Clustering with Applications to Fast Object Search

Computer Vision
CS 543/ECE 549
University of Illinois

Kevin Shih

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

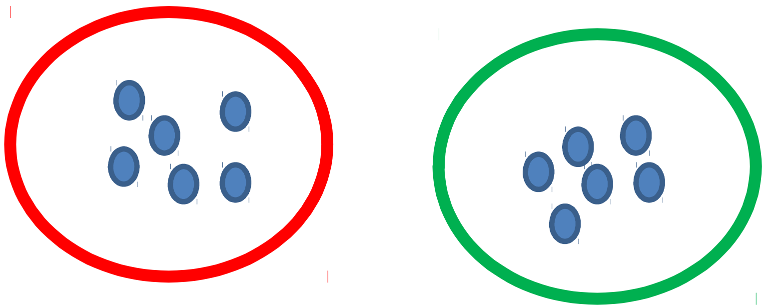
- EM: soft clustering, or parameter estimation with hidden data

Today’s class

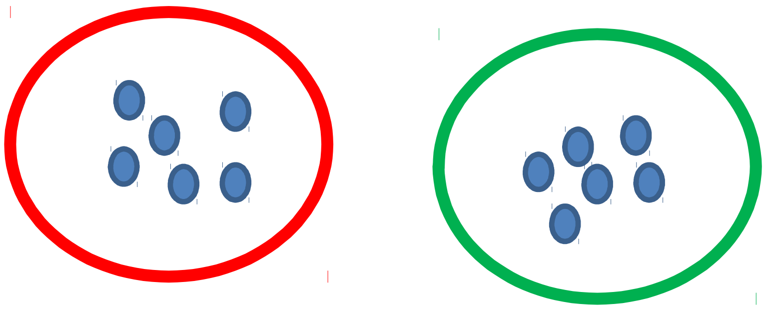
- Clustering algorithms
  - K-means
    - Application to fast object search
  - Hierarchical clustering
    - Bottom-up agglomerative clustering
    - Top-down divisive clustering
  - Spectral Clustering

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

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



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



Key Questions:

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

2) How do we determine the grouping from pairwise similarities?

Why do we cluster?

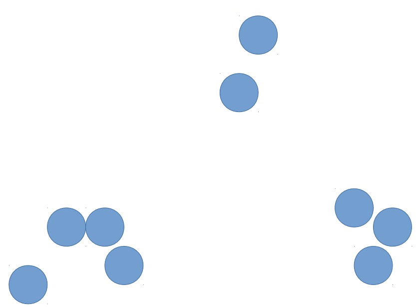
- **Summarizing data**
  - Visualization
  - Patch-based compression
- **Counting**
  - Represent a large continuous vector with the cluster number
  - Histograms of texture, color, SIFT vectors
  - Otherwise impossible with continuous values
- **Segmentation**
  - Separate the image into different regions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

K-means algorithm

$$\underset{S,\mu_{i,i=1..K}}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} ||x - \mu_i||^2$$

We wish to partition the data into K sets S = {S_1, S_2, … S_K} with corresponding centers μ_i

Partition such that variance in each partition is as low as possible

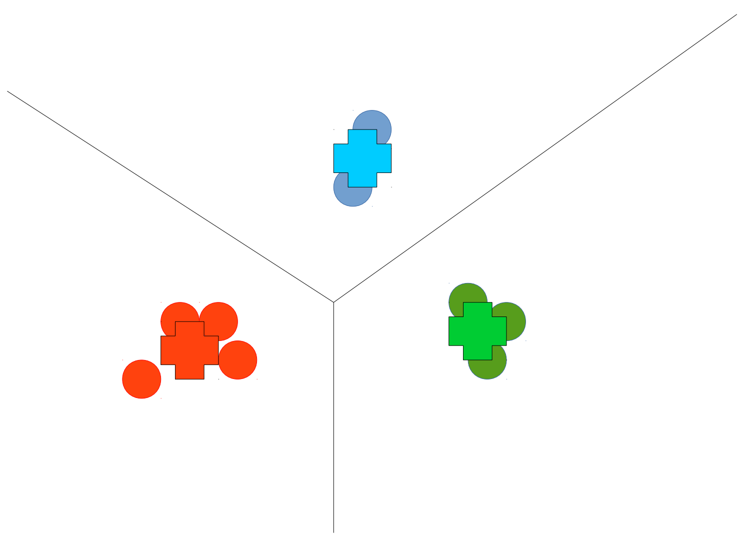


K-means algorithm

$$\underset{S,\mu_{i,i=1..K}}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} ||x - \mu_i||^2$$

We wish to partition the data into K sets S = {S_1, S_2, ... S_K} with corresponding centers μ_i

