Clustering with Application to Fast Object Search

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

This section

- Clustering: grouping together similar points, images, feature vectors, etc.

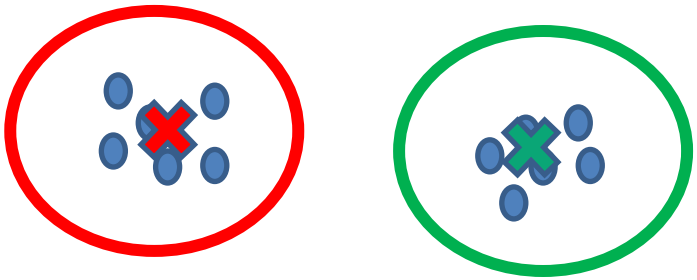
- Segmentation: dividing the image into meaningful regions
  - Segmentation by clustering: K-means and mean-shift
  - Graph approaches to segmentation: graph cuts and normalized cuts
  - Segmentation from boundaries: watershed

- EM: soft clustering, or parameter estimation with hidden data

Today's class

- Clustering algorithms
  - K-means
    - Application to fast object search
  - Hierarchical clustering
  - Spectral clustering

Clustering: group together similar points and represent them with a single token



Key Challenges:

1) What makes two points/images/patches similar?

2) How do we compute an overall grouping from pairwise similarities?

Why do we cluster?

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number
- **Counting**
  - Histograms of texture, color, SIFT vectors
- **Segmentation**
  - Separate the image into different regions
- **Prediction**
  - Images in the same cluster may have the same labels

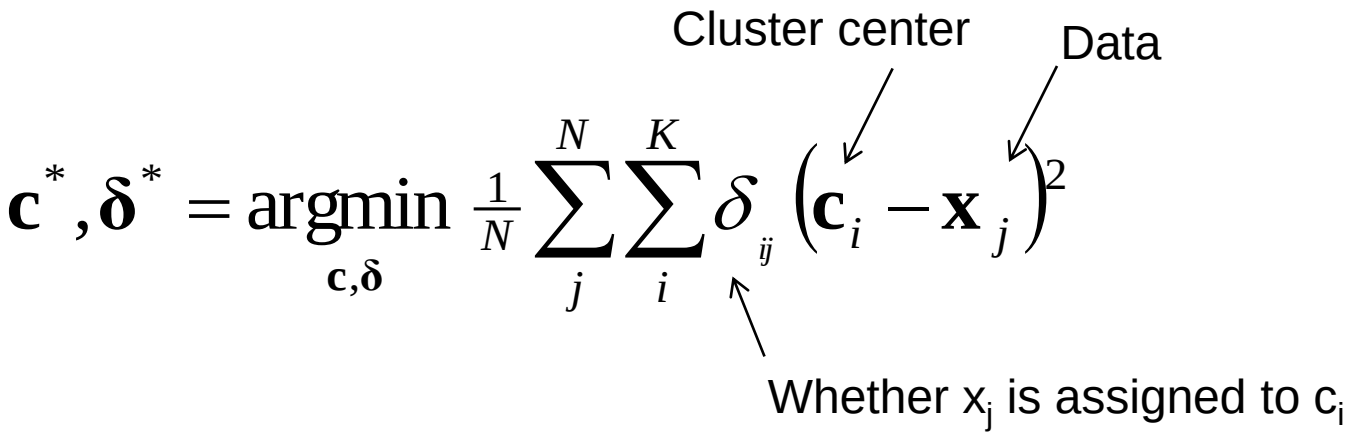
How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

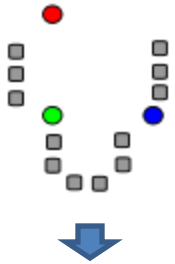
Goal: cluster to minimize variance in data given clusters

– Preserve information

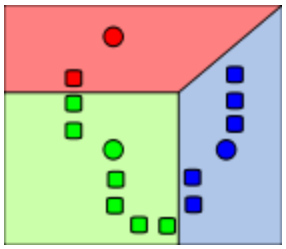


K-means algorithm

1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster

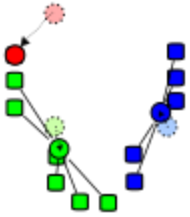
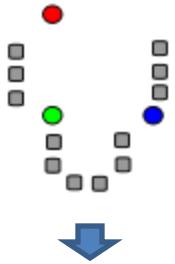


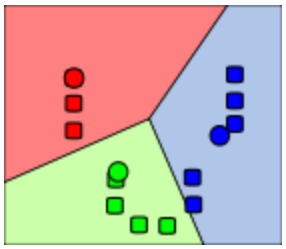
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means algorithm

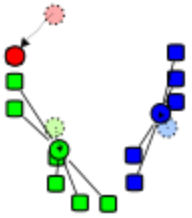
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means demos

General
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Color clustering
http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster/

K-means

1. Initialize cluster centers: $\mathbf{c}^0$; t=0

2. Assign each point to the closest center

$$\boldsymbol{\delta}^t = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

3. Update cluster centers as the mean of the points

$$\mathbf{c}^t = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t \left(\mathbf{c}_i - \mathbf{x}_j \right)^2$$

4. Repeat 2-3 until no points are re-assigned (t=t+1)

Kmeans: Matlab code

```
function C = kmeans(X, K)

% Initialize cluster centers to be randomly sampled points
[N, d] = size(X);
rp = randperm(N);
C = X(rp(1:K), :);

lastAssignment = zeros(N, 1);
while true

  % Assign each point to nearest cluster center
  bestAssignment = zeros(N, 1);
  mindist = Inf*ones(N, 1);
  for k = 1:K
    for n = 1:N
      dist = sum((X(n, :)-C(k, :)).^2);
      if dist < mindist(n)
        mindist(n) = dist;
        bestAssignment(n) = k;
      end
    end
  end

  % break if assignment is unchanged
  if all(bestAssignment==lastAssignment), break; end;

  % Assign each cluster center to mean of points within it
  for k = 1:K
    C(k, :) = mean(X(bestAssignment==k, :));
  end
end
```

K-means: design choices

- Initialization
  - Randomly select K points as initial cluster center
  - Or greedily choose K points to minimize residual
- Distance measures
  - Traditionally Euclidean, could be others
- Optimization
  - Will converge to a *local minimum*
  - May want to perform multiple restarts

How to choose the number of clusters?

- Minimum Description Length (MDL) principal for model comparison

- Minimize Schwarz Criterion
  - also called Bayes Information Criteria (BIC)

$$\text{Distortion} + \lambda\,(\#\text{parameters})\log R$$

$$= \text{Distortion} + \lambda mk \log R$$

sum squared error

m=#dimensions

k=#Centers

R=#Records

How to choose the number of clusters?

- Validation set
  - Try different numbers of clusters and look at performance
    - When building dictionaries (discussed later), more clusters typically work better

How to evaluate clusters?

- Generative
  - How well are points reconstructed from the clusters?
- Discriminative
  - How well do the clusters correspond to labels?
    - Purity
  - Note: unsupervised clustering does not aim to be discriminative

Common similarity/distance measures

- P-norms
  - City Block (L1)
  - Euclidean (L2)
  - L-infinity

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right)$$

Here x_i is the distance between two points

- Mahalanobis
  - Scaled Euclidean

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

- Cosine distance

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Conclusions: K-means

Good

- Finds cluster centers that minimize conditional variance (good representation of data)
- Simple to implement, widespread application

Bad

- Prone to local minima
- Need to choose K
- All clusters have the same parameters (e.g., distance measure is non-adaptive)
- Can be slow: each iteration is O(KNd) for N d-dimensional points

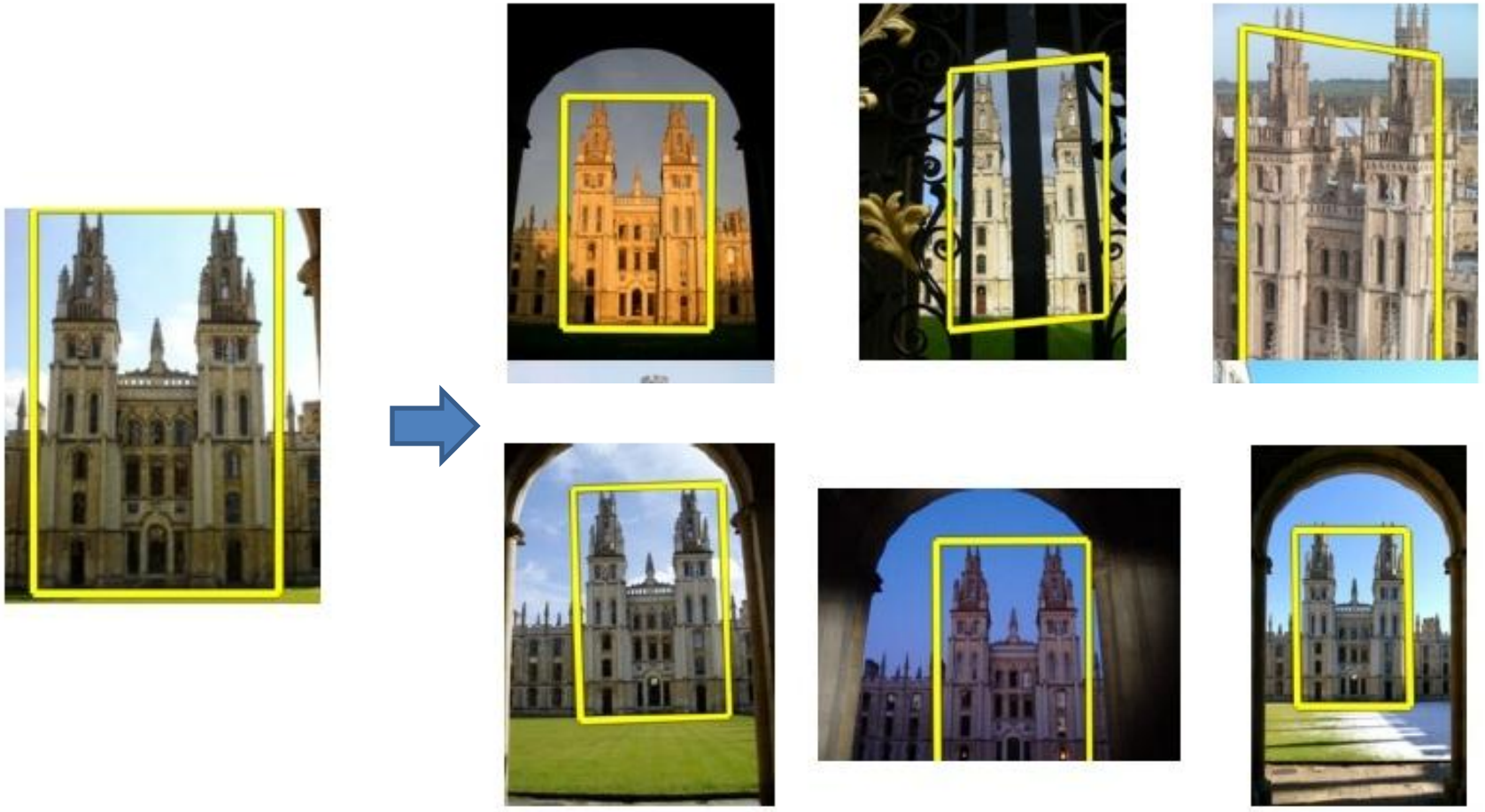
K-medoids

- Just like K-means except
  - Represent the cluster with one of its members, rather than the mean of its members
  - Choose the member (data point) that minimizes cluster dissimilarity

- Applicable when a mean is not meaningful
  - E.g., clustering values of hue or using L-infinity similarity

Application of Clustering

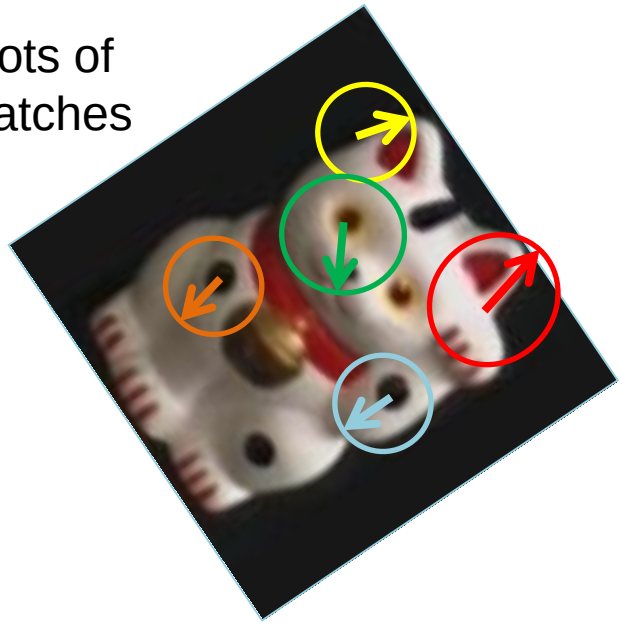
How to quickly find images in a large database that match a given image region?



Simple idea

See how many SIFT keypoints are close to SIFT keypoints in each other image

Lots of Matches



Few or No Matches

But this will be really, really slow!

Key idea 1: “Visual Words”

- Cluster the keypoint descriptors
- Assign each descriptor to a cluster number
  - What does this buy us?
  - Each descriptor was 128 dimensional floating point, now is 1 integer (easy to match!)
  - Is there a catch?
    - Need **a lot** of clusters (e.g., 1 million) if we want points in the same cluster to be very similar
    - Points that really are similar might end up in different clusters

Key idea 1: “Visual Words”

- Cluster the keypoint descriptors
- Assign each descriptor to a cluster number
- Represent an image region with a count of these “visual words”

Key idea 1: “Visual Words”

- Cluster the keypoint descriptors
- Assign each descriptor to a cluster number
- Represent an image region with a count of these “visual words”
- An image is a good match if it has a lot of the same visual words as the query region

Naïve matching is still too slow

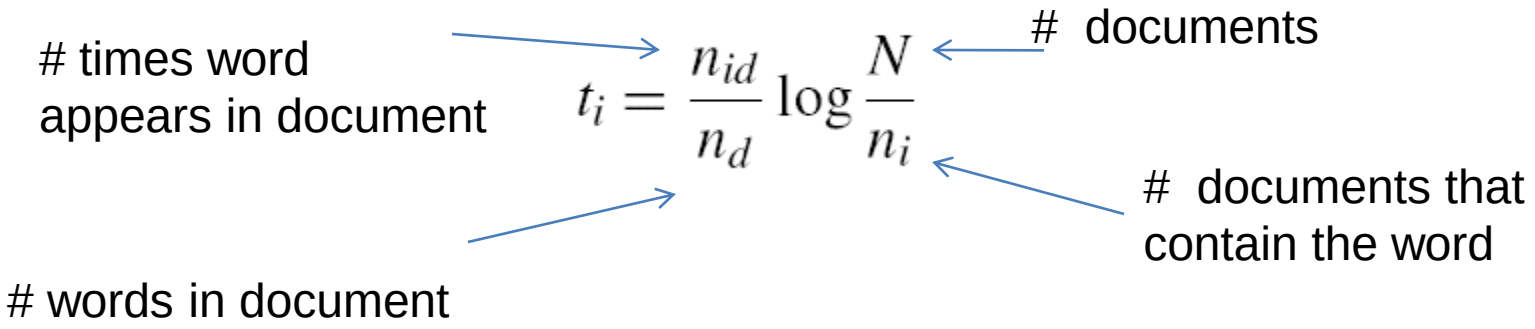
- Imagine matching 1,000,000 images, each with 1,000 keypoints

Key Idea 2: Inverse document file

- Like a book index: keep a list of all the words (keypoints) and all the pages (images) that contain them.

