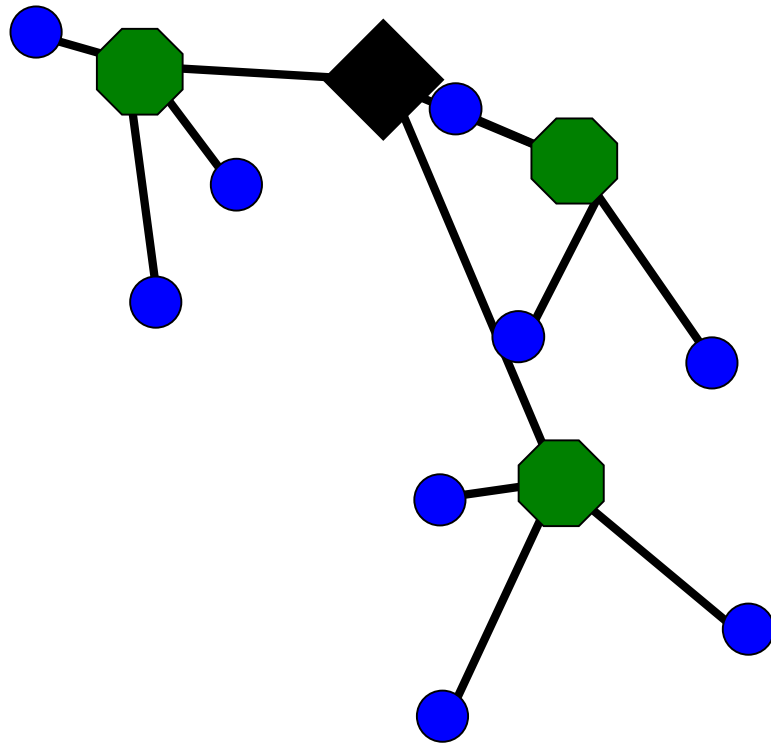


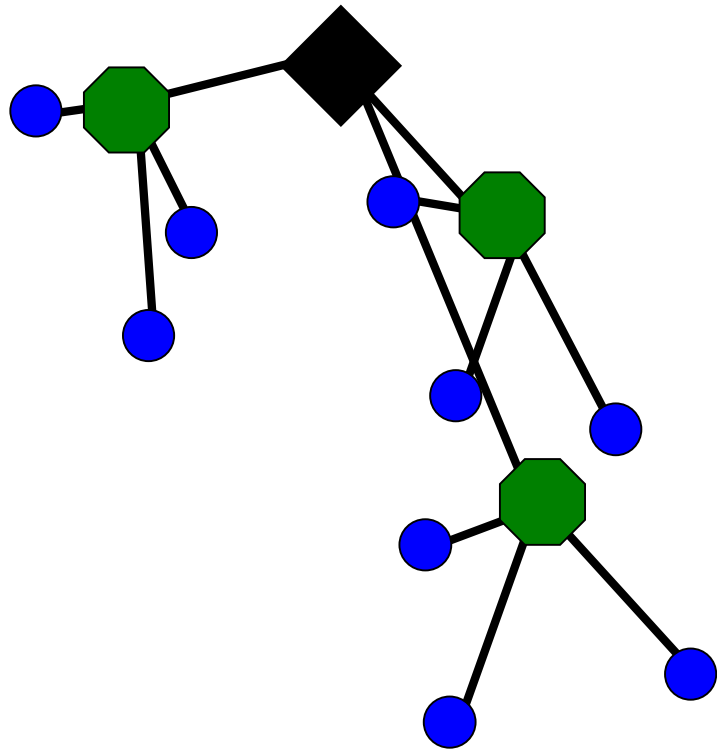


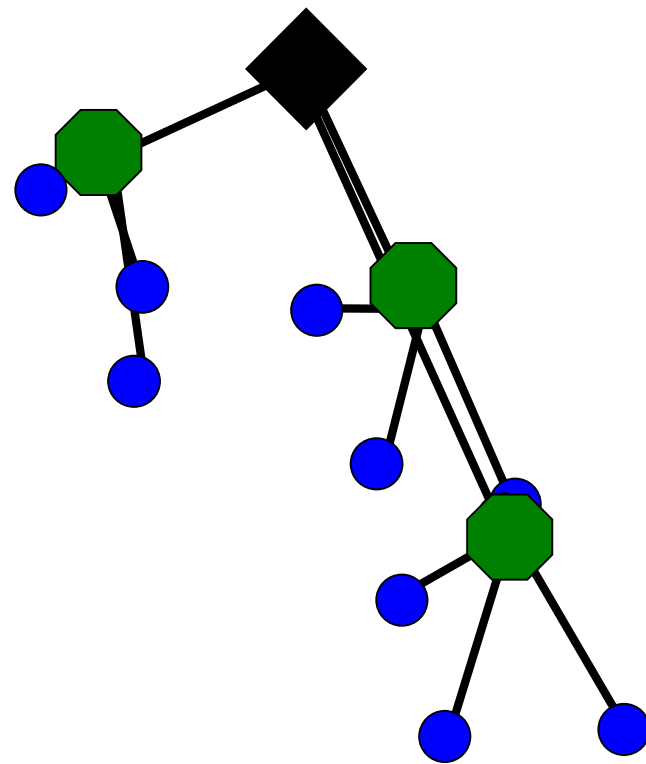


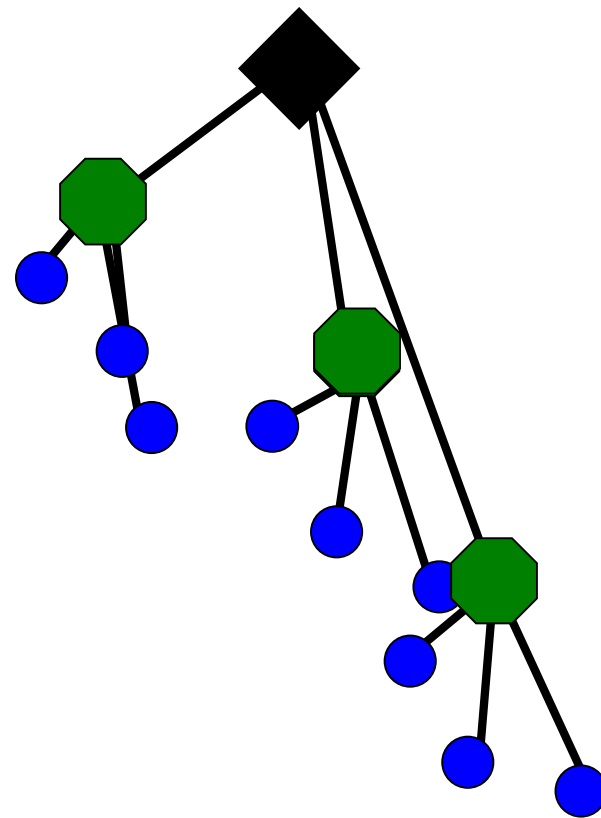


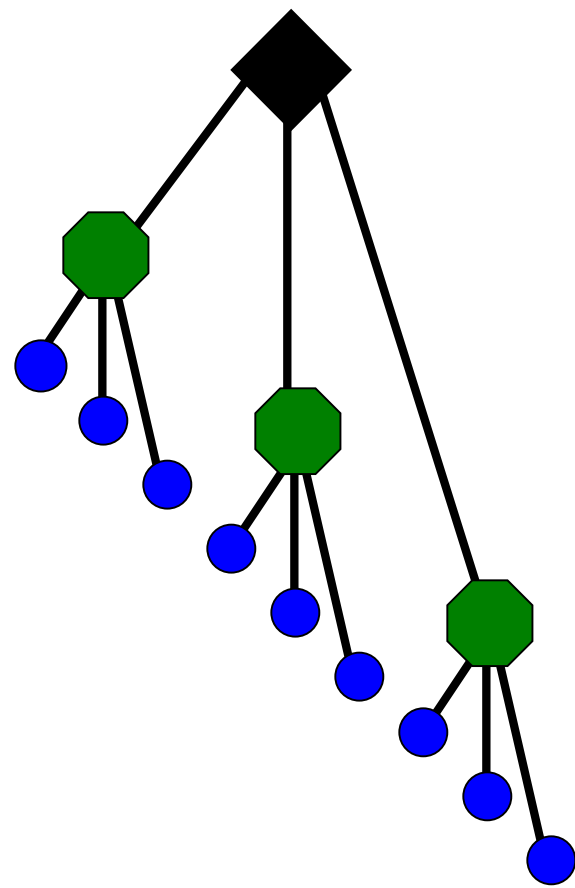