Partition such that variance in each partition is as low as possible



K-means algorithm

1. Randomly
select K centers

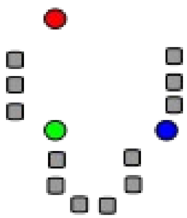


Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means algorithm

1. Randomly select K centers

2. Assign each point to nearest center

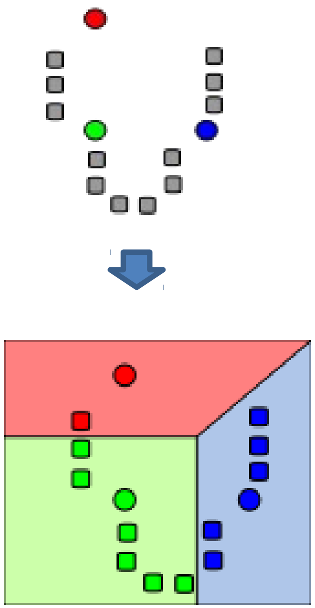


Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means algorithm

1. Randomly select K centers

2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

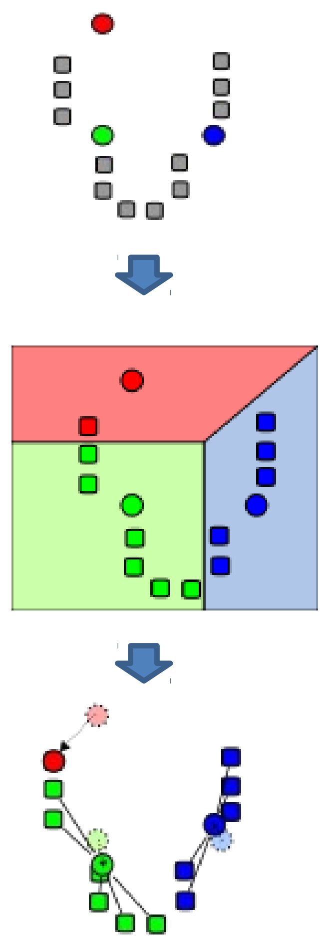


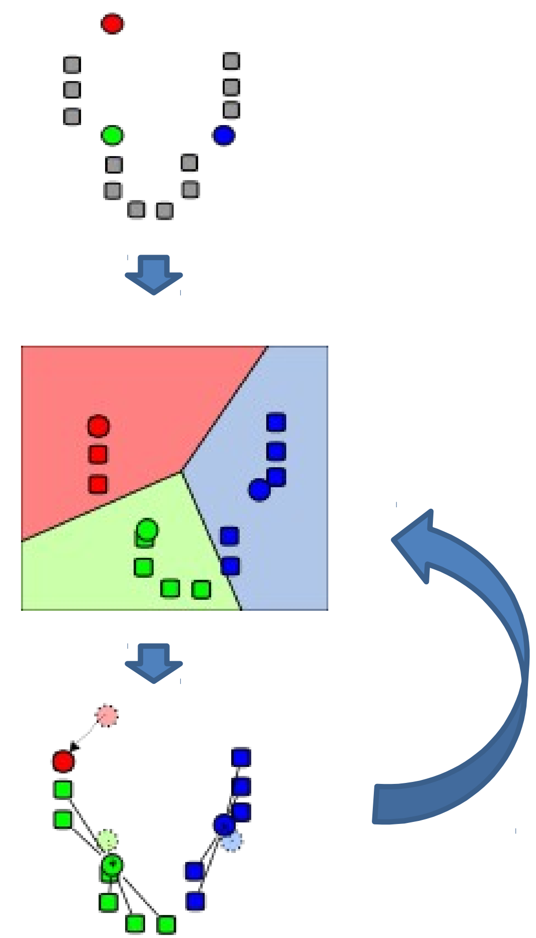
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

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 algorithm

1. Initialize K centers μ_i (usually randomly)

2. Assign each point x to its nearest center:

$$S^t = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} ||x - \mu_i||^2$$

3. Update cluster centers as the mean of its members

$$\mu^t = \underset{\mu_{i,i=1..K}}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} ||x - \mu_i||^2$$