- Rank database images based on tf-idf measure.

tf-idf: Term Frequency – Inverse Document Frequency



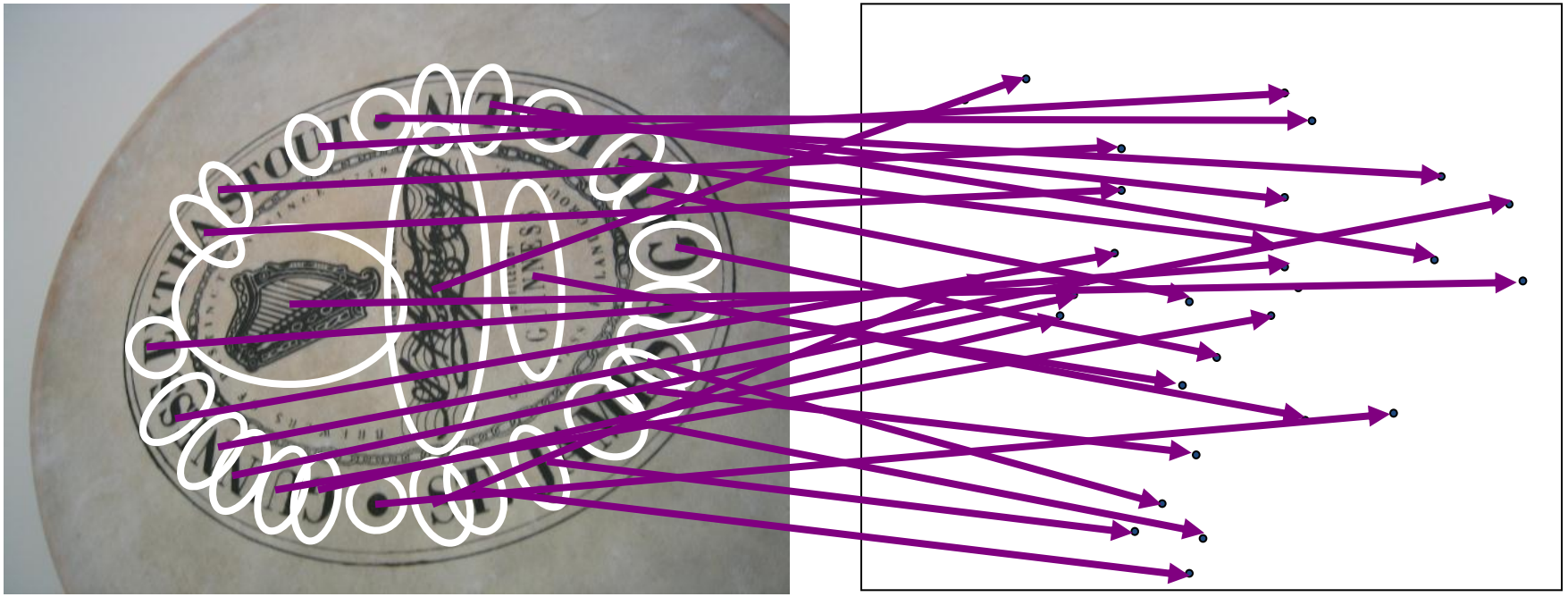
Fast visual search

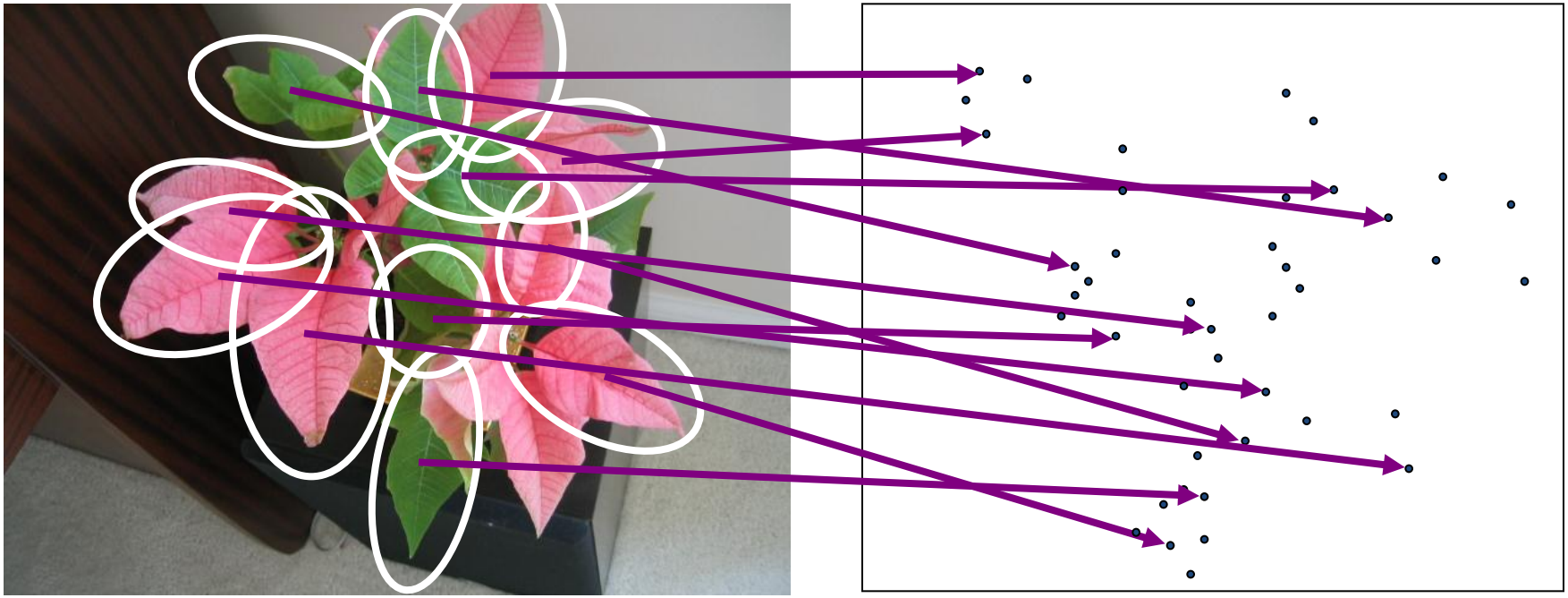
“Video Google”, Sivic and Zisserman, ICCV 2003

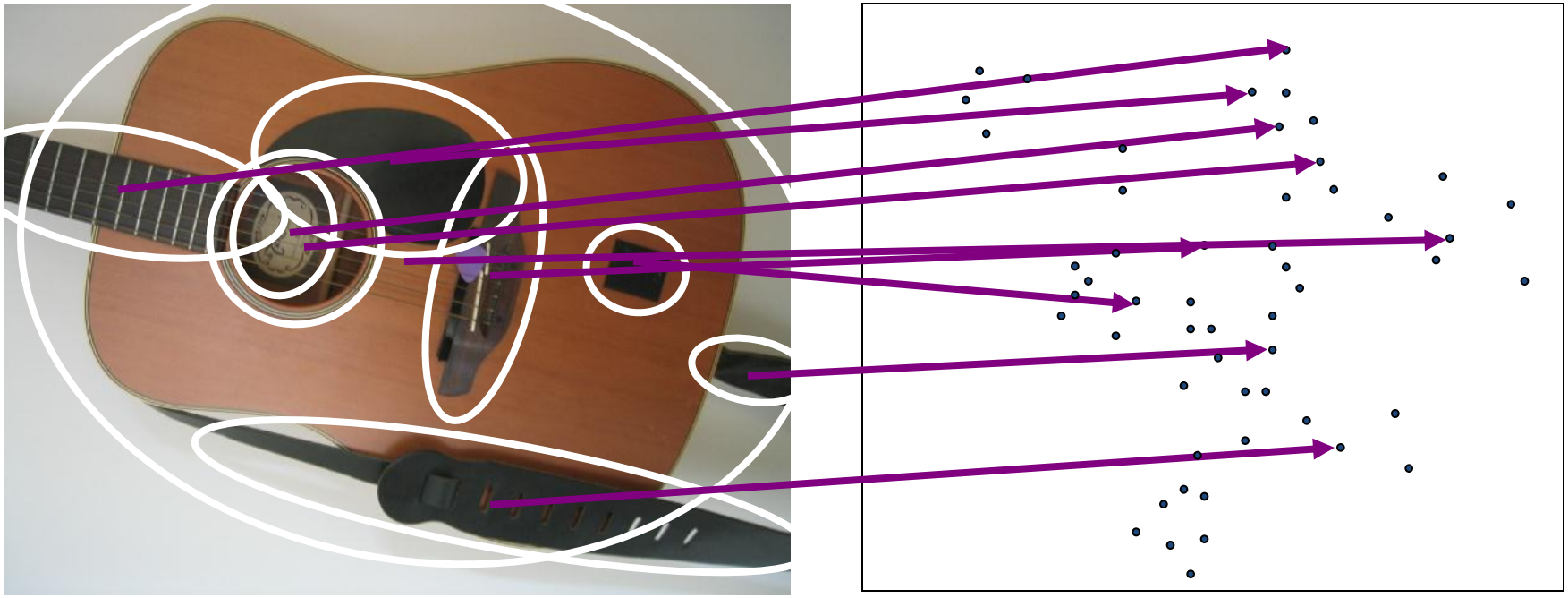
“Scalable Recognition with a Vocabulary Tree”, Nister and Stewenius, CVPR 2006.

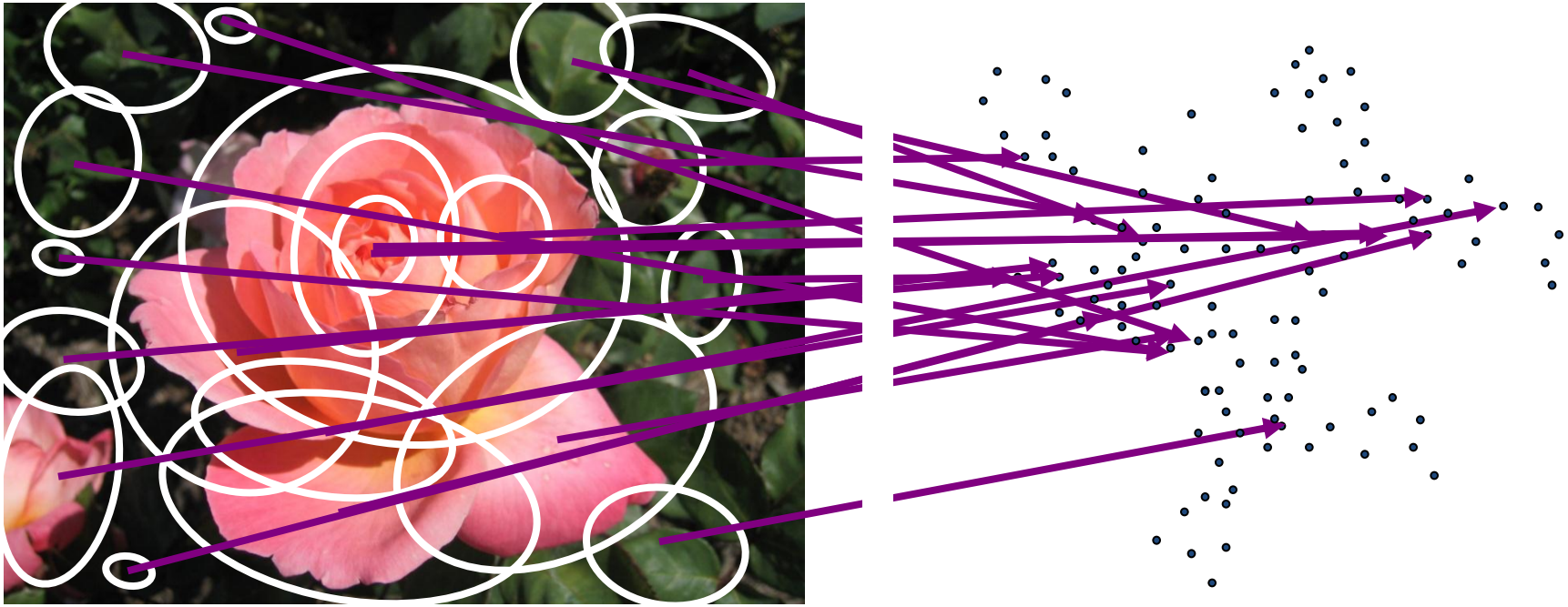
110,000,000 Images in 5.8 Seconds

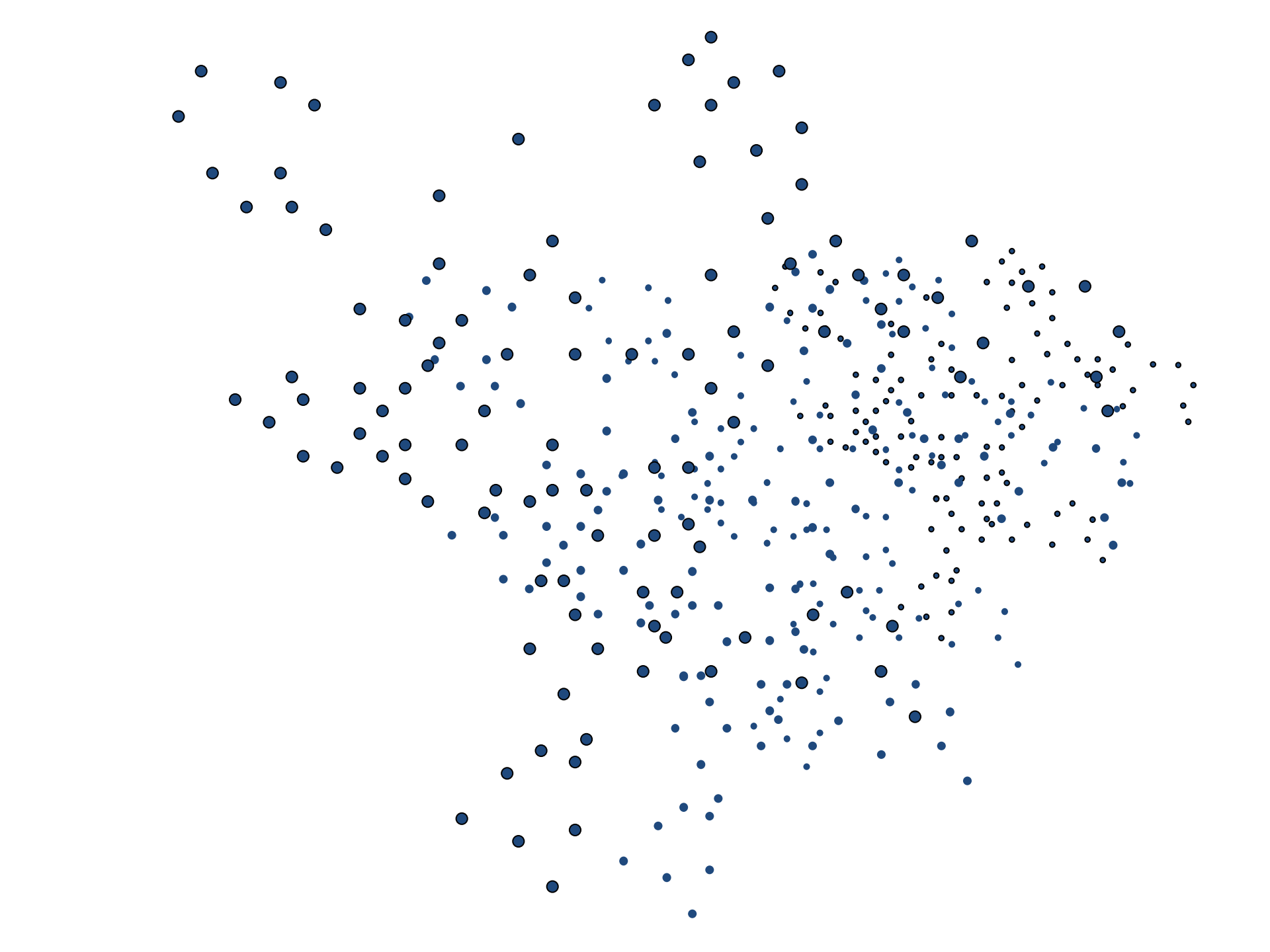
This slide and following by David Nister