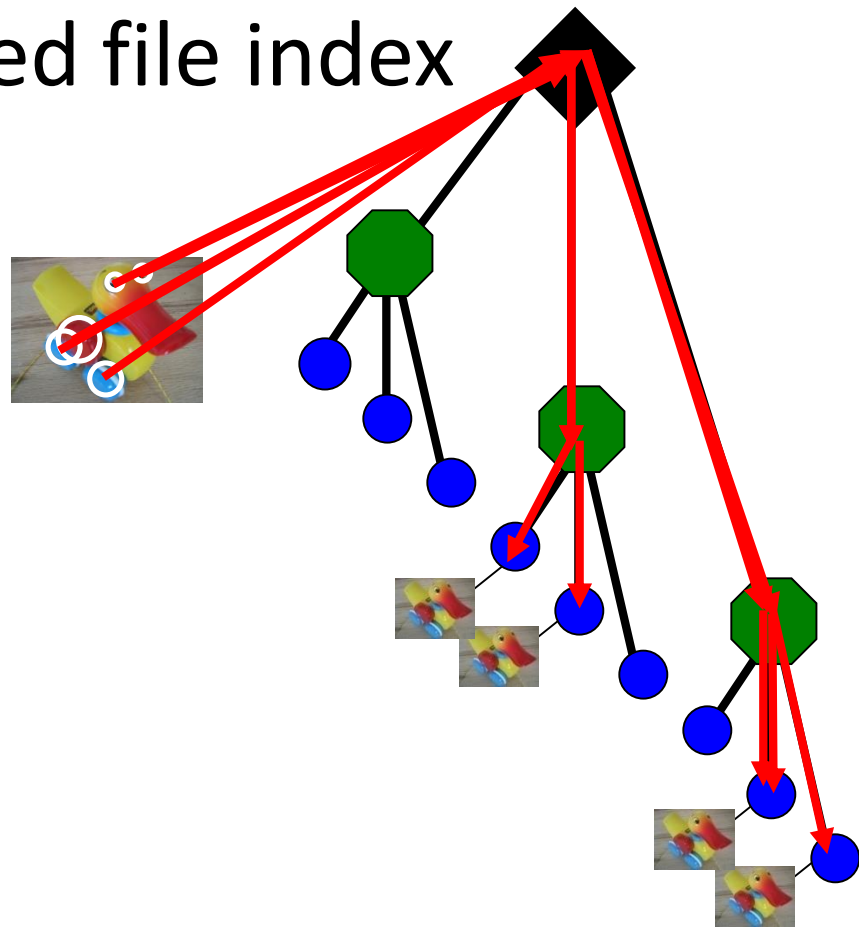




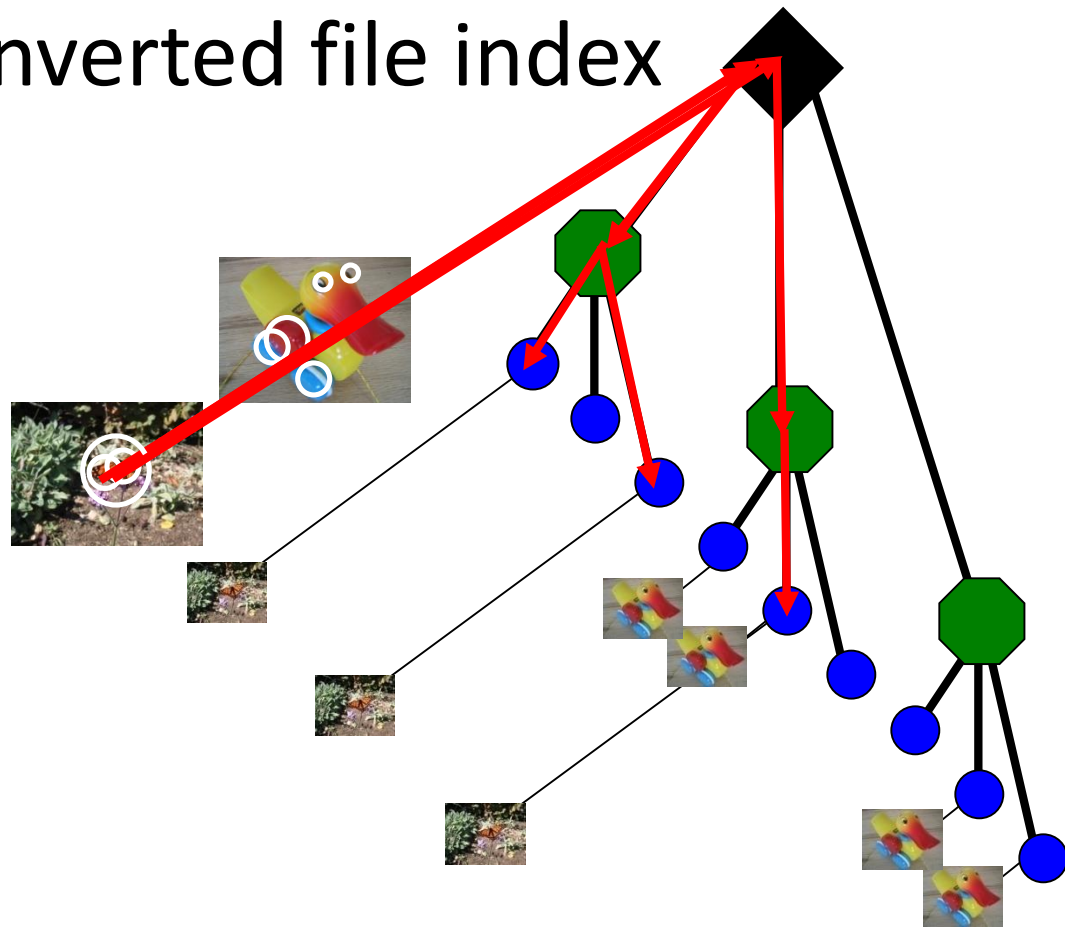




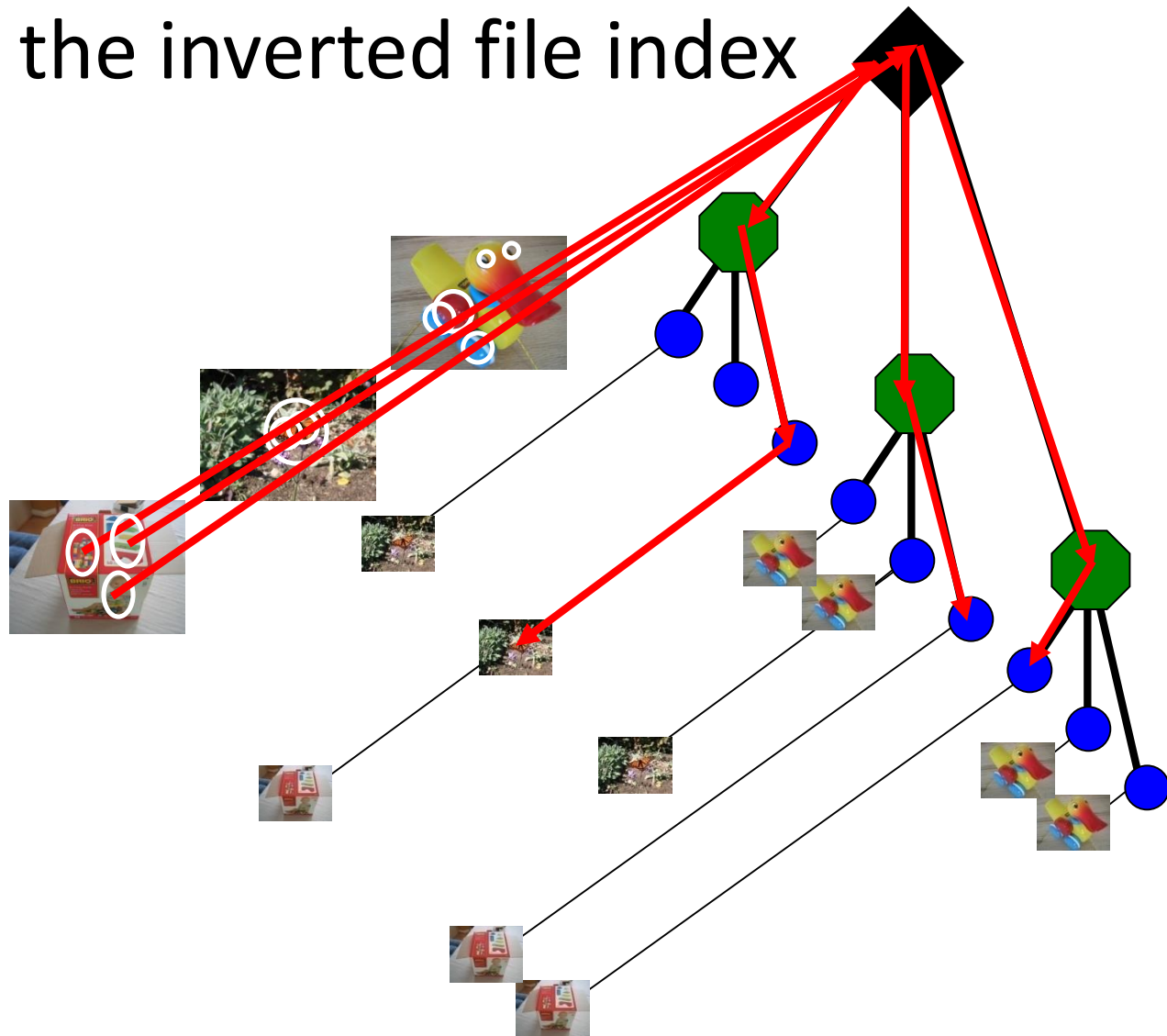
Building the inverted file index