4. Repeat 2-3 until convergence (t = t+1)

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/

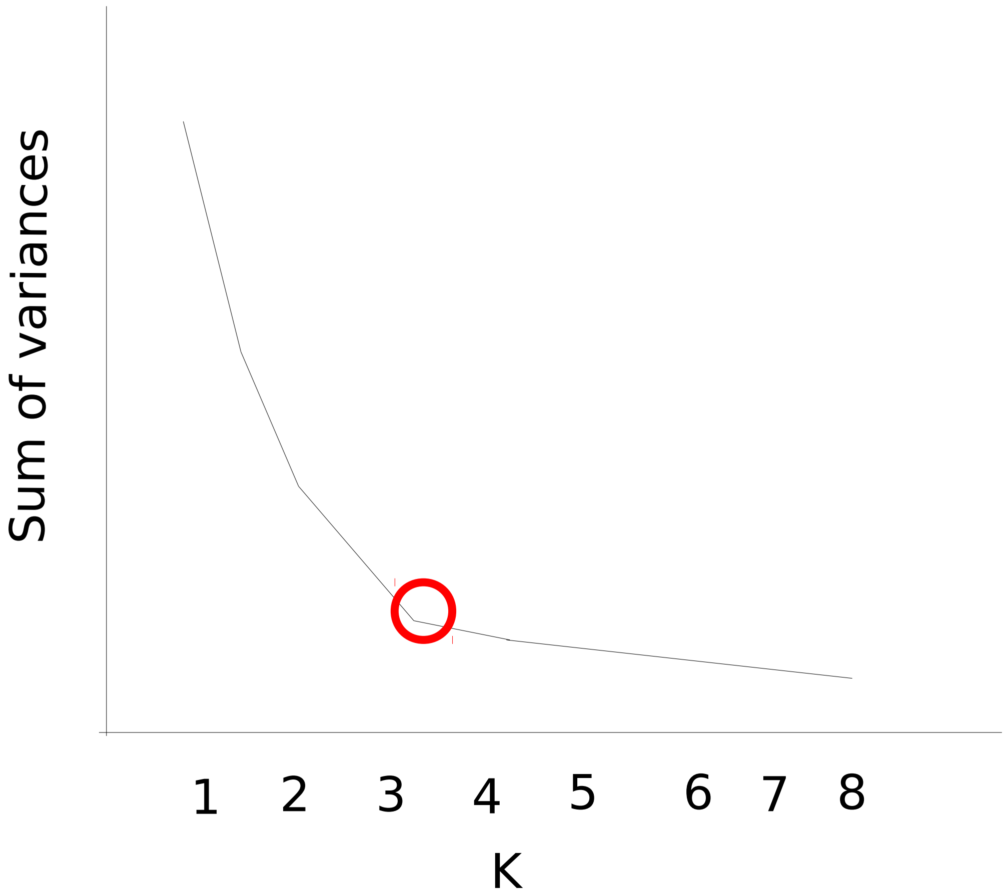
Kmeans: Matlab code

```
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 centers
  - Greedily choose K points to minimize residual
- Distance measures
  - Traditionally Euclidean, could be others
- Optimization
  - Converges to a local minimum
  - May want to perform multiple restarts (re-initialize and try again)

How to choose the number of clusters?



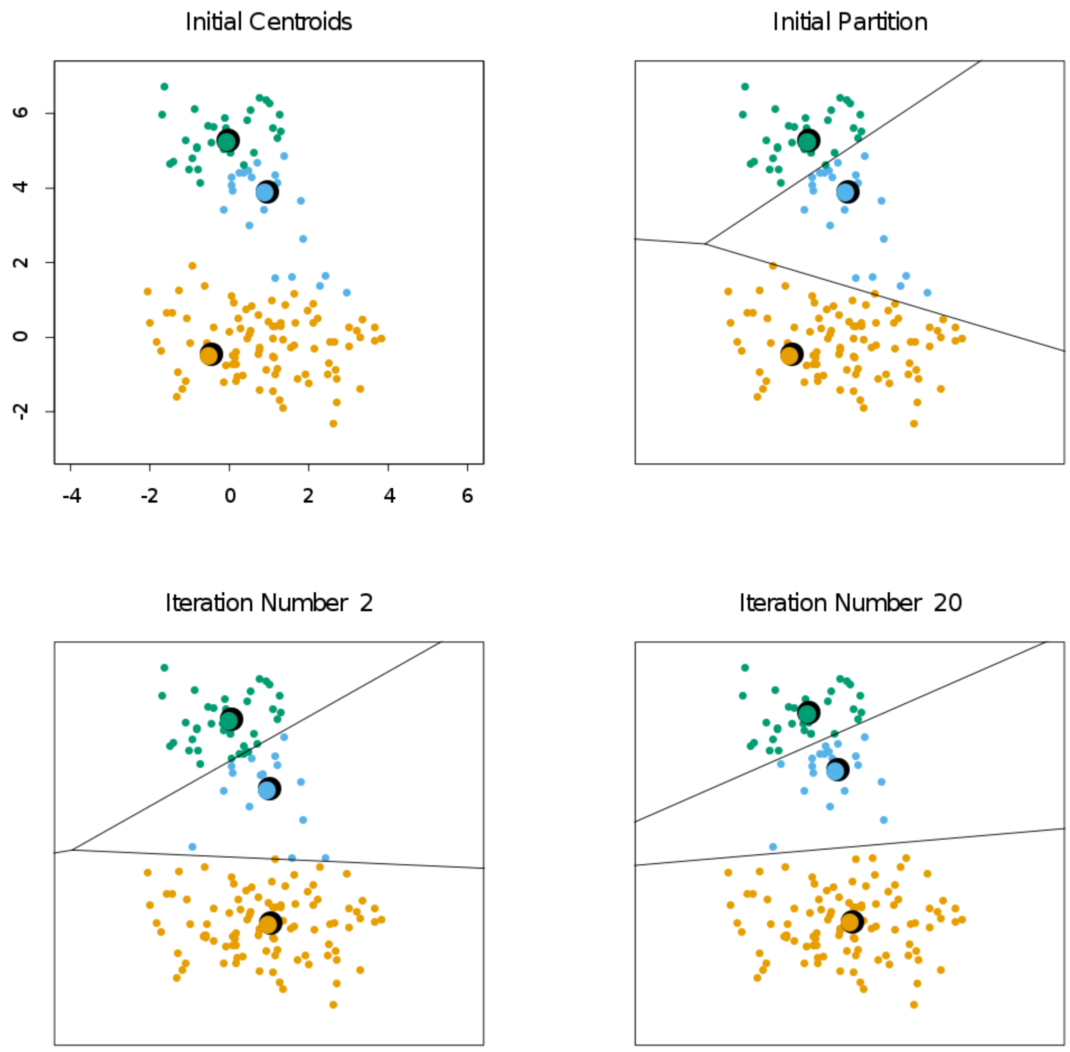
- Elbow method
- Stop adding clusters when improvement is small

How to choose the number of clusters?

- Validation set
  - Try different number of clusters and look at performance

K-means space partitioning

- Creates a voronoi partitioning
  - Generally convex shaped partitions



Source: The Elements of Statistical Learning, Hastie et al.

Conclusions: K-means

Good

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

Bad

- Sensitive to starting locations