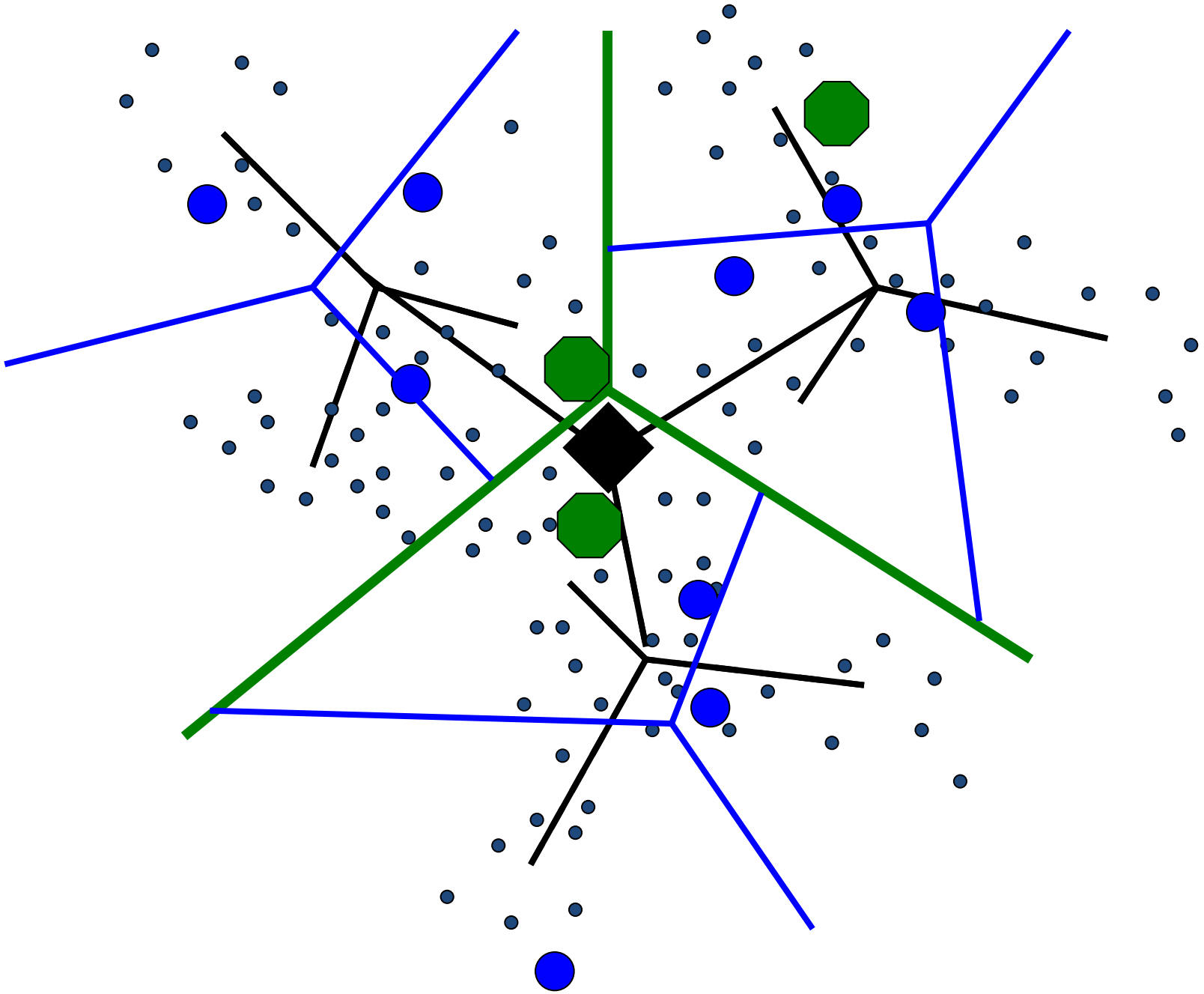


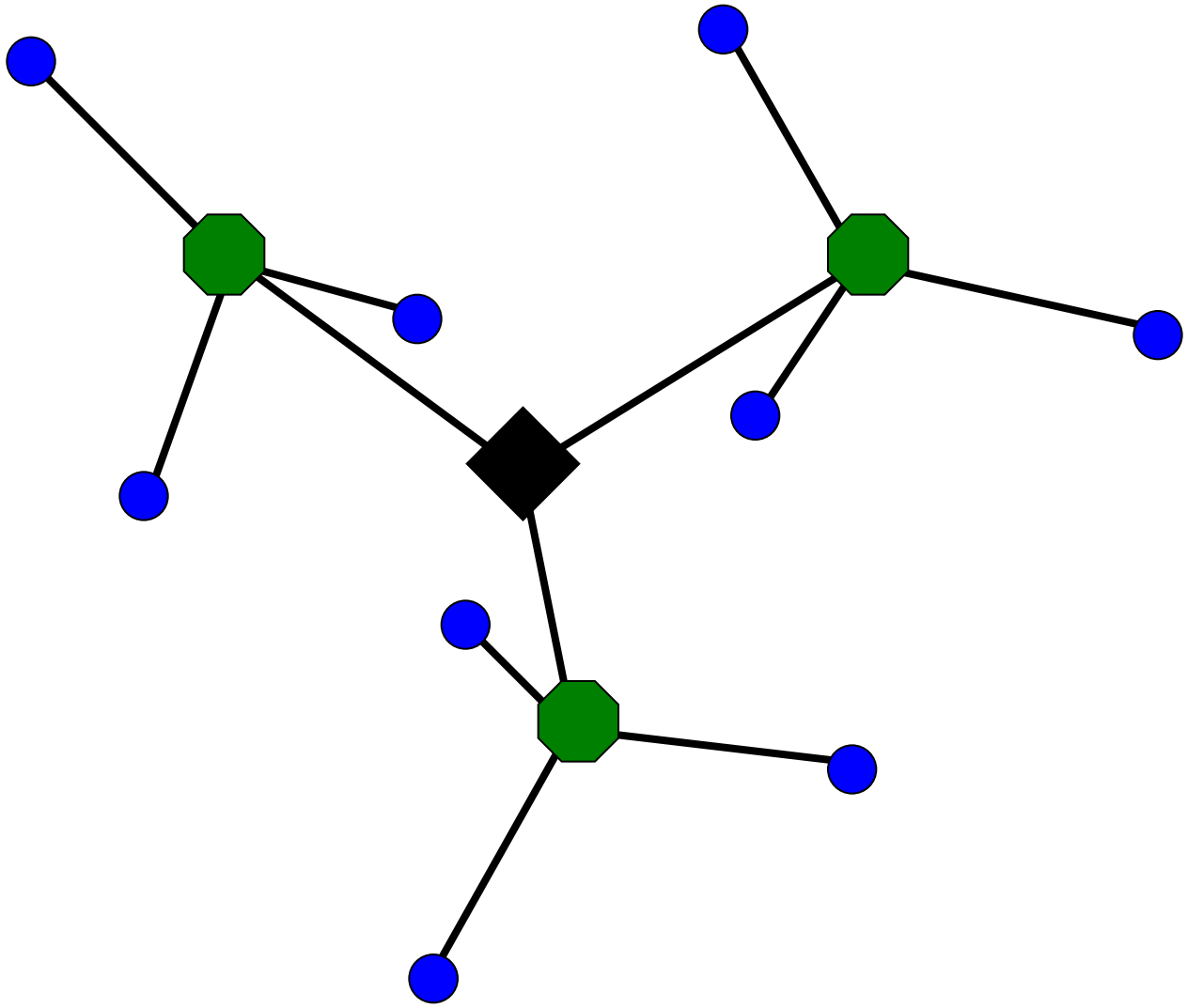


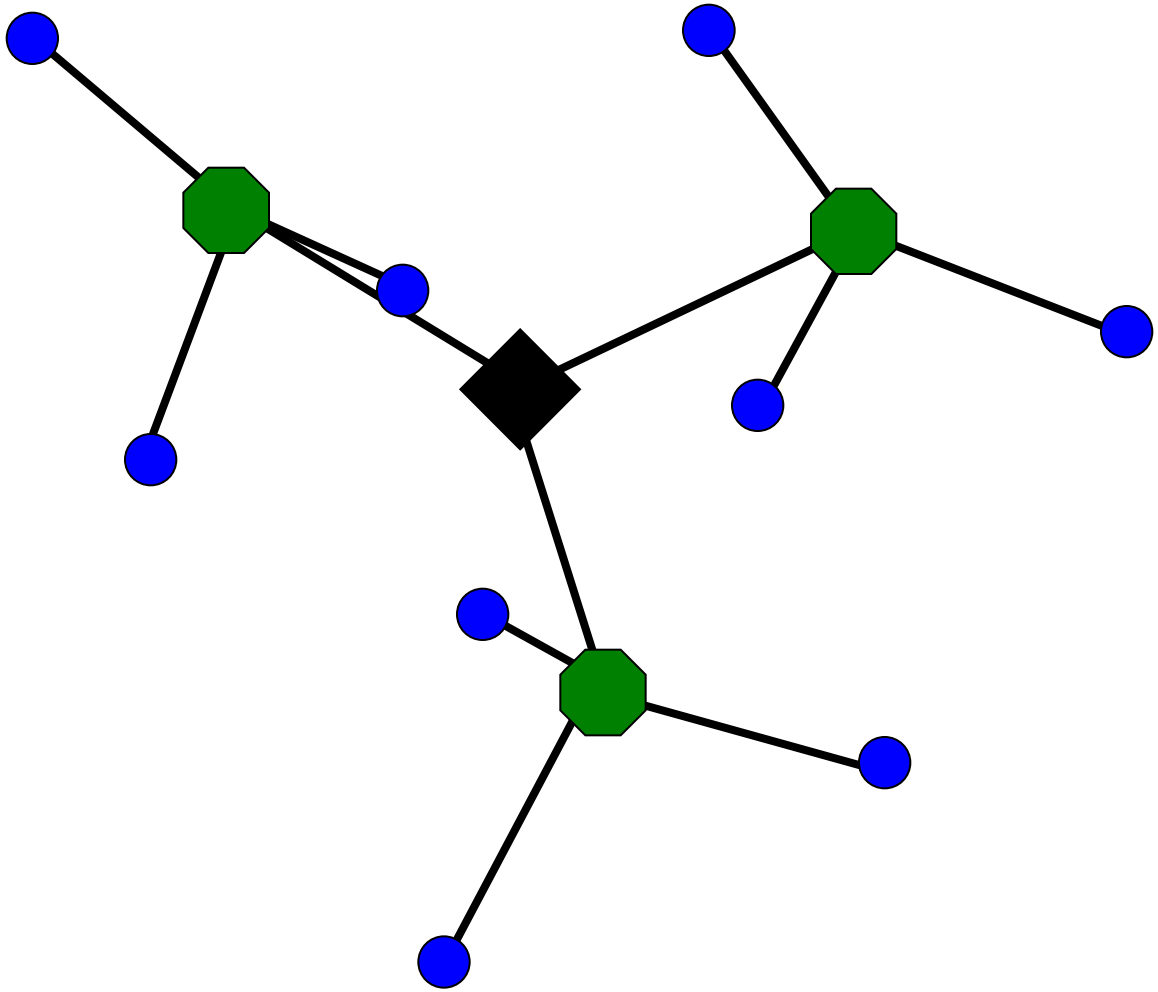


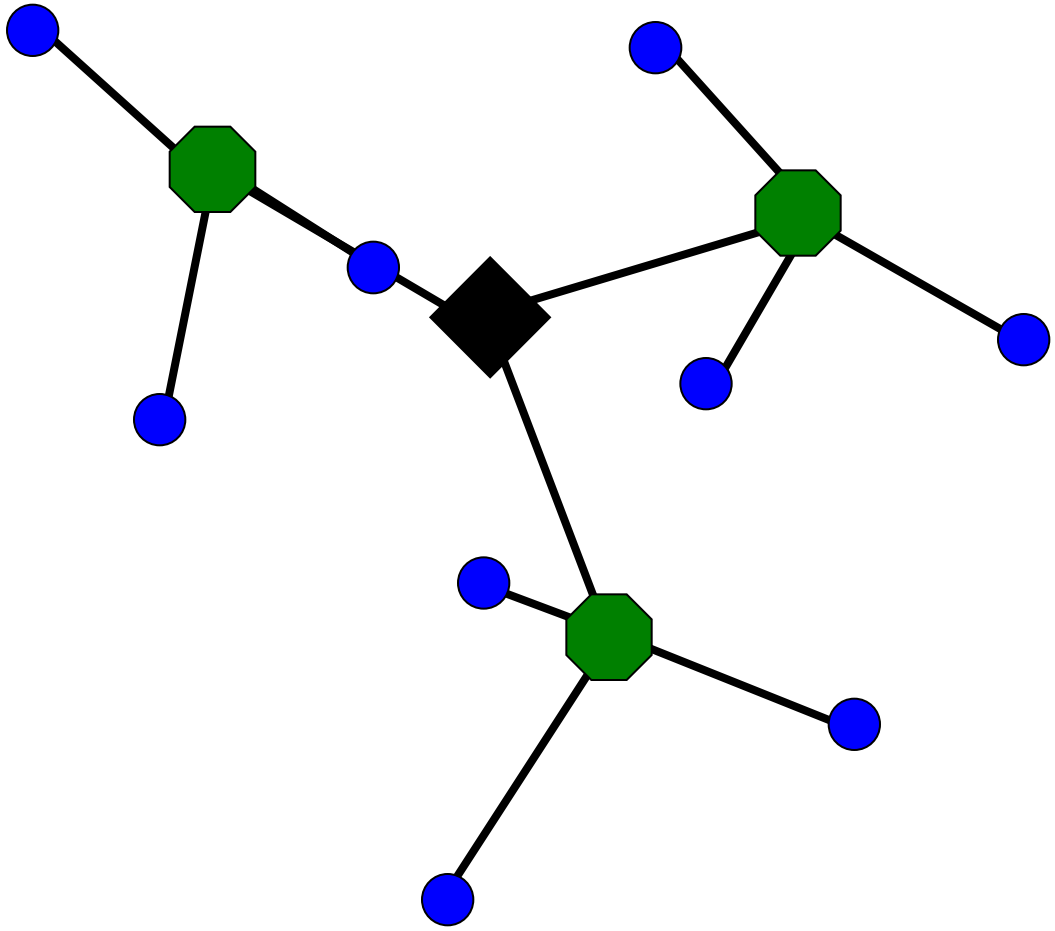


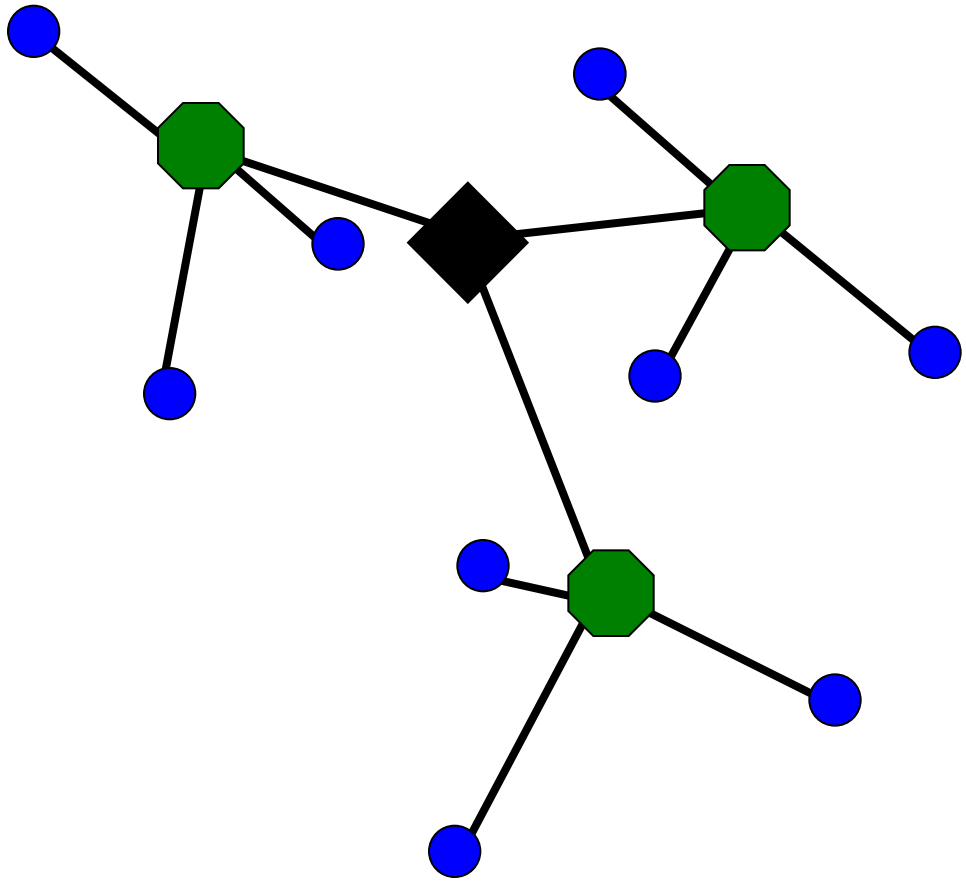


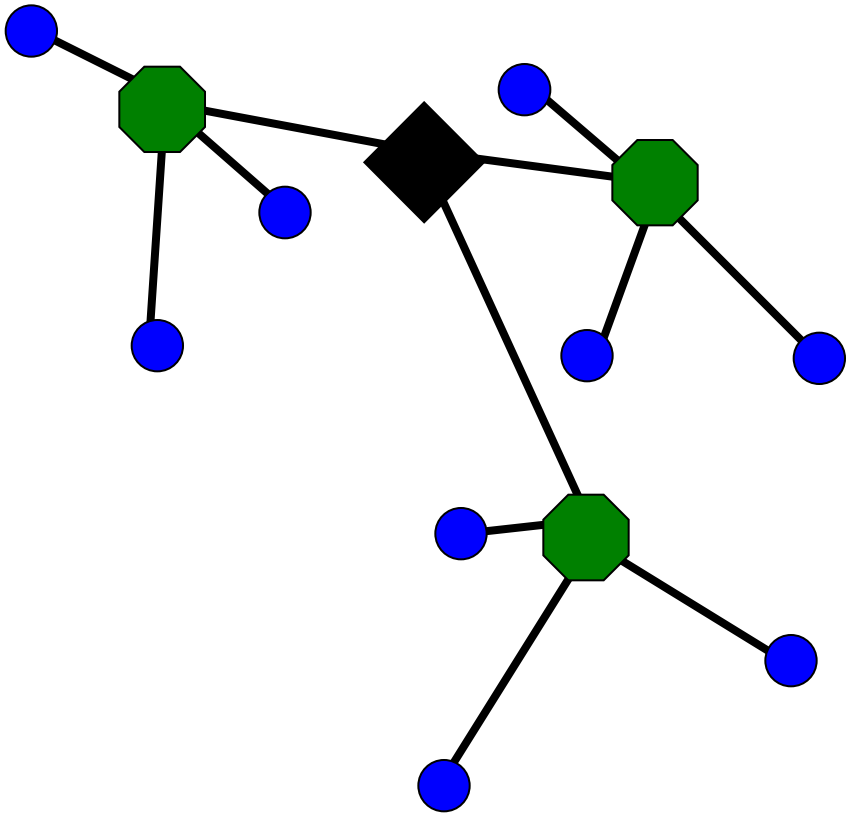