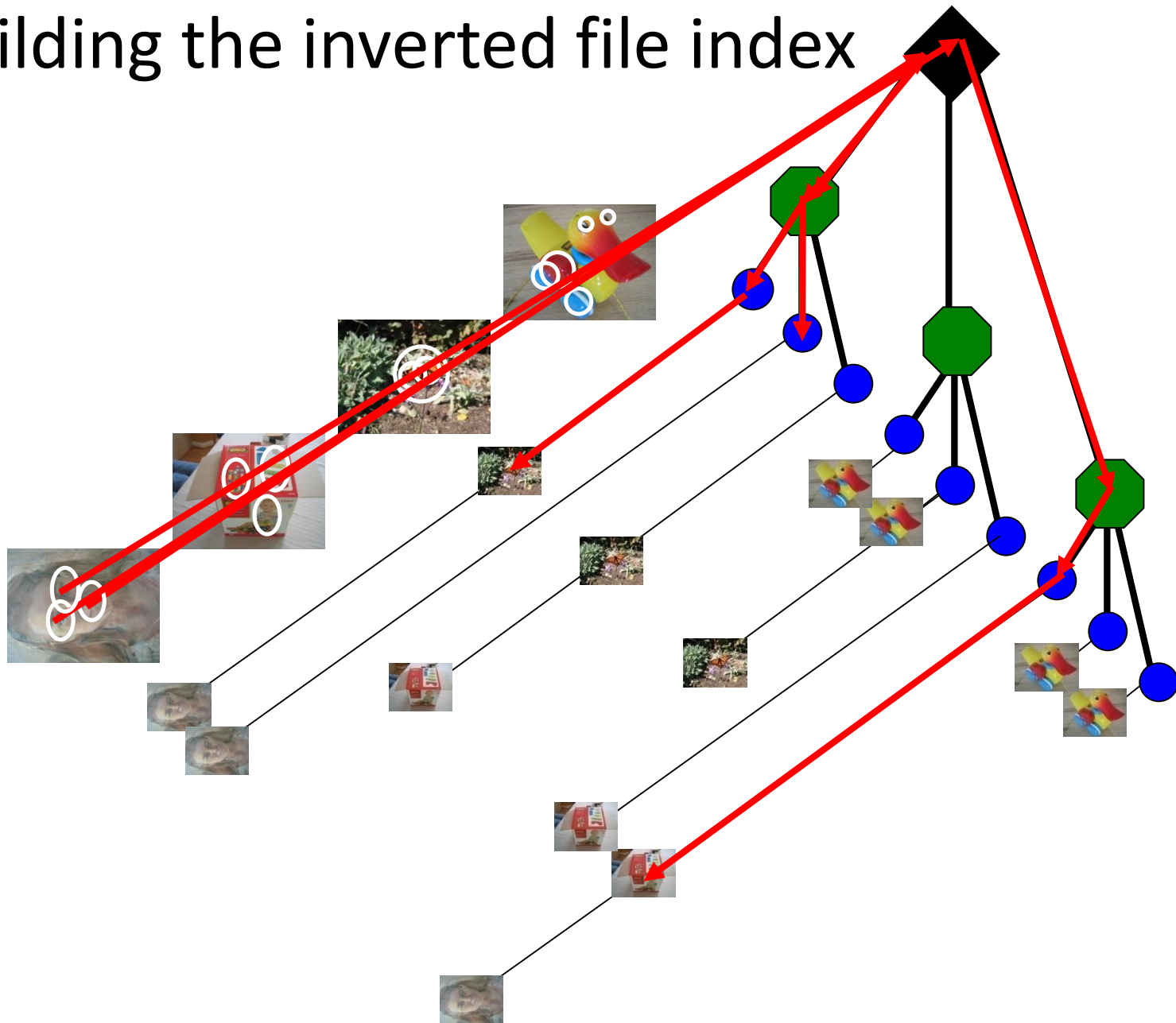
Building the inverted file index



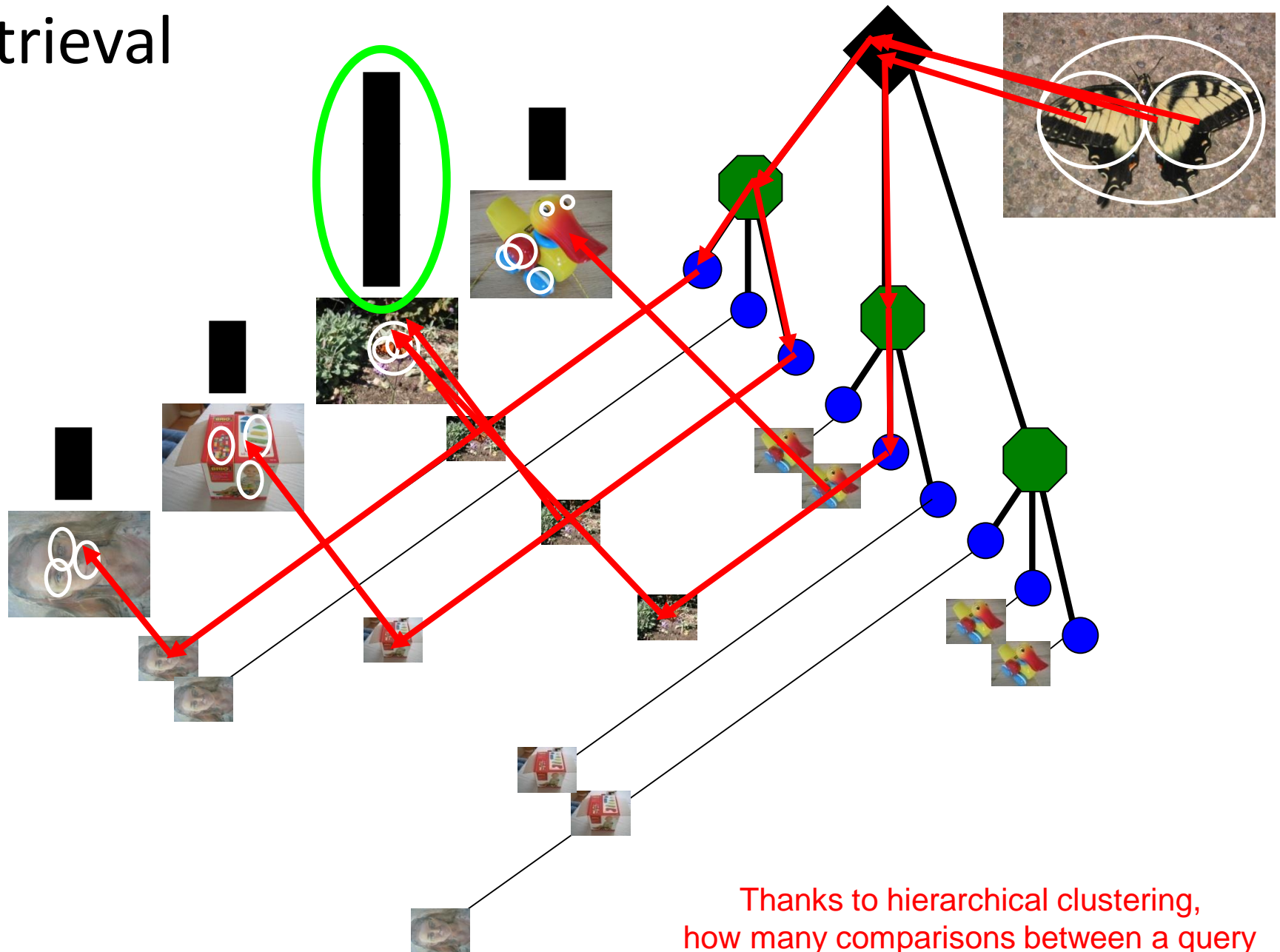
Building the inverted file index



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 levels?

Example

Query an image in a database of 100 million images

– Example:

- assume our query image has 1,000 SIFT features $\rightarrow M = 1,000$
- assume 10 branches and 6 levels (i.e., 1,000,000 visual words)
- \rightarrow Number of feature comparisons $= 1,000 \cdot 10 \cdot 6 = 60,000$
- If we assume 0.1 ms per feature comparison \rightarrow 1 image query would take **6 seconds!**