- Need to choose K
- All clusters have the same parameters (e.g., distance measure is non-adaptive)

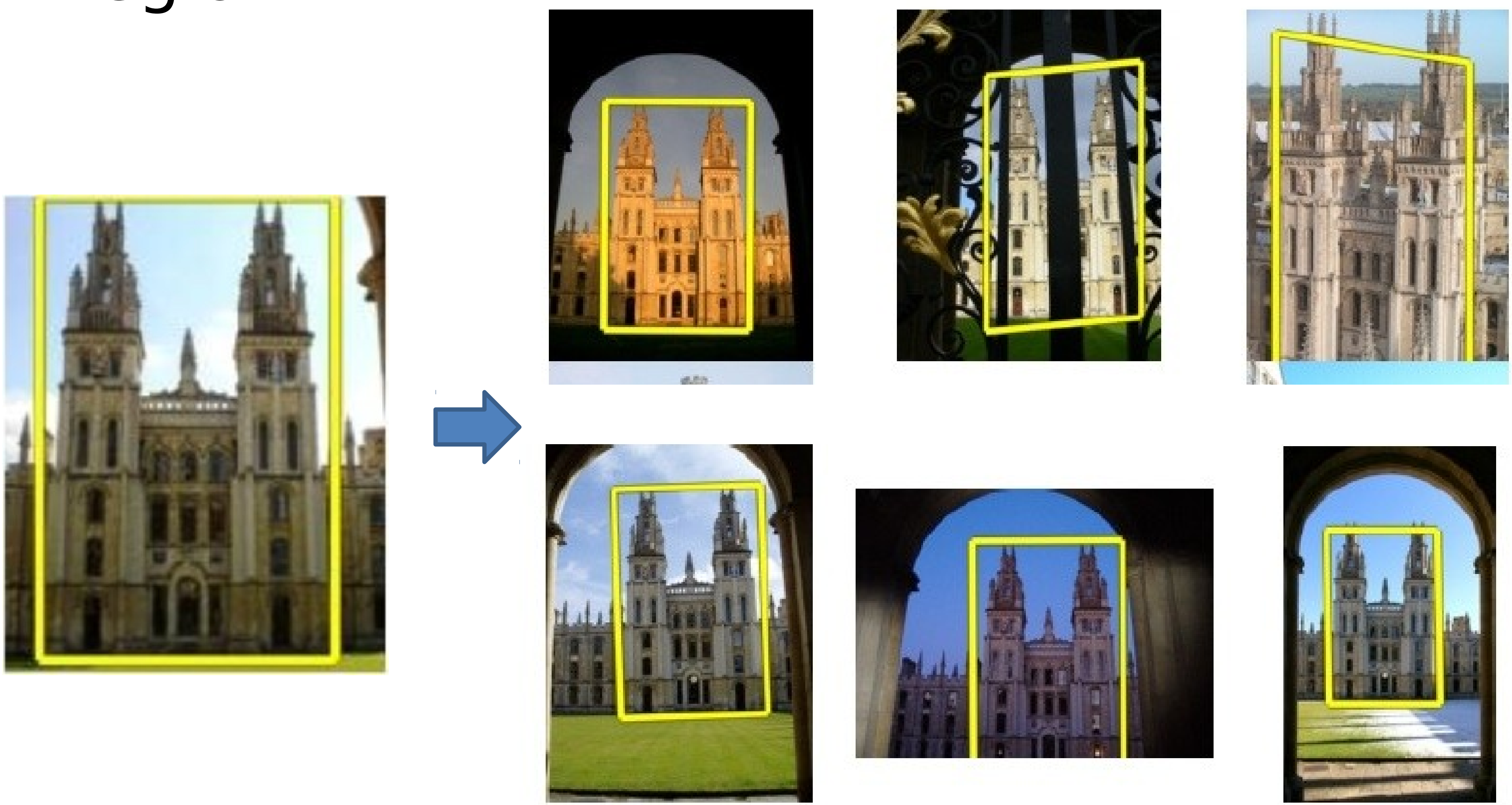
K-medoids

- 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
  - Clustering hue values
    - Average of red and green would be yellow-ish
- Less sensitive to outliers

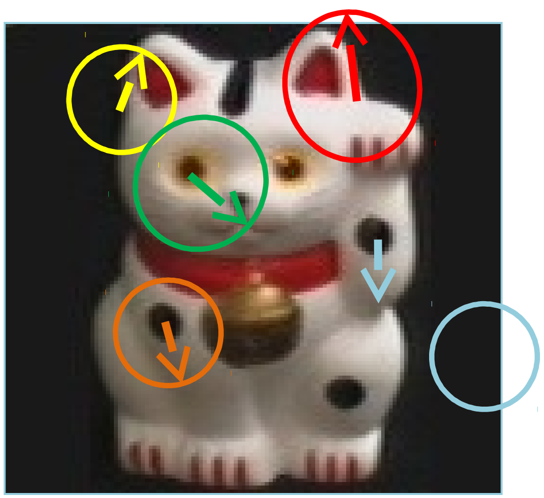
Application of K-means

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

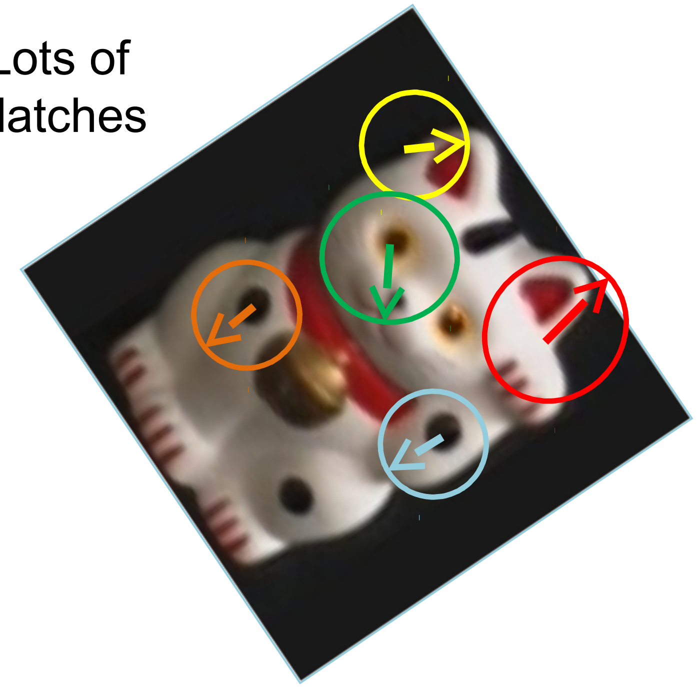


Simple idea

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



Lots of Matches

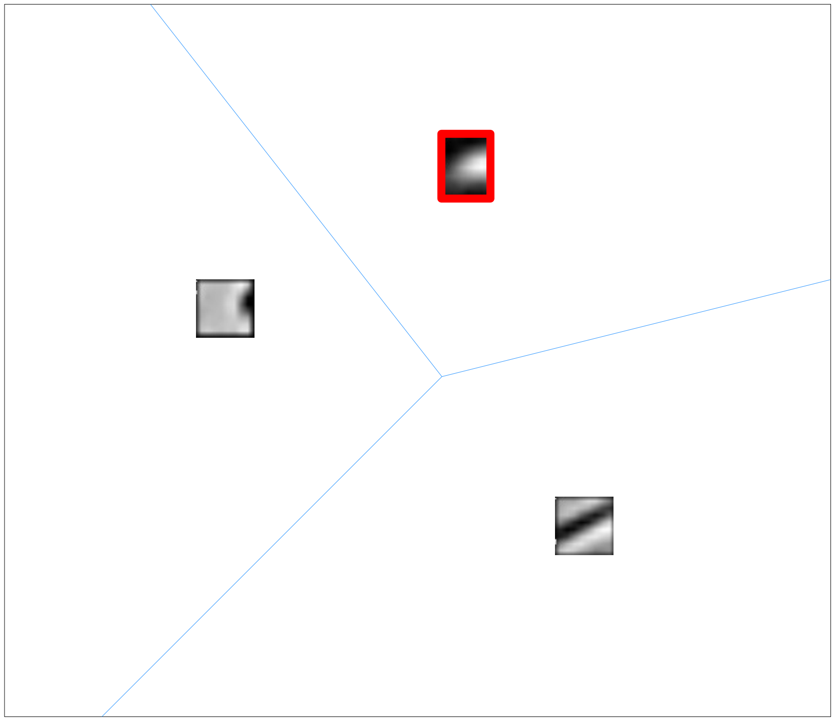


Few or No Matches

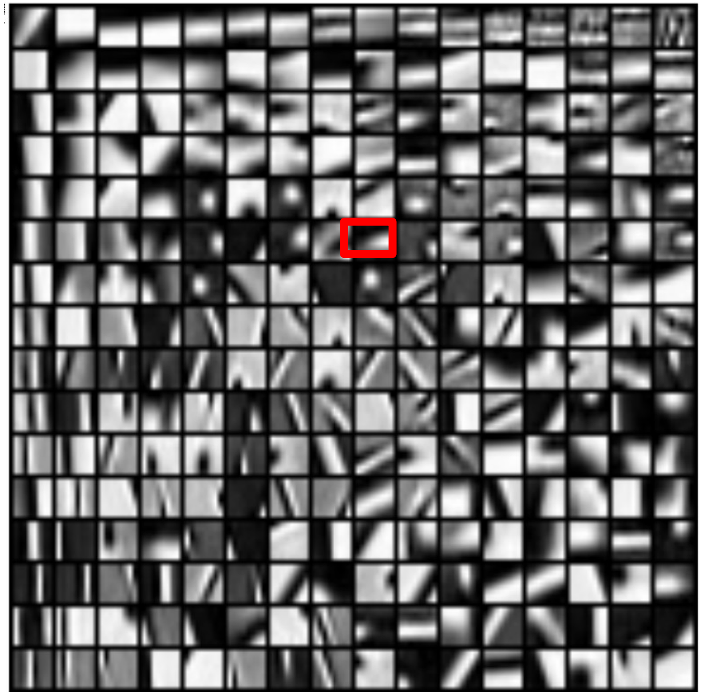
But this will be really, really slow!

Bag of Visual Words

- Cluster the keypoint descriptors into a managable vocabulary size
- Assign each descriptor to a cluster number

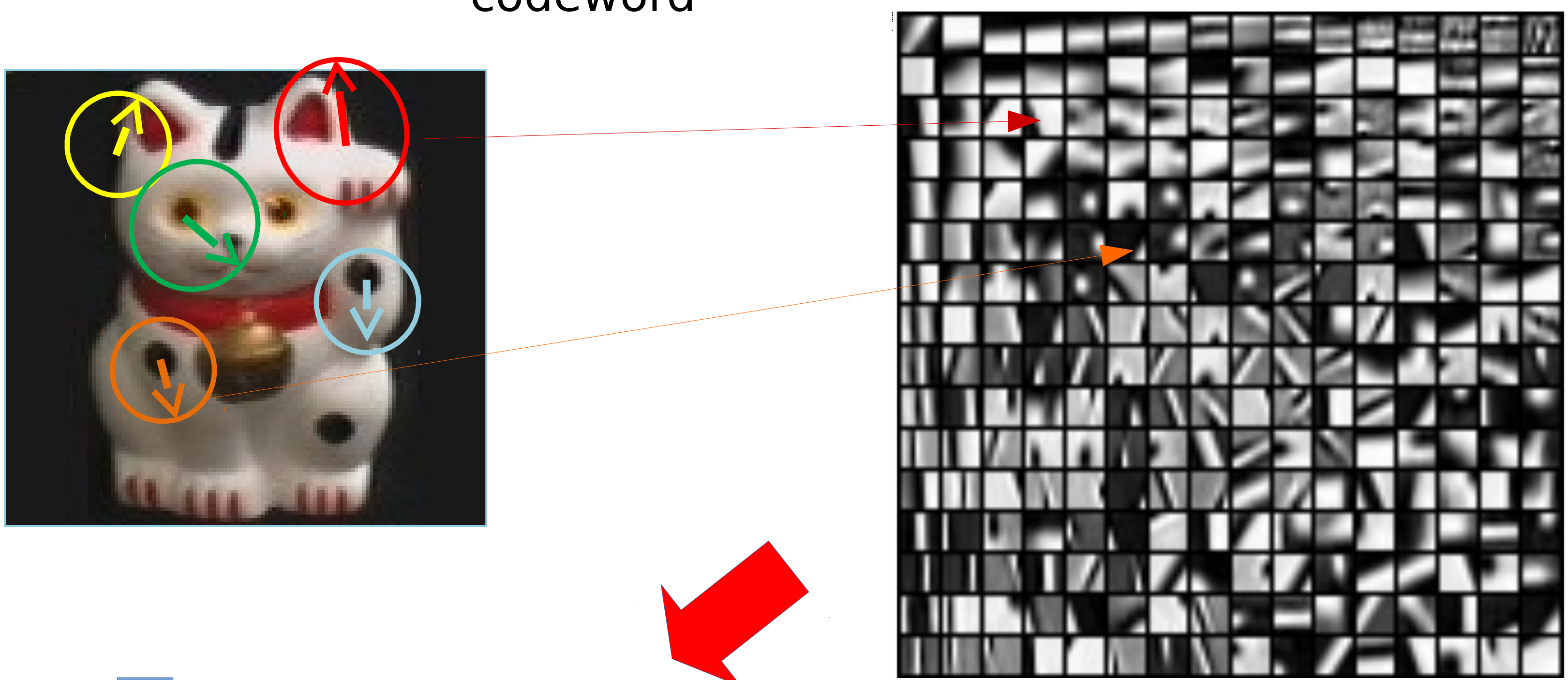


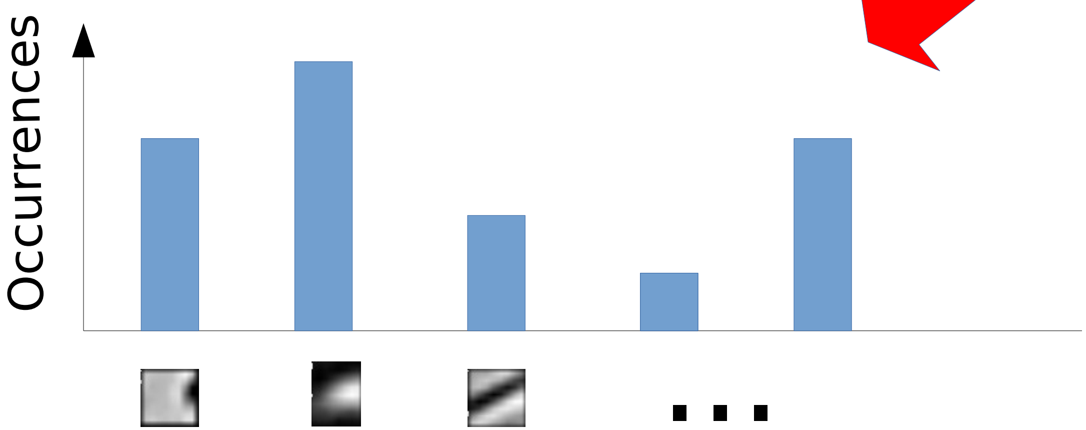
Codebook of cluster centers



Bag of Visual Words

Assign to nearest codeword

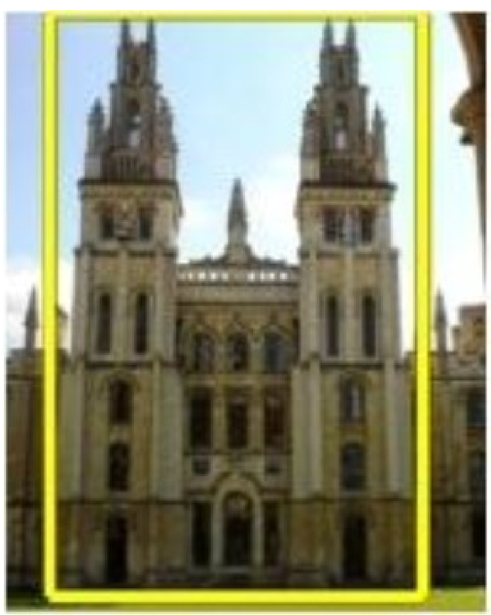
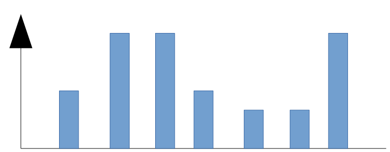
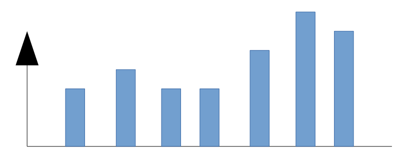


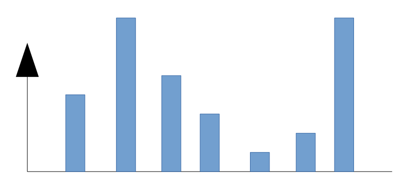


How many instances of each codeword appeared in this image?

Bag of Visual Words

- Each image is represented by a histogram of codeword frequencies
- Similar images should have similar histograms




Bag of Visual Words

- How to match?
- First normalize histogram vectors
  - Compute similarity using cosine distance, histogram intersection, inner product, etc.

Bag of Visual Words

- How to match?
- First normalize histogram vectors
  - Compute similarity using cosine distance, histogram intersection, inner product, etc.
- What if we're querying from a large dataset?
  - Pairwise comparisons will take forever

Bag of Visual Words

- How to match?
- First normalize histogram vectors
  - Compute similarity using cosine distance, histogram intersection, inner product, etc.
- What if we're querying from a large dataset?
  - Pairwise comparisons will take forever
- Are all codewords equally important?

Bag of Visual Words

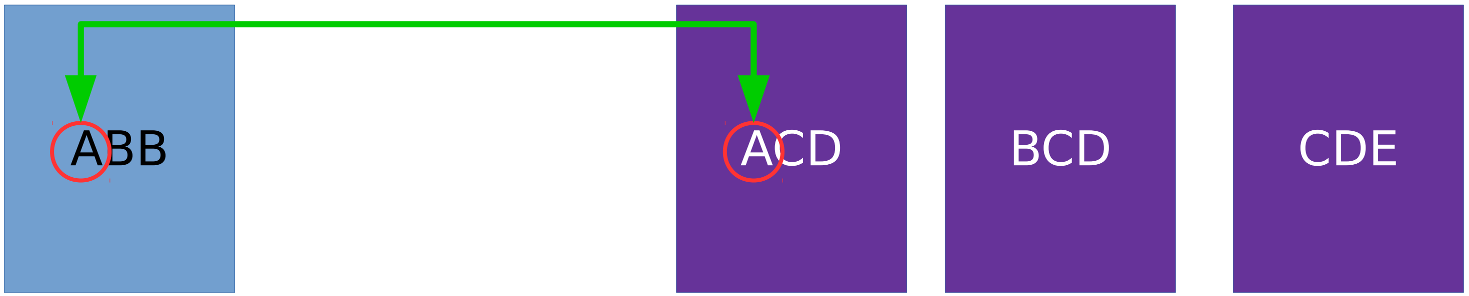
- Inverted Index
  - Analogous to index section of a book
  - Store mapping from codeword to images it appears in
- Fast inner product computation on large datasets:
  - Only operate on images containing relevant codewords

Bag of Visual Words

- Inverted Index
  - Analogous to index section of a book
  - Store mapping from codeword to images it appears in
- Fast inner product computation on large datasets:
  - Only operate on images containing relevant codewords

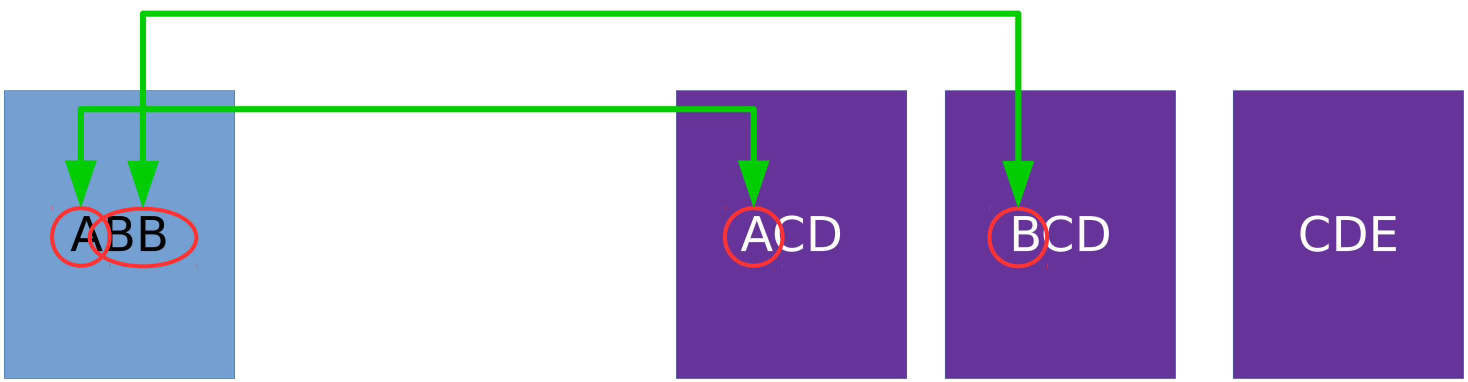
Bag of Visual Words

- Inverted Index
  - Analogous to index section of a book
  - Store mapping from codeword to images it appears in
- Fast inner product computation on large datasets:
  - Only operate on images containing relevant codewords



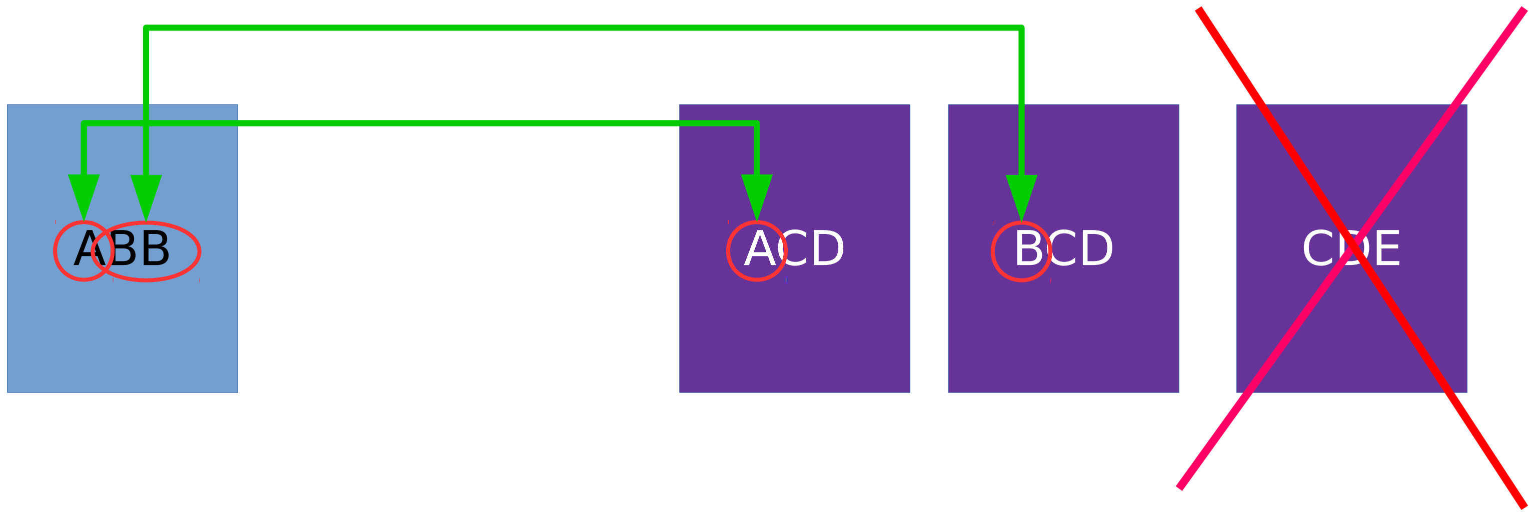
Bag of Visual Words

- Inverted Index
  - Analogous to index section of a book
  - Store mapping from codeword to images it appears in
- Fast inner product computation on large datasets:
  - Only operate on images containing relevant codewords



Bag of Visual Words

- Inverted Index
  - Analogous to index section of a book
  - Store mapping from codeword to images it appears in
- Fast inner product computation on large datasets:
  - Only operate on images containing relevant codewords



Bag of Visual Words

- Term Frequency-Inverse Document Frequency
  - The more documents/images the word appears in, the less informative it is

tf-idf: Term Frequency – Inverse Document Frequency

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

times word appears in document

documents

documents that contain the word

words in document

Bag of Visual Words

- How many codewords for a retrieval task?

Bag of Visual Words

- How many codewords for a retrieval task?
  - Fixed dataset, don't worry about overfitting
  - Generally, the more the better
    - Codewords better approximate data

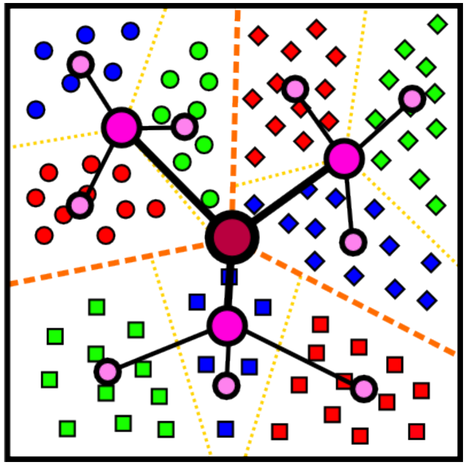
Bag of Visual Words

- How many codewords for a retrieval task?
  - Fixed dataset, don't worry about overfitting
  - Generally, the more the better
    - Codewords better approximate data
- Computational cost of too many codewords?

Bag of Visual Words

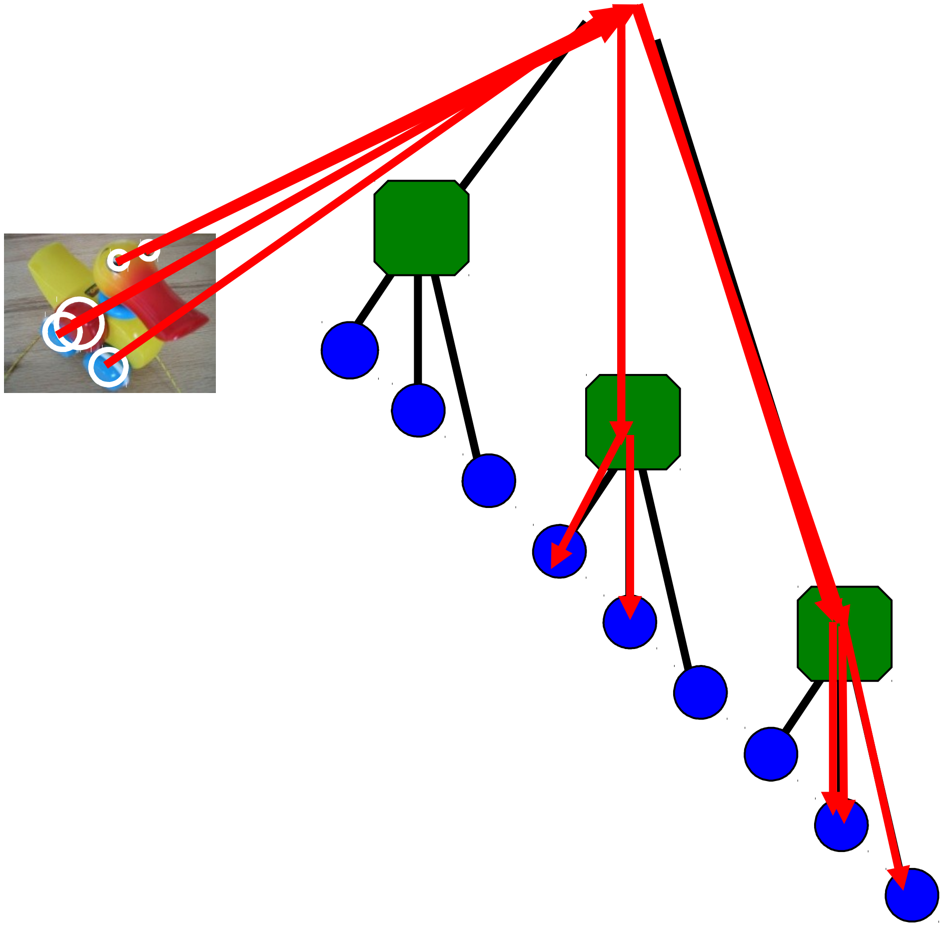
- How many codewords for a retrieval task?
  - Fixed dataset, don't worry about overfitting
  - Generally, the more the better
    - Codewords better approximate data
- Computational cost of too many codewords?
  - C codewords and F unmapped features vectors
  - C*F distance calculations to encode
  - Can we do better?

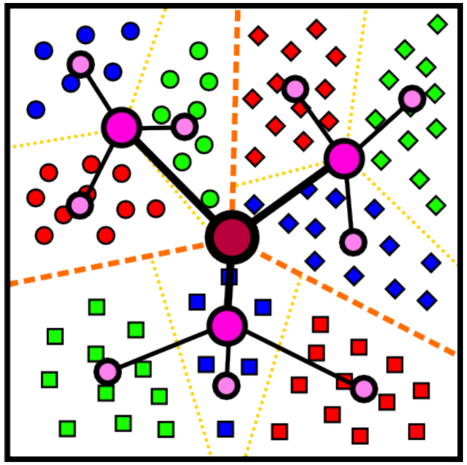
Bag of Visual Words



- Hierarchical K-means
- Iteratively partition space into smaller voronoi partitions
- C codewords, F unmapped features, branching factor K
  - $(K*\log_K C)*F$ distance calcualtions to encode

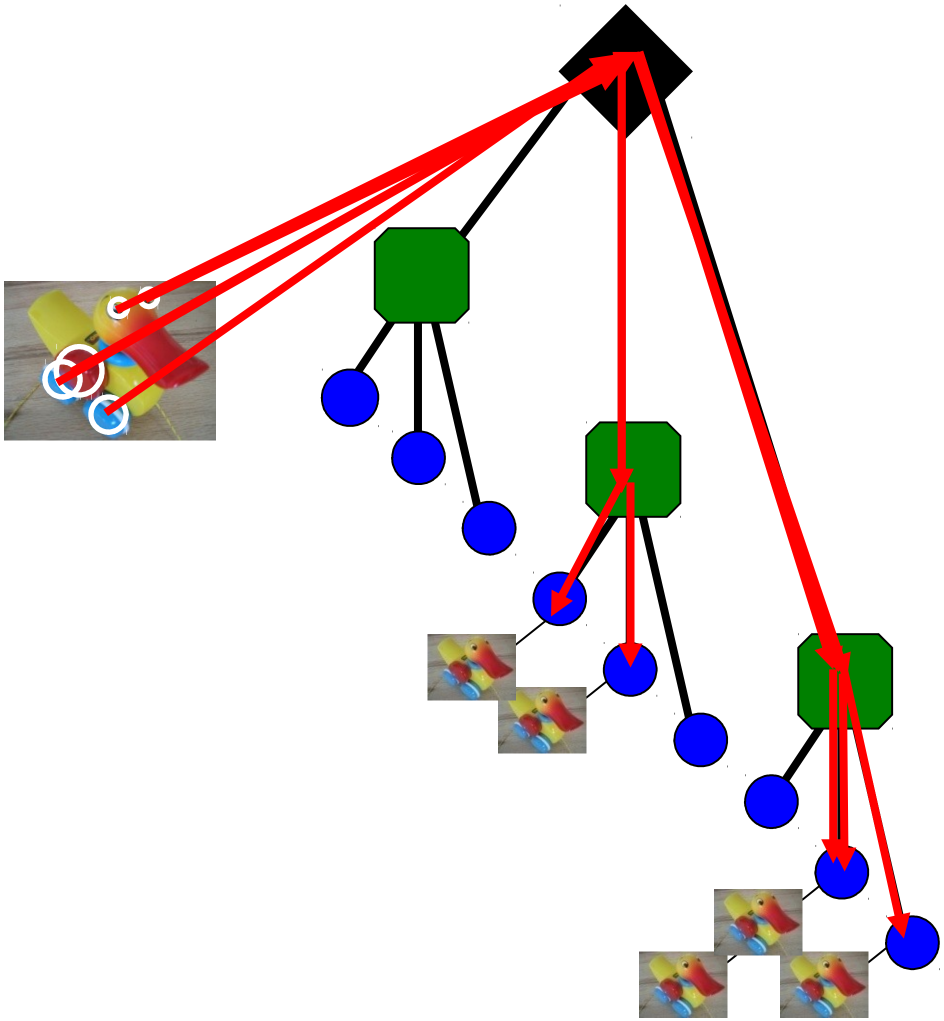
Bag of Visual Words

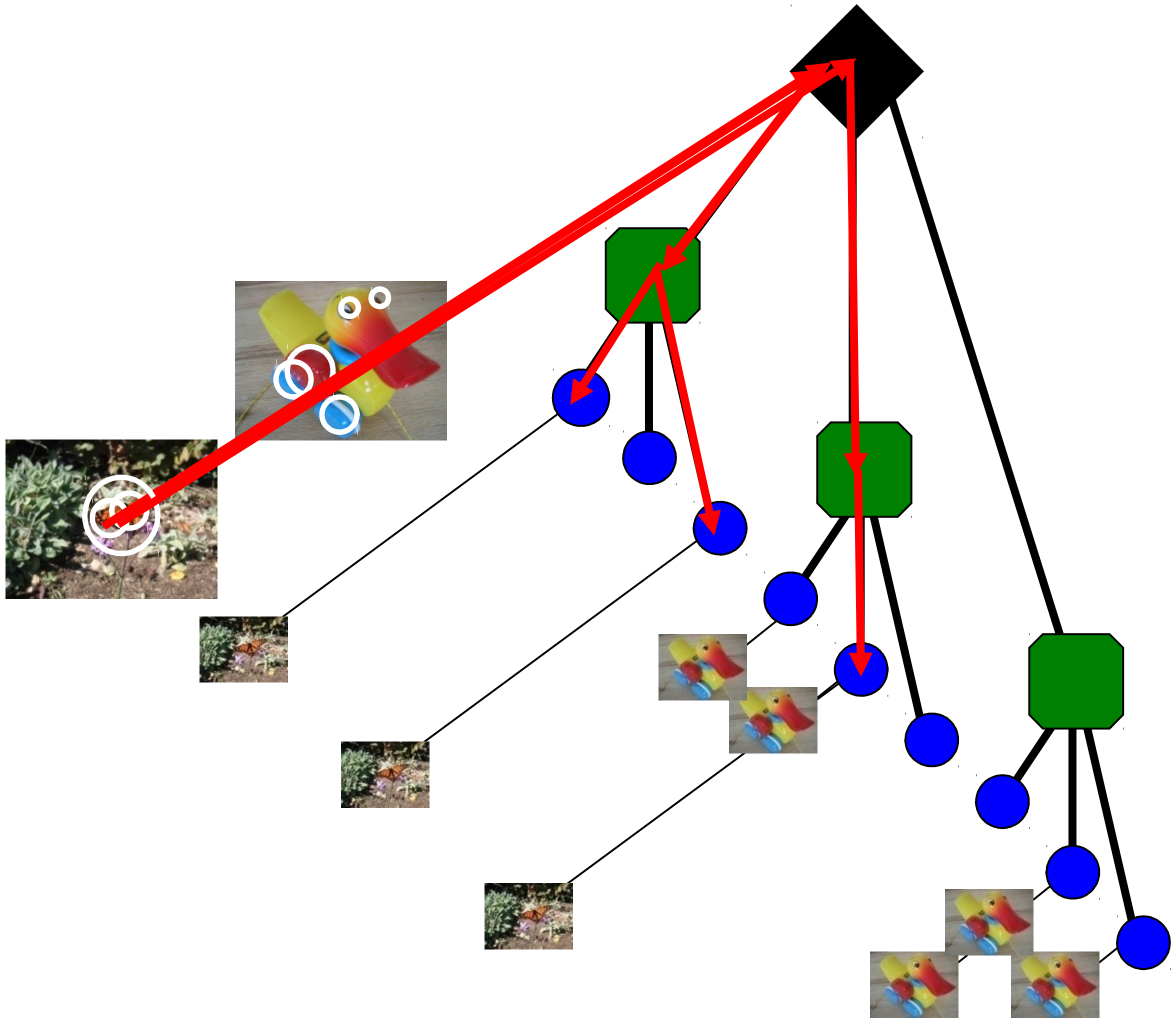


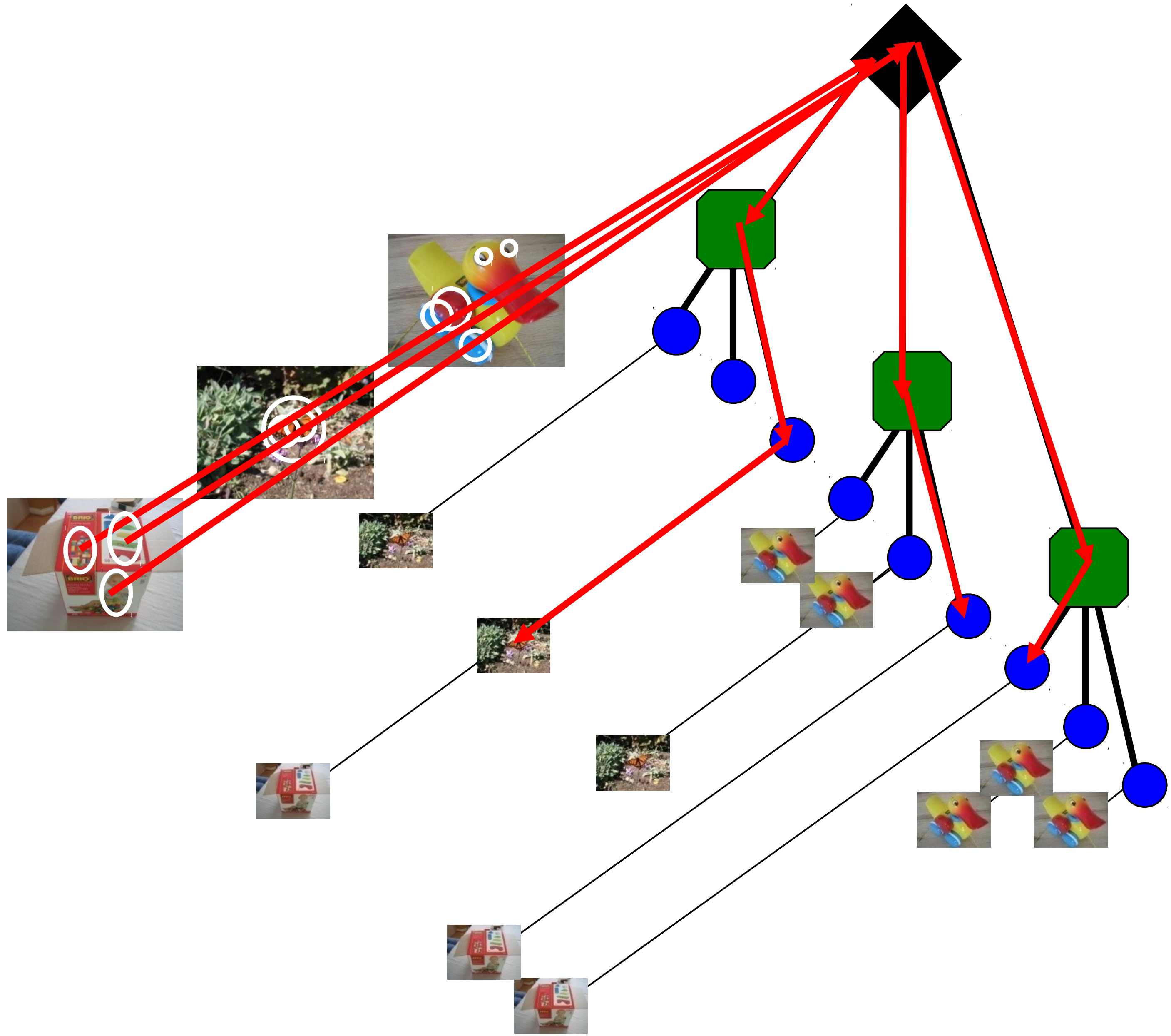