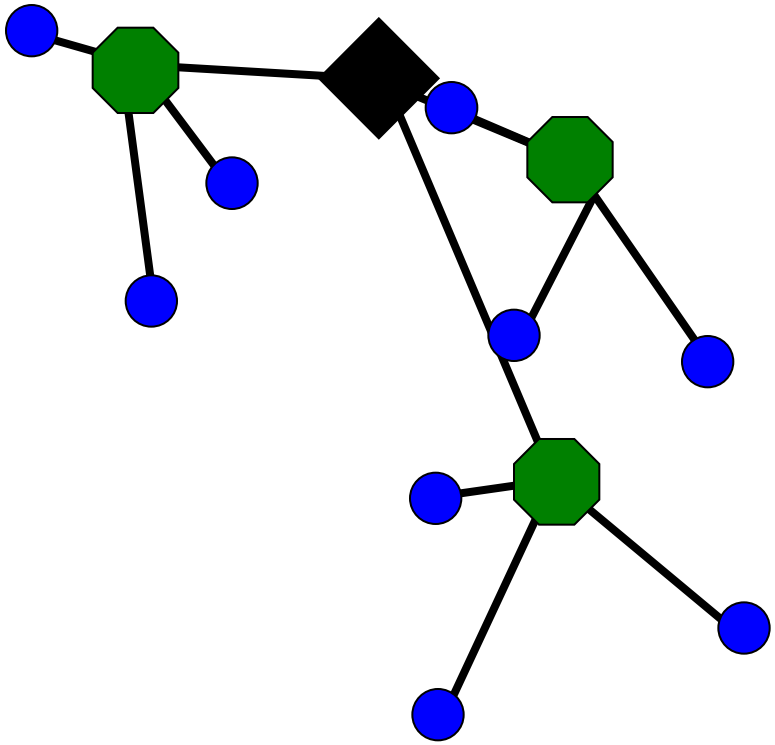


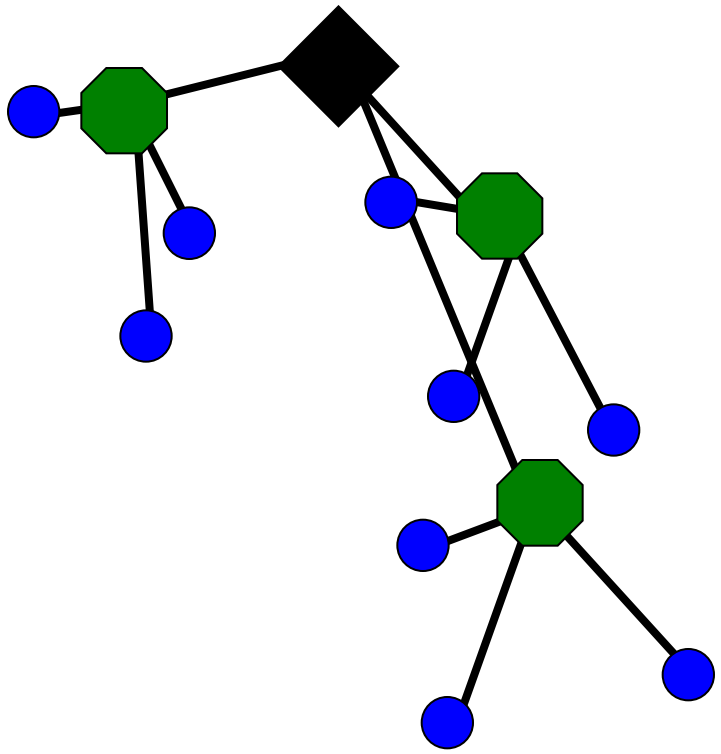


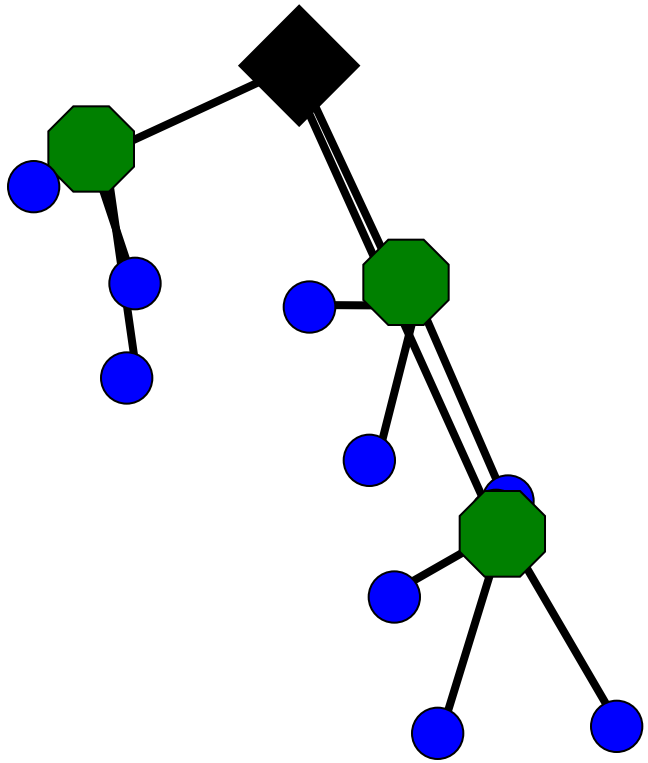


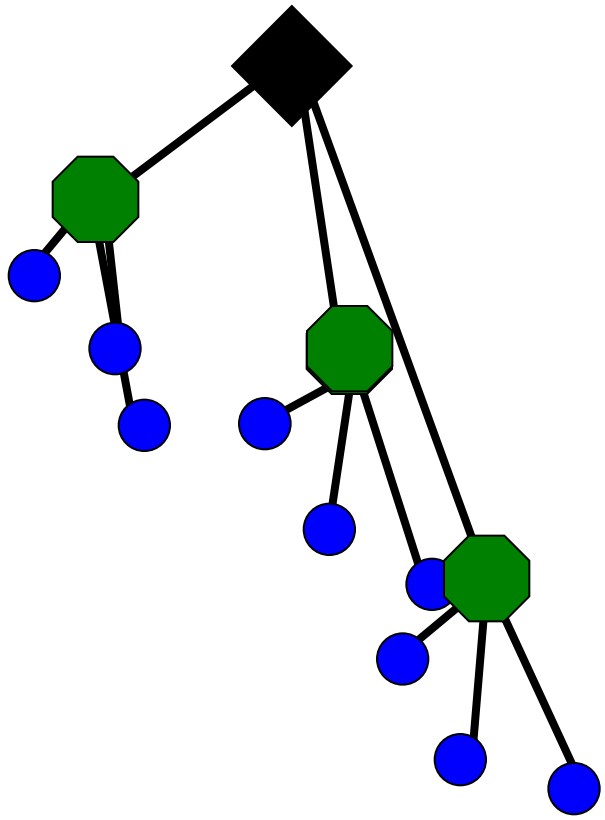


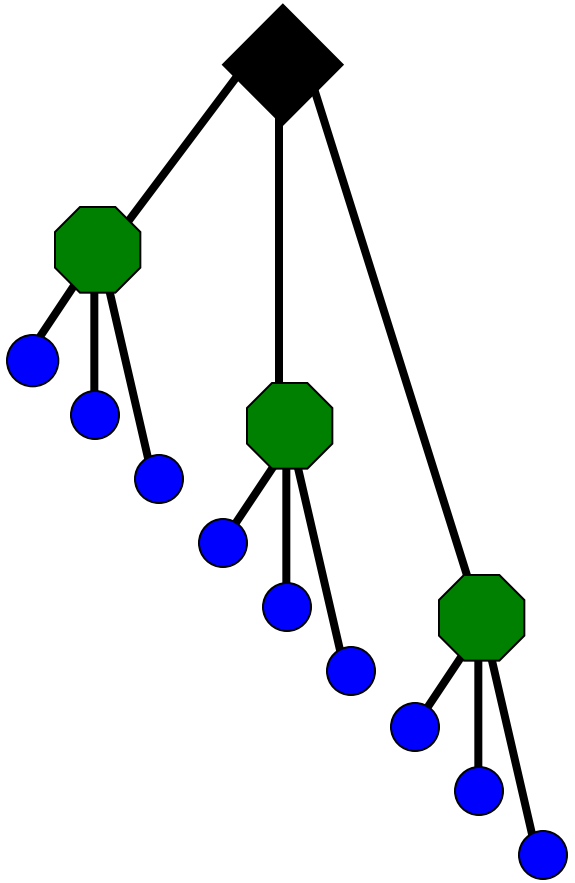


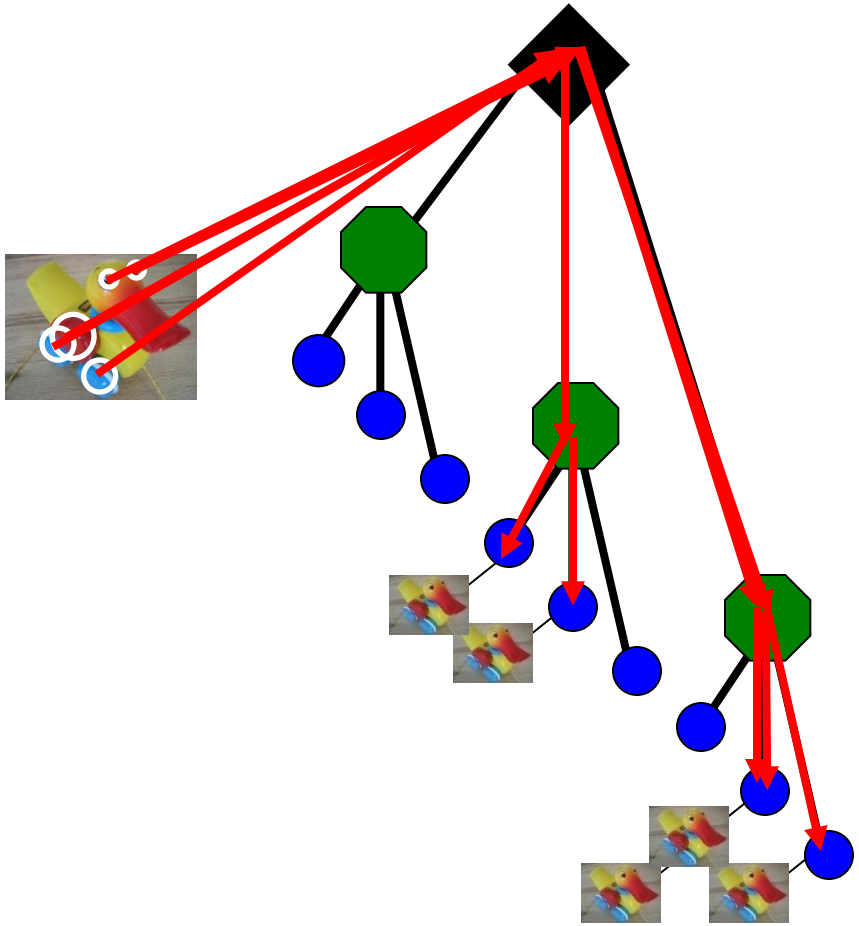


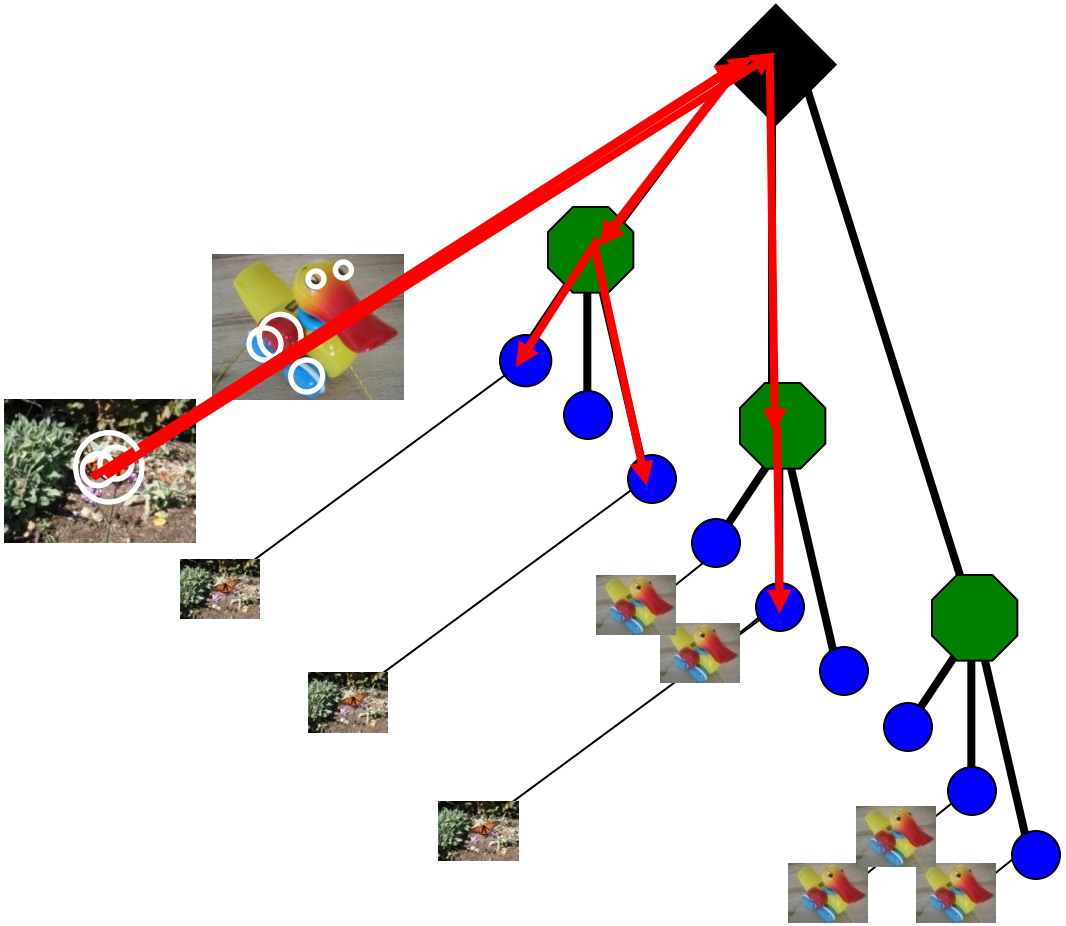


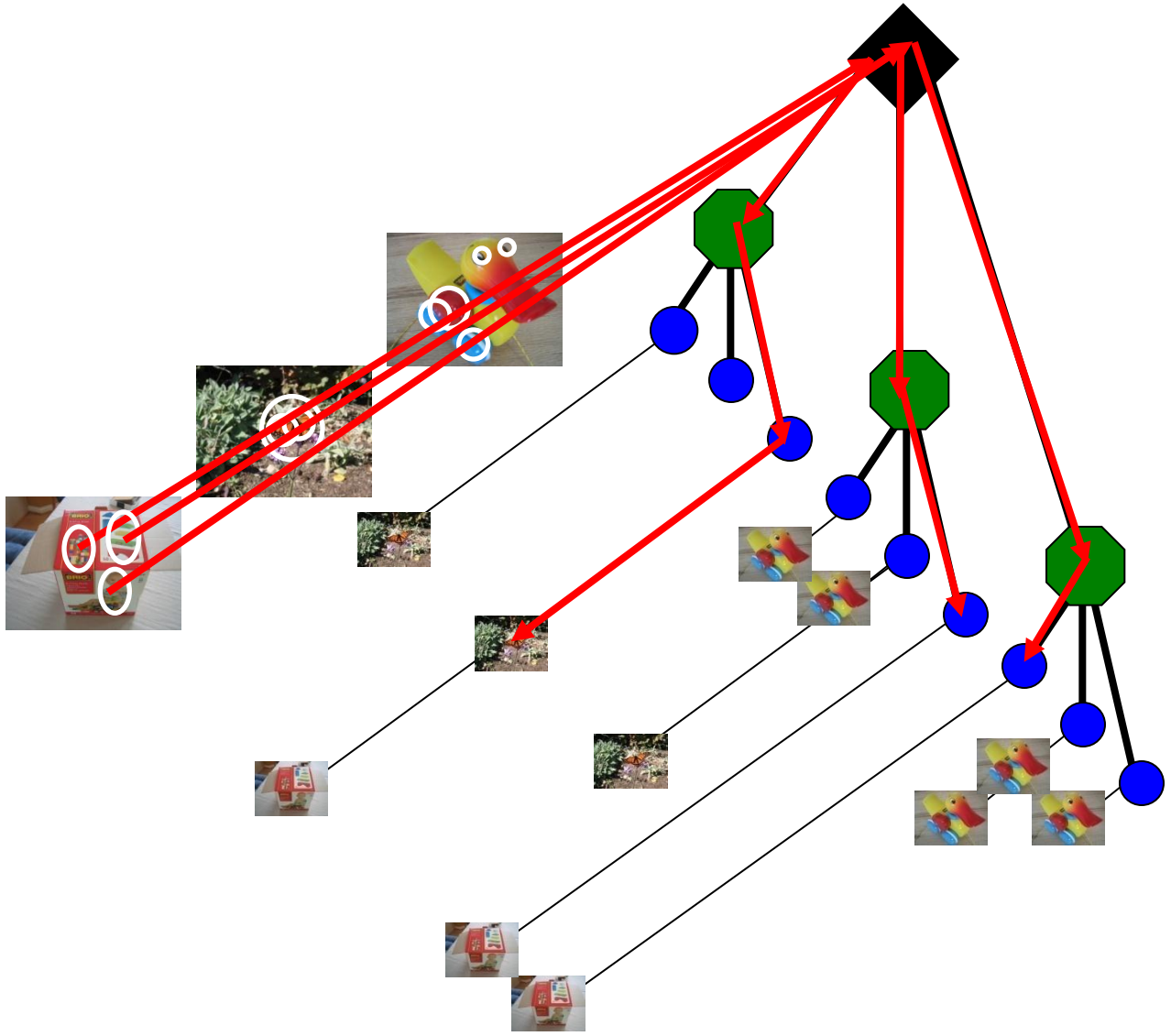


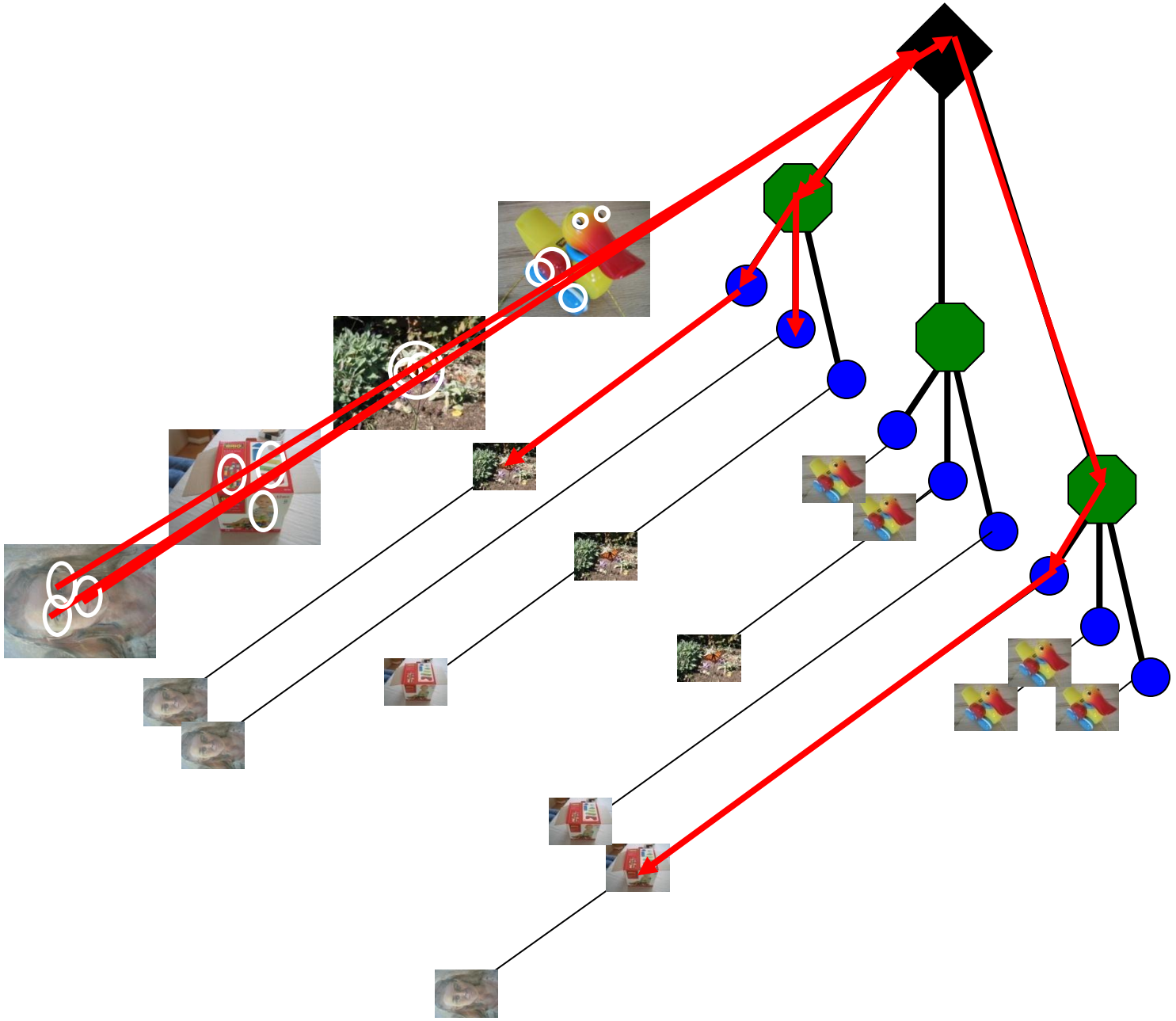


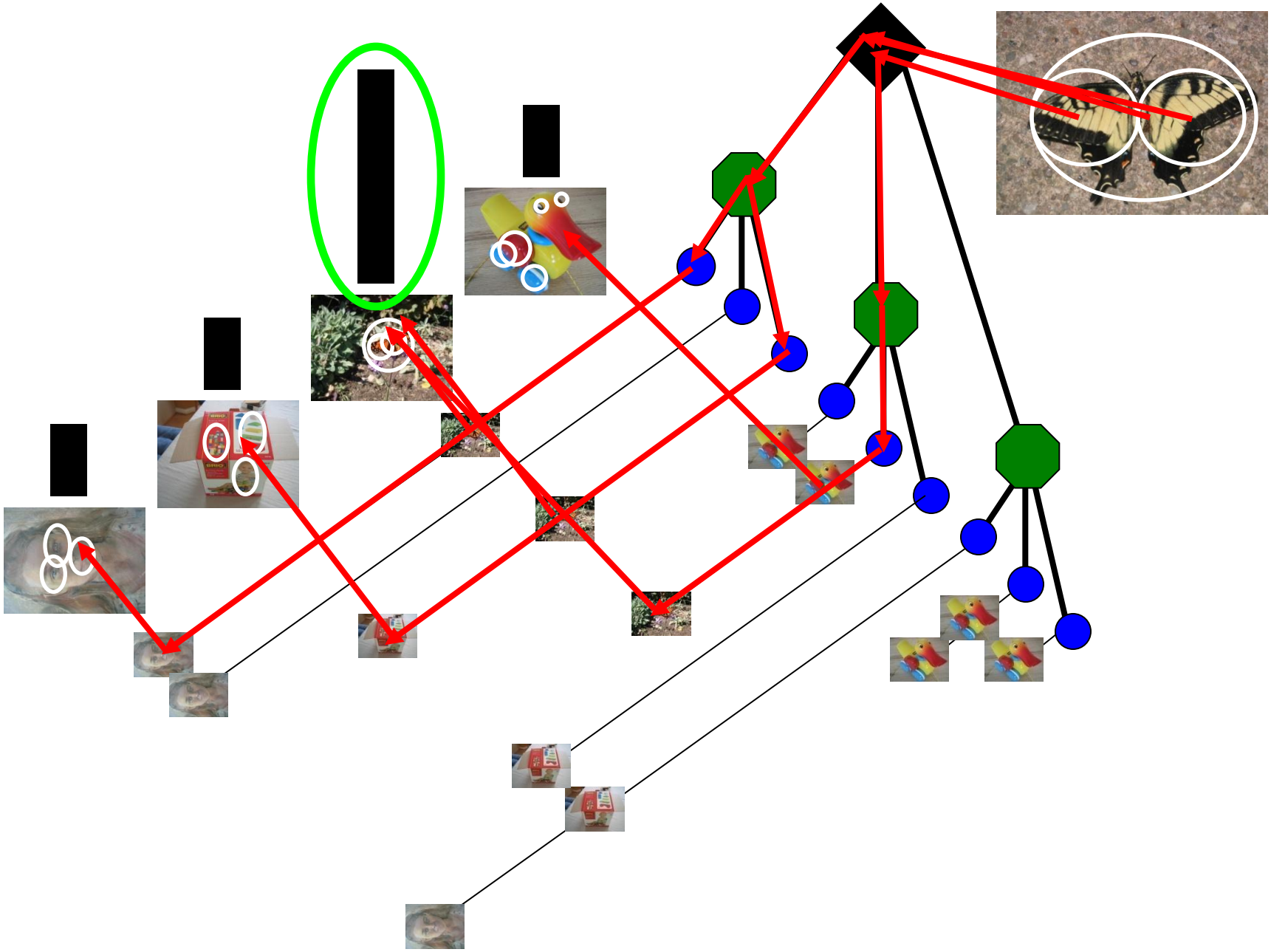




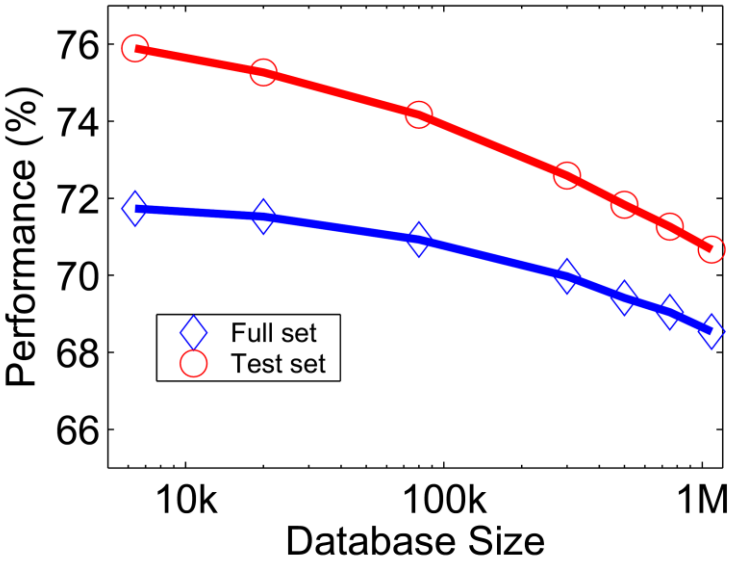








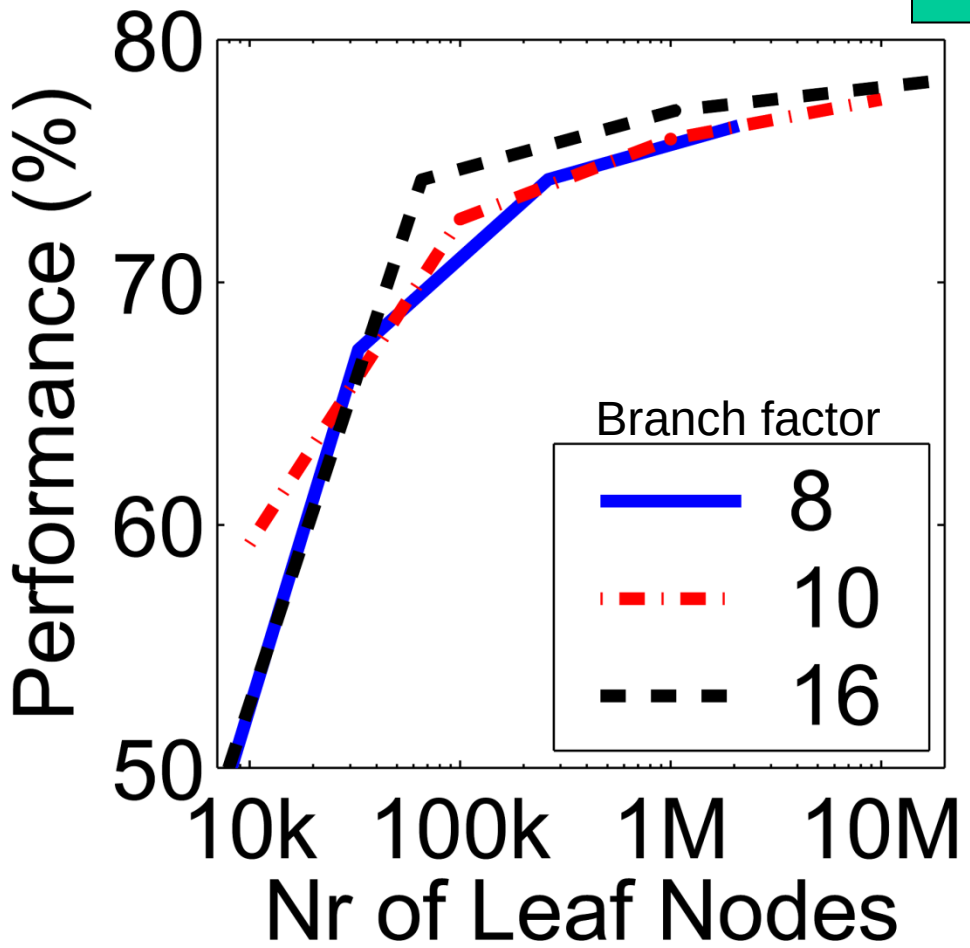
Performance



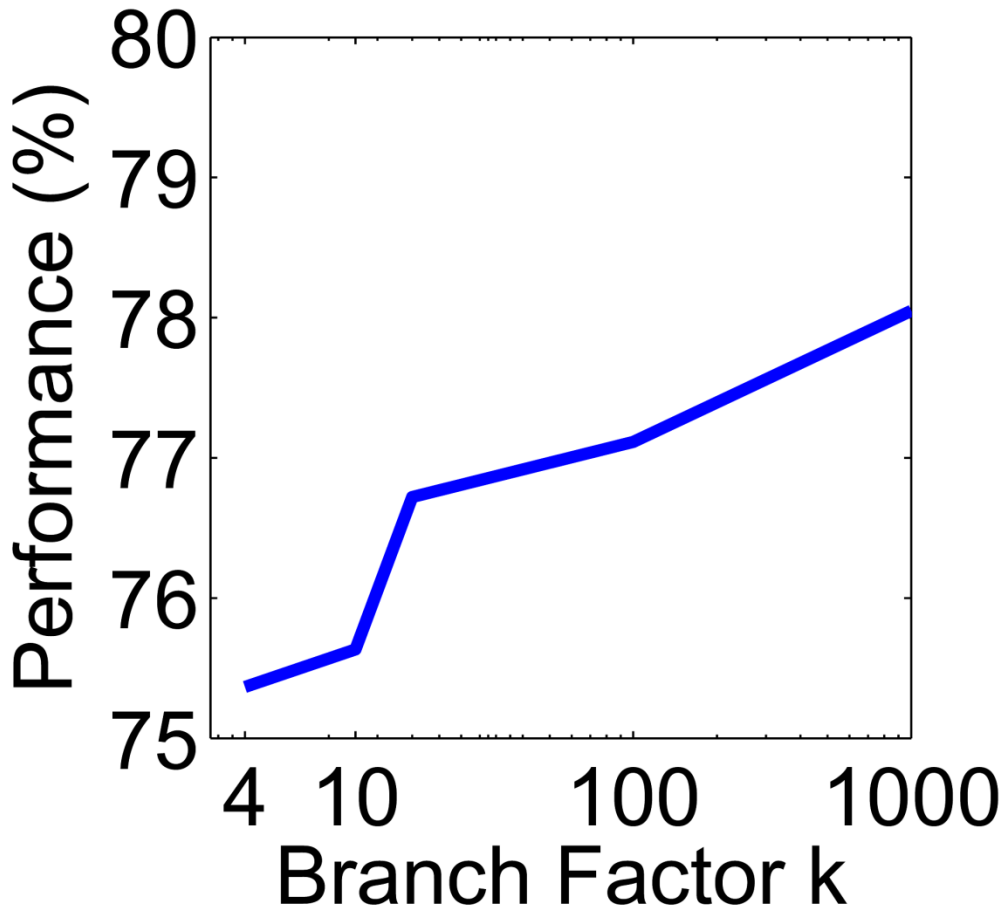
More words is better

Improves Retrieval

Improves Speed

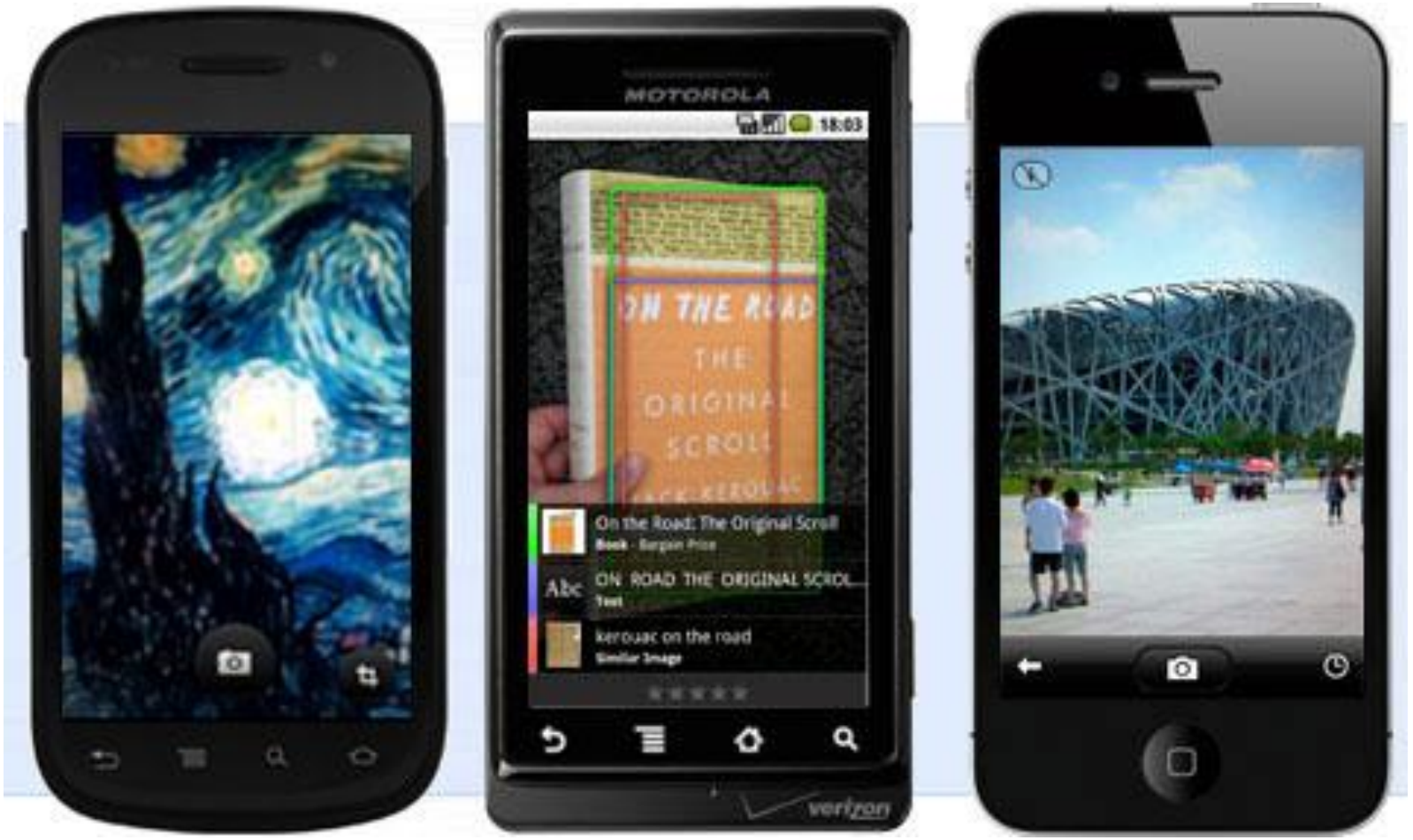


Higher branch factor works better (but slower)



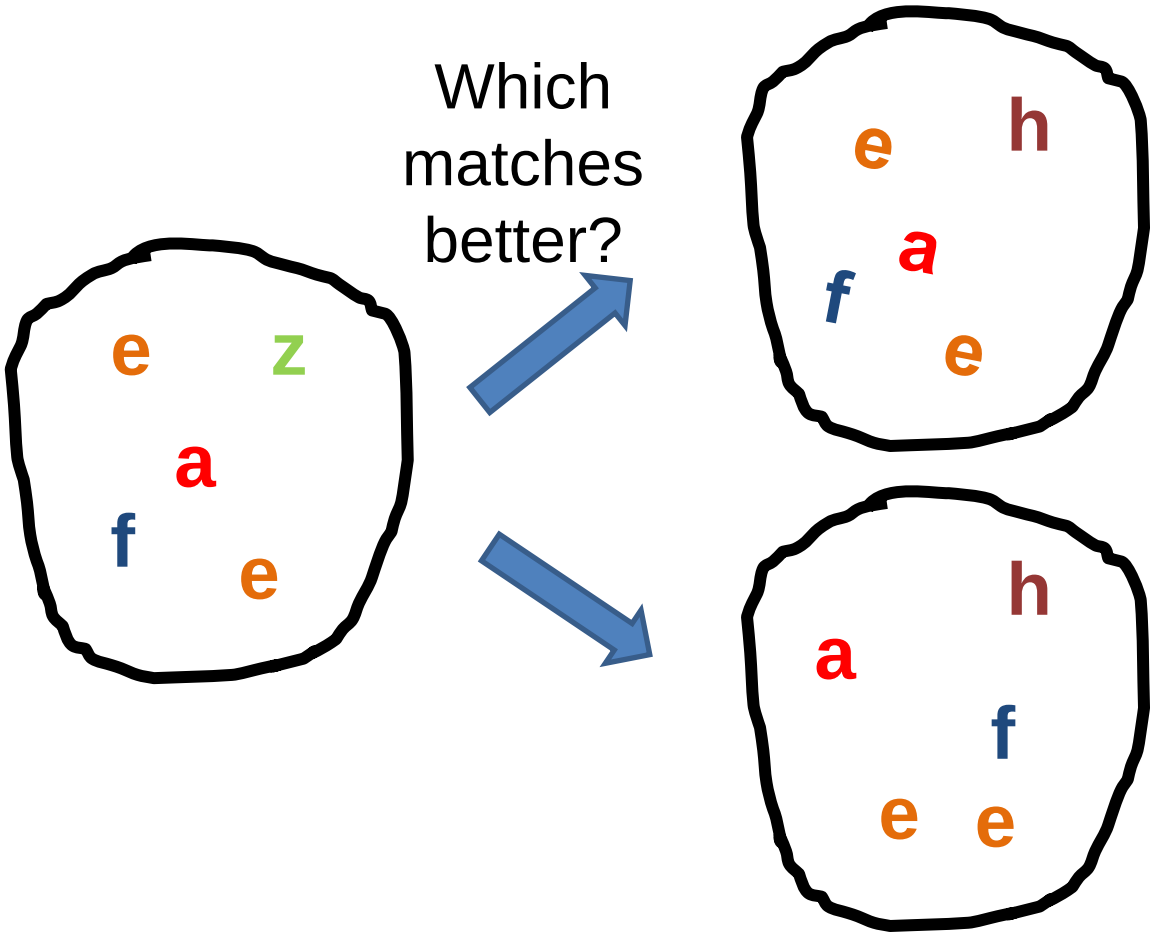
Application: Google Goggles

http://www.google.com/mobile/goggles/#text



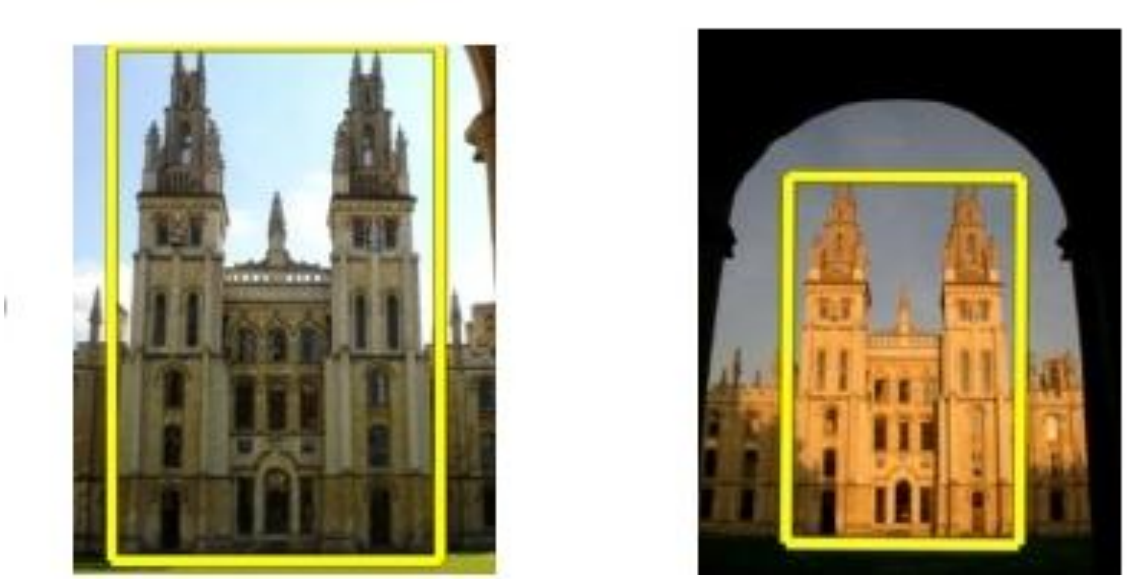
Can we be more accurate?

So far, we treat each image as containing a "bag of words", with no spatial information



Can we be more accurate?

So far, we treat each image as containing a "bag of words", with no spatial information



Real objects have consistent geometry

Final key idea: geometric verification

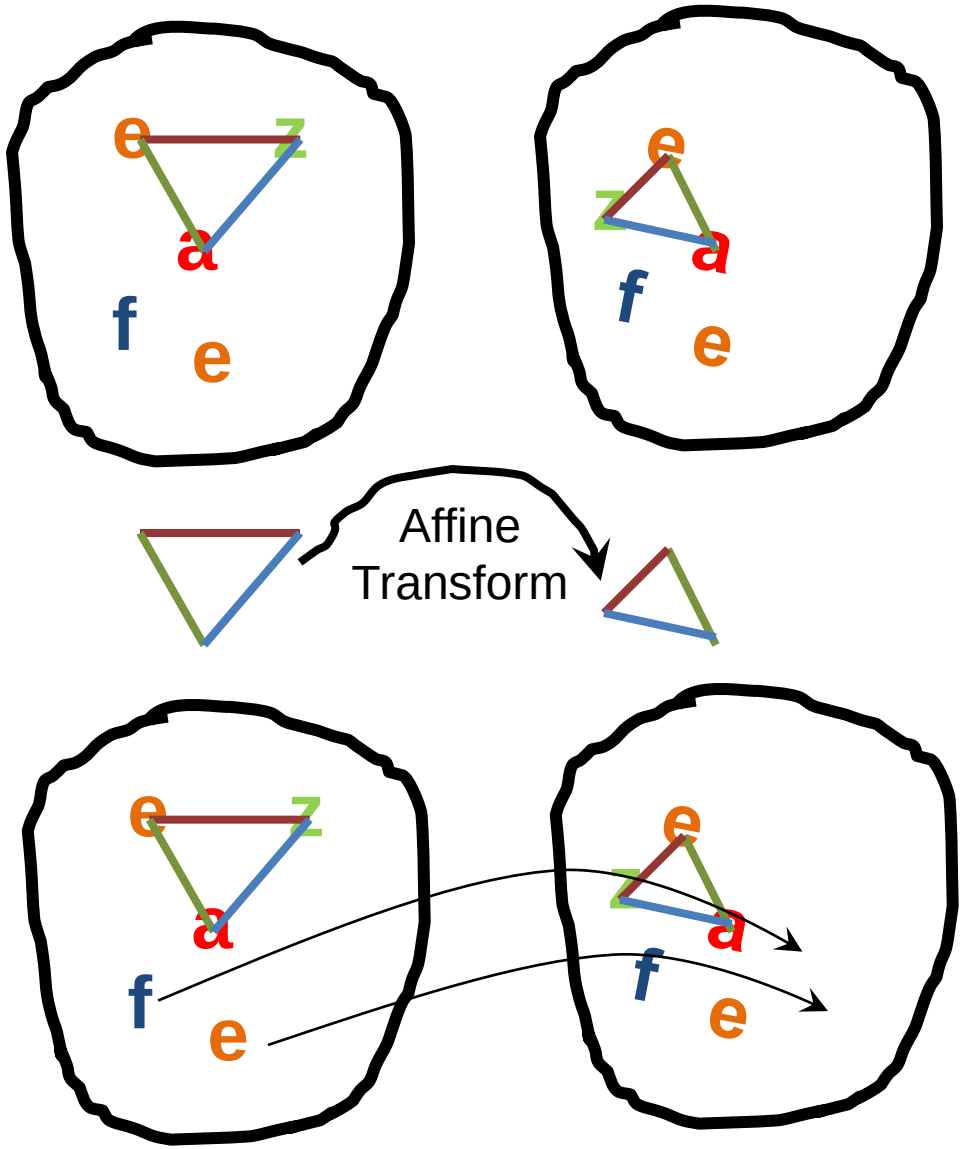
RANSAC for affine transform

Repeat N times:

Randomly choose 3 matching pairs

Estimate transformation

Predict remaining points and count “inliers”



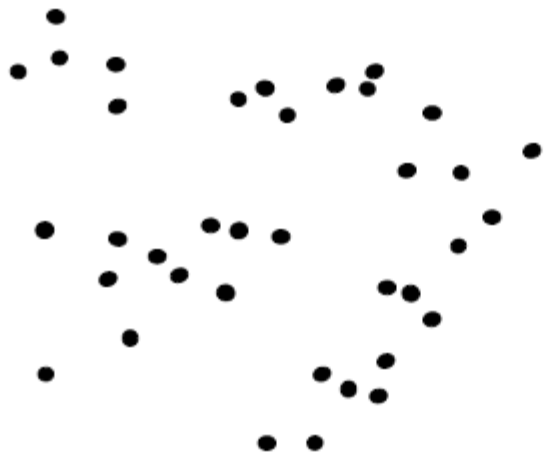
Video Google System

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

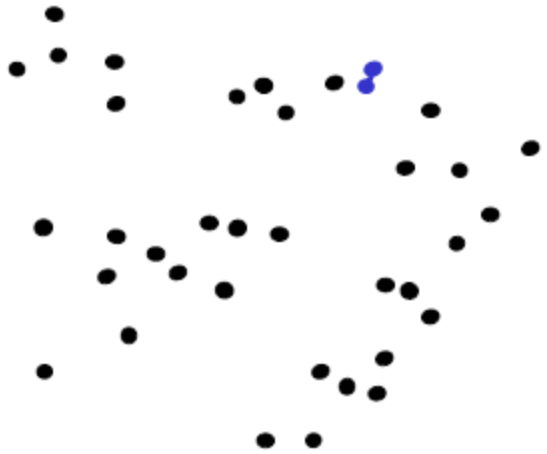
- Demo online at : http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

Agglomerative clustering



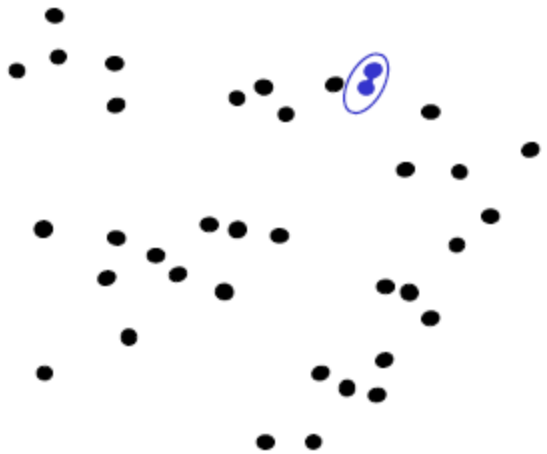
1. Say "Every point is its own cluster"

Agglomerative clustering



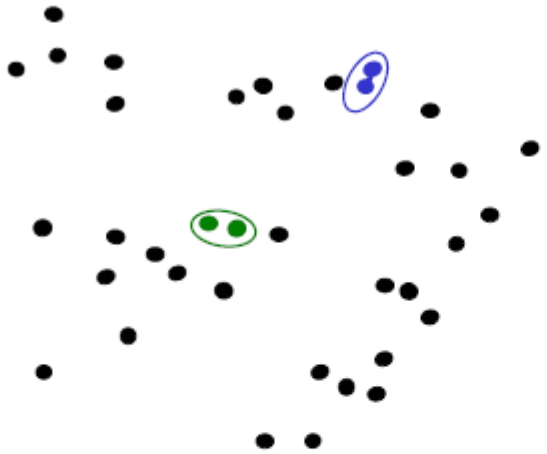
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Agglomerative clustering



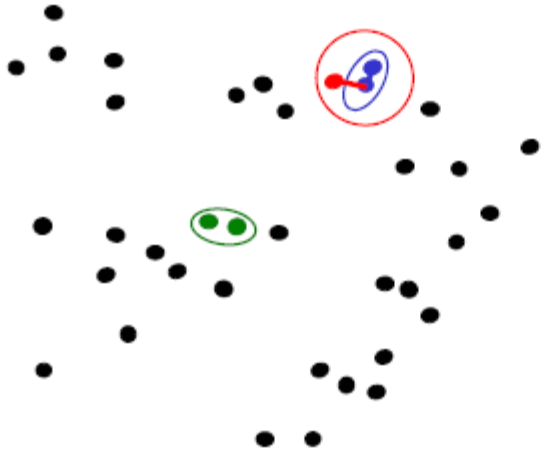
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Agglomerative clustering



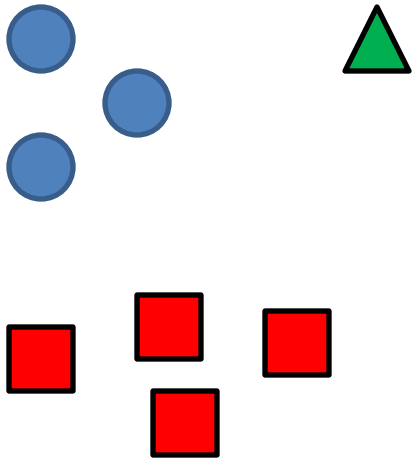
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering

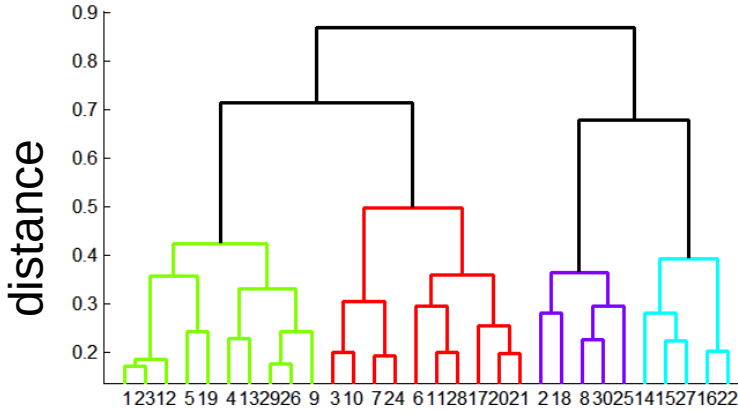


How to define cluster similarity?

- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?

- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges



Agglomerative clustering demo

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletH.html

Conclusions: Agglomerative Clustering

Good

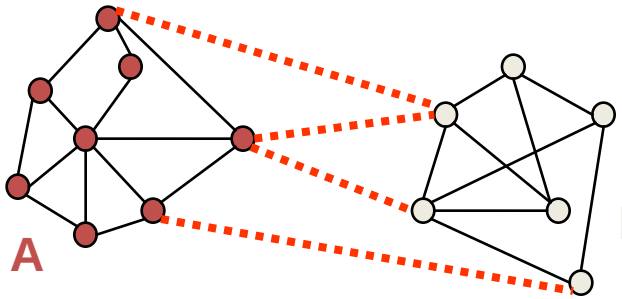
- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

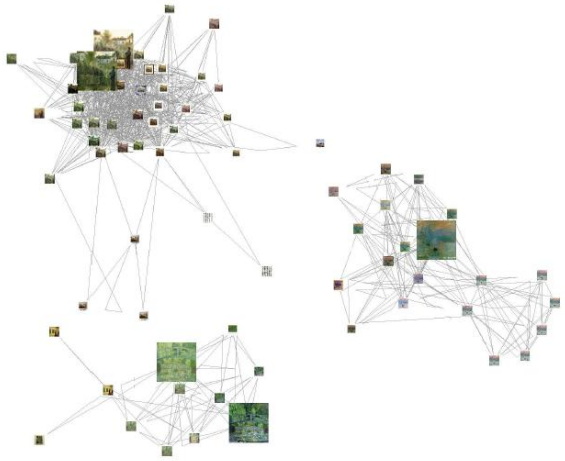
Bad

- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an “ultrametric” to get a meaningful hierarchy

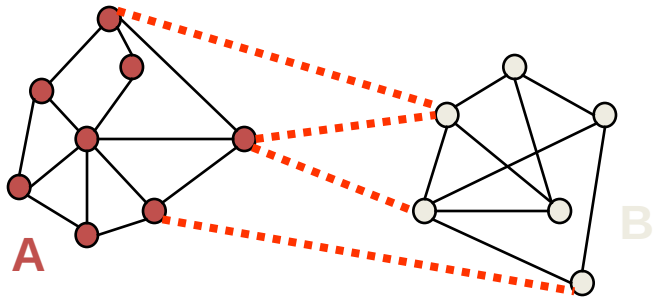
Spectral clustering

Group points based on links in a graph





Cuts in a graph



Normalized Cut

- the raw cut cost encouraging splitting out just one node
- fix by normalizing for size of segments

$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

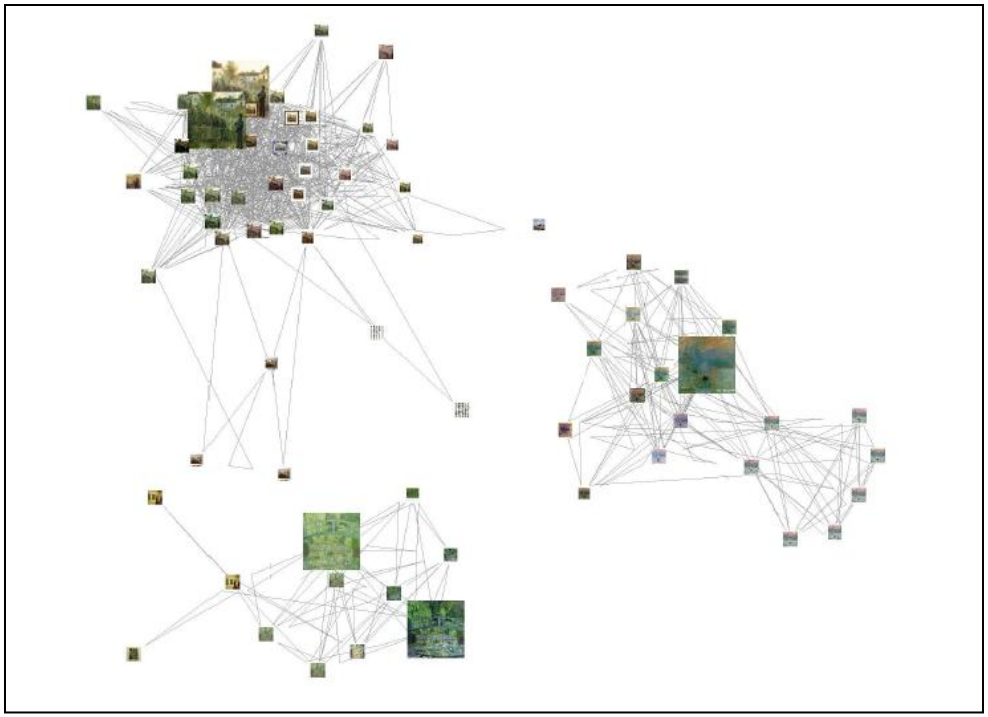
- volume(A) = sum of costs of all edges that touch A

Source: Seitz

Normalized cuts for segmentation

Visual PageRank

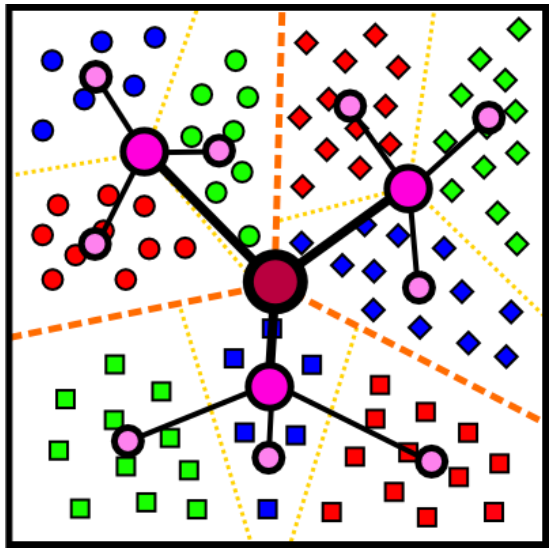
- Determining importance by random walk
  - What's the probability that you will randomly walk to a given node?
    - Create adjacency matrix based on visual similarity
    - Edge weights determine probability of transition



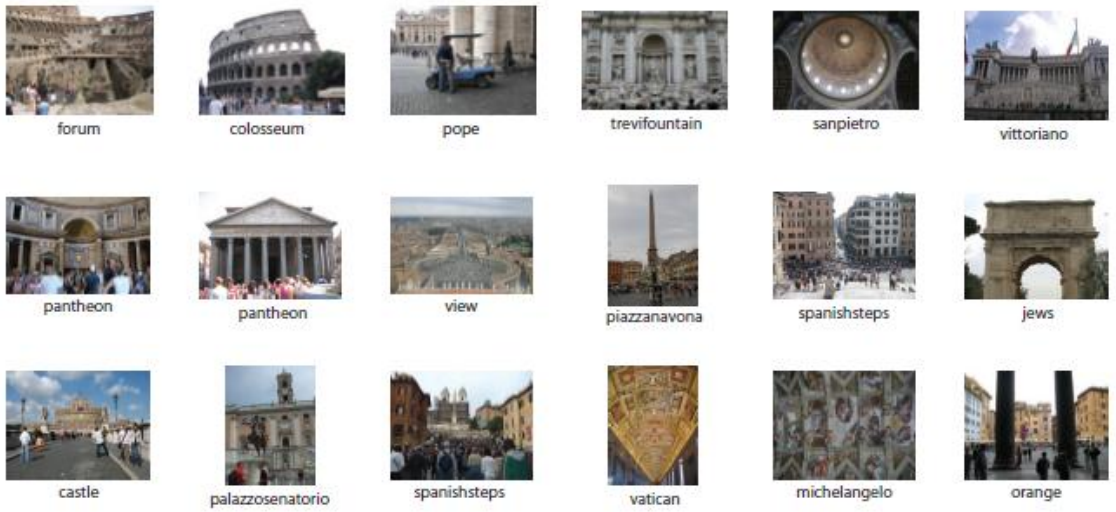
Jing Baluja 2008

Which algorithm to use?

- Quantization/Summarization: K-means
  - Aims to preserve variance of original data
  - Can easily assign new point to a cluster



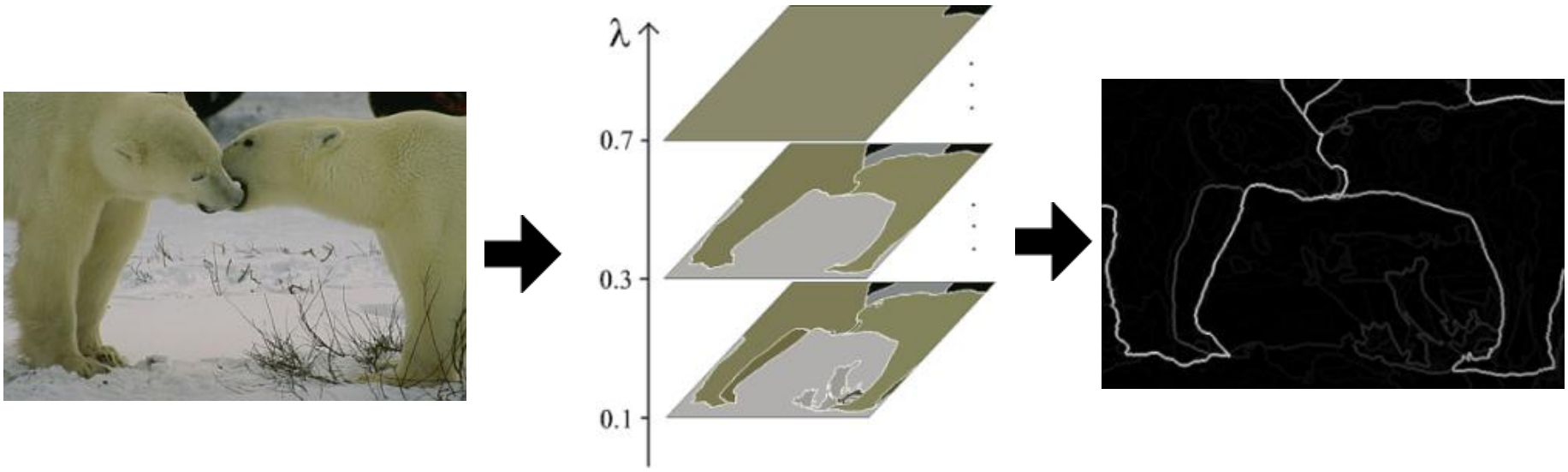
Quantization for computing histograms



Summary of 20,000 photos of Rome using "greedy k-means"
http://grail.cs.washington.edu/projects/canonview/

Which algorithm to use?

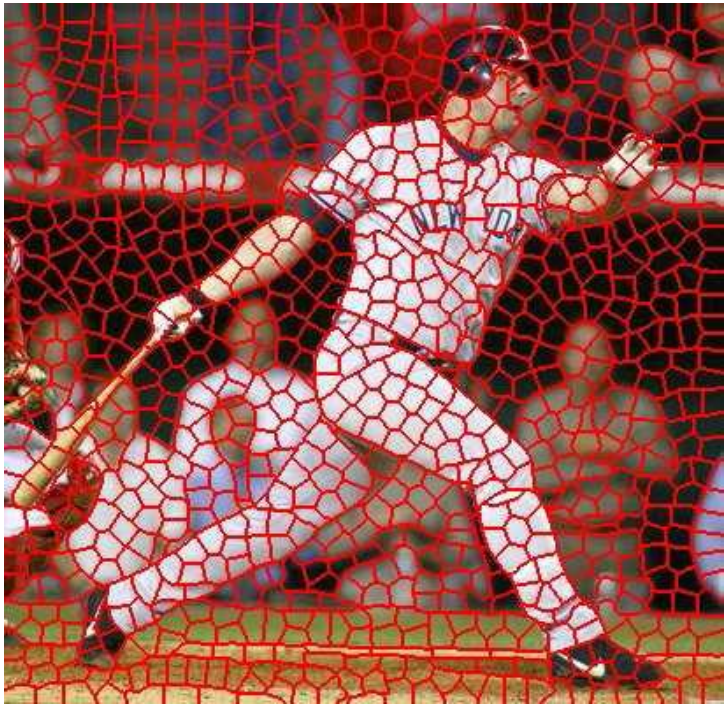
- Image segmentation: agglomerative clustering
  - More flexible with distance measures (e.g., can be based on boundary prediction)
  - Adapts better to specific data
  - Hierarchy can be useful



http://www.cs.berkeley.edu/~arbelaez/UCM.html

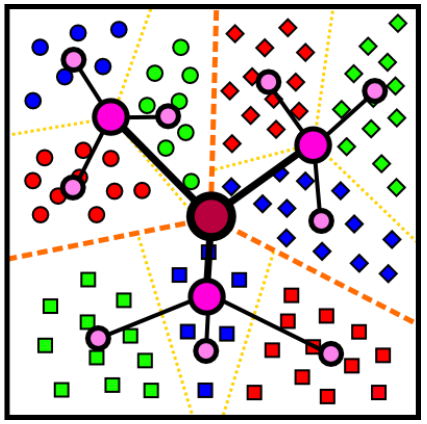
Which algorithm to use?

- Image segmentation: spectral clustering
  - Can provide more regular regions
  - Spectral methods also used to propagate global cues (e.g., Global pB)



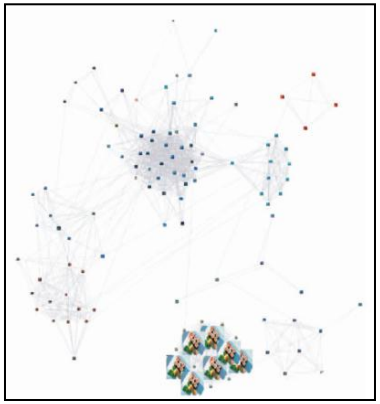
Things to remember

- K-means useful for summarization, building dictionaries of patches, general clustering
  - Fast object retrieval using visual words and inverse index table



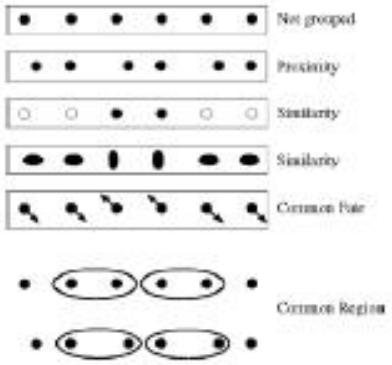
- Agglomerative clustering useful for segmentation, general clustering

- Spectral clustering useful for determining relevance, summarization, segmentation

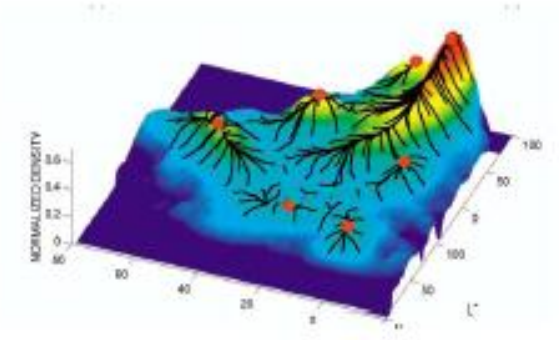


Next class

- Gestalt grouping



- Image segmentation
  - Mean-shift segmentation



  - Watershed segmentation