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

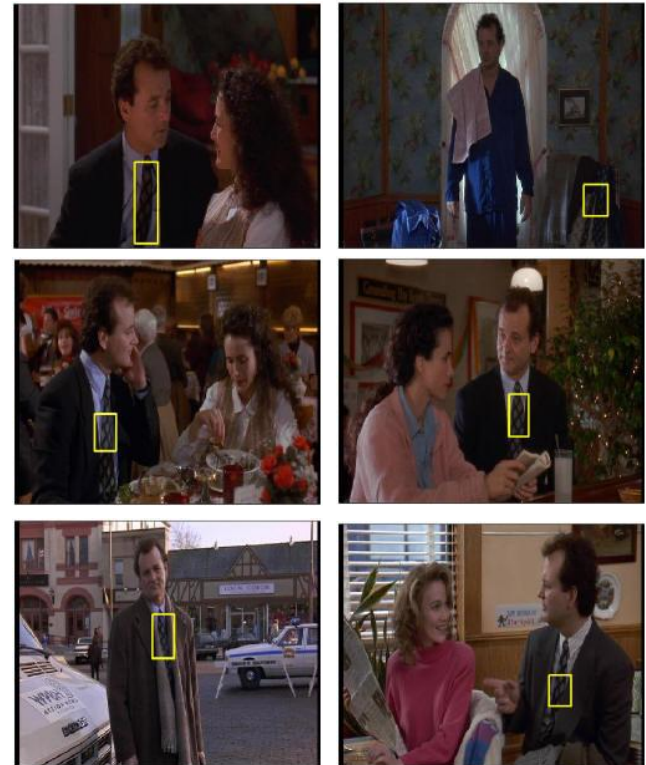
Sivic & Zisserman, ICCV 2003

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

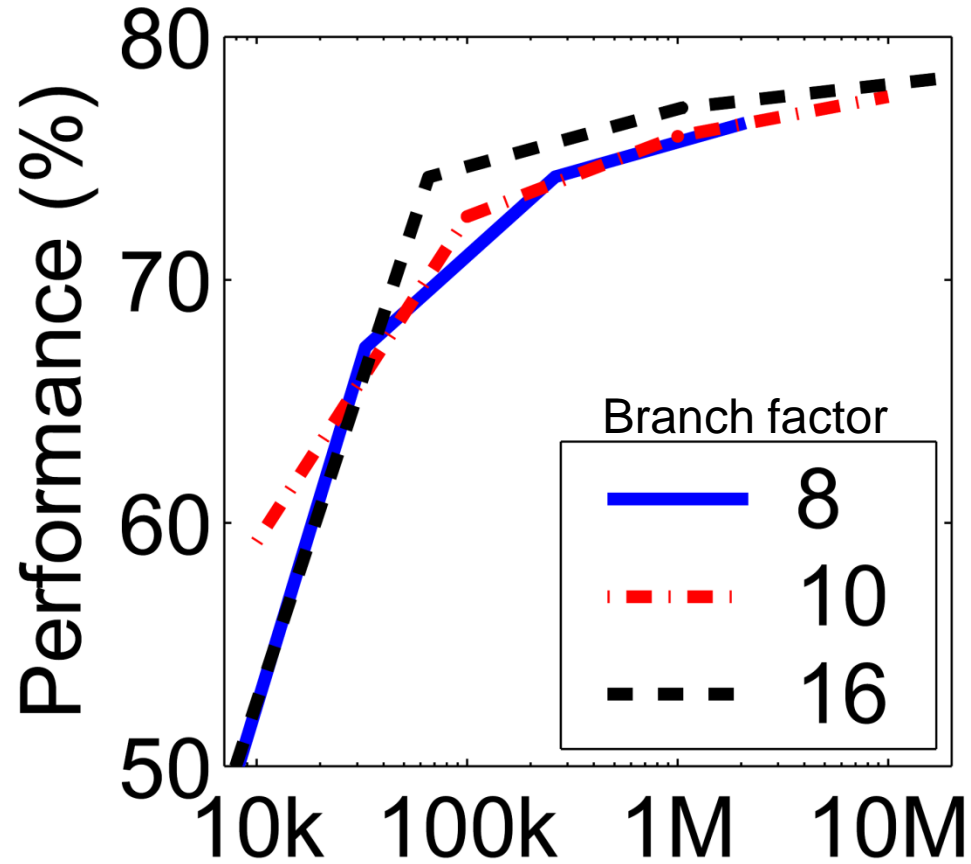
Query region



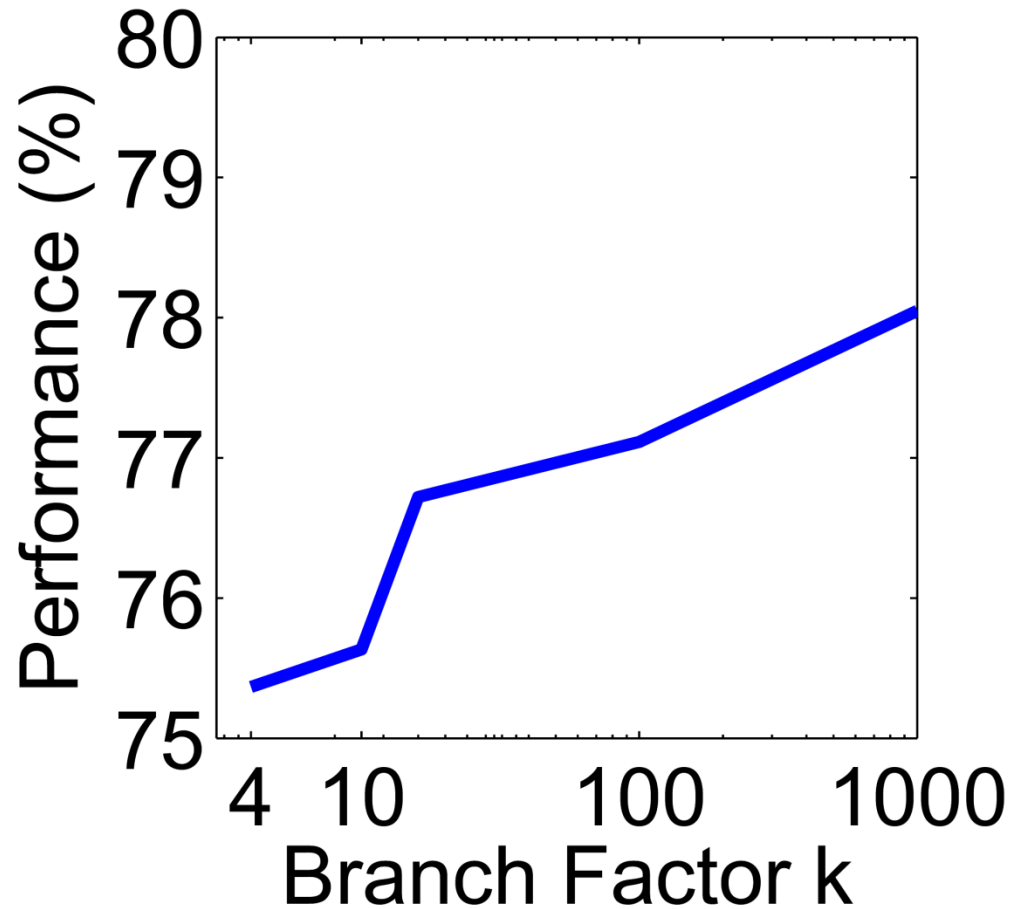
Retrieved frames



More words is better

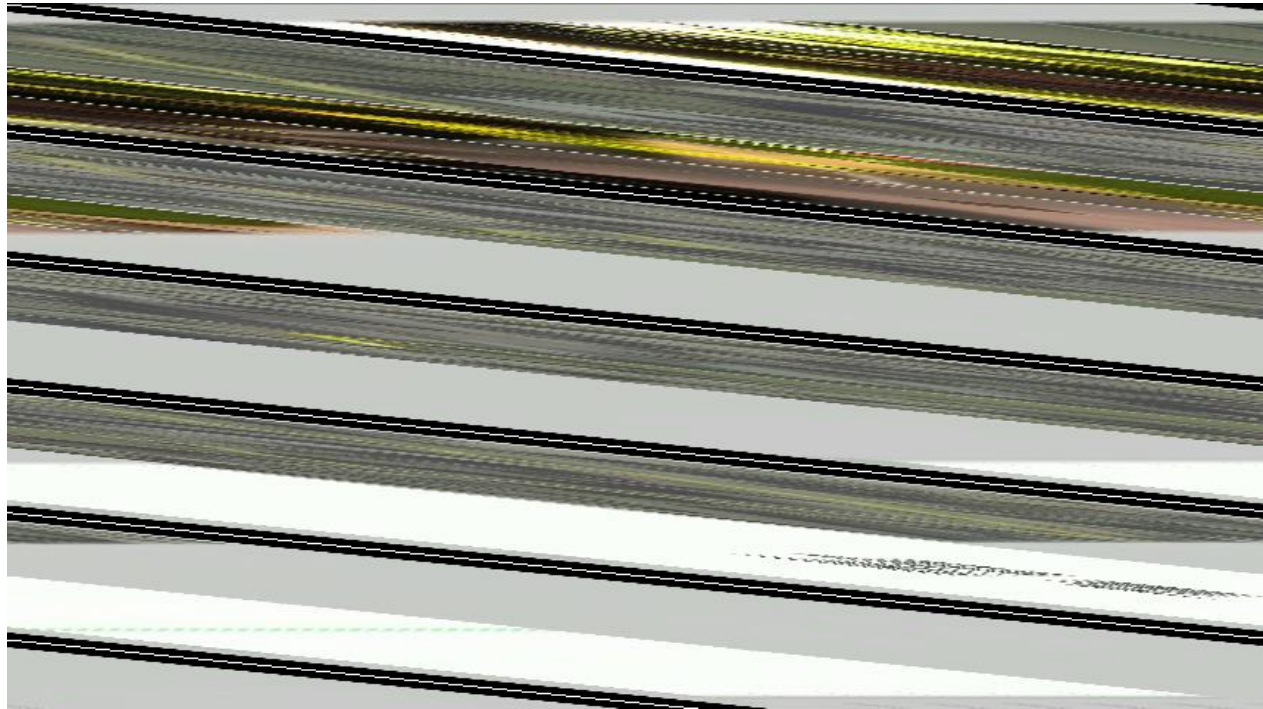


Higher branch factor works better (but slower)



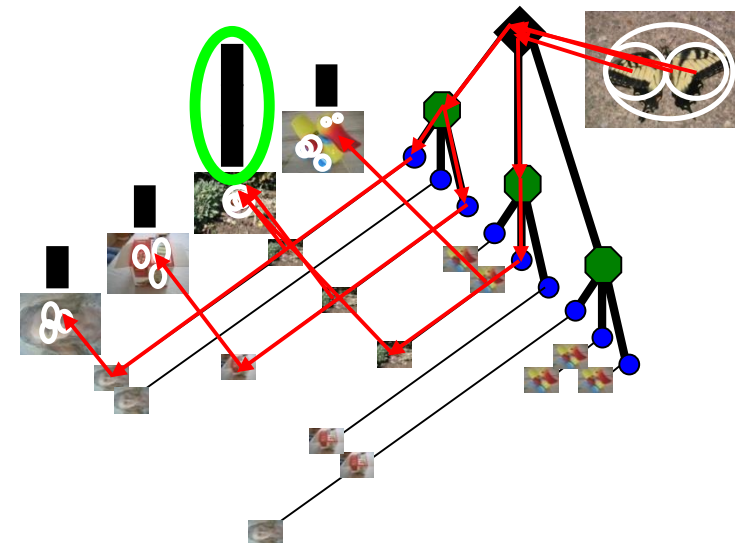
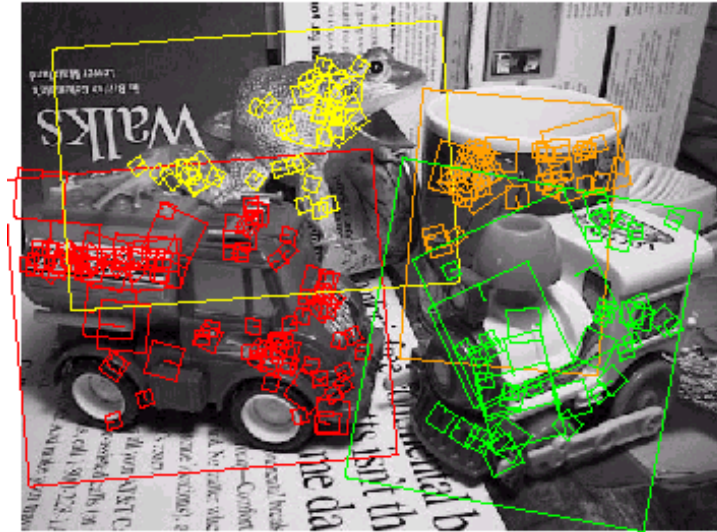
FABMAP [Cummins and Newman IJRR 2011]

- 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:
 - $P(\text{being at a known place})$
 - $P(\text{being at a new place})$
- 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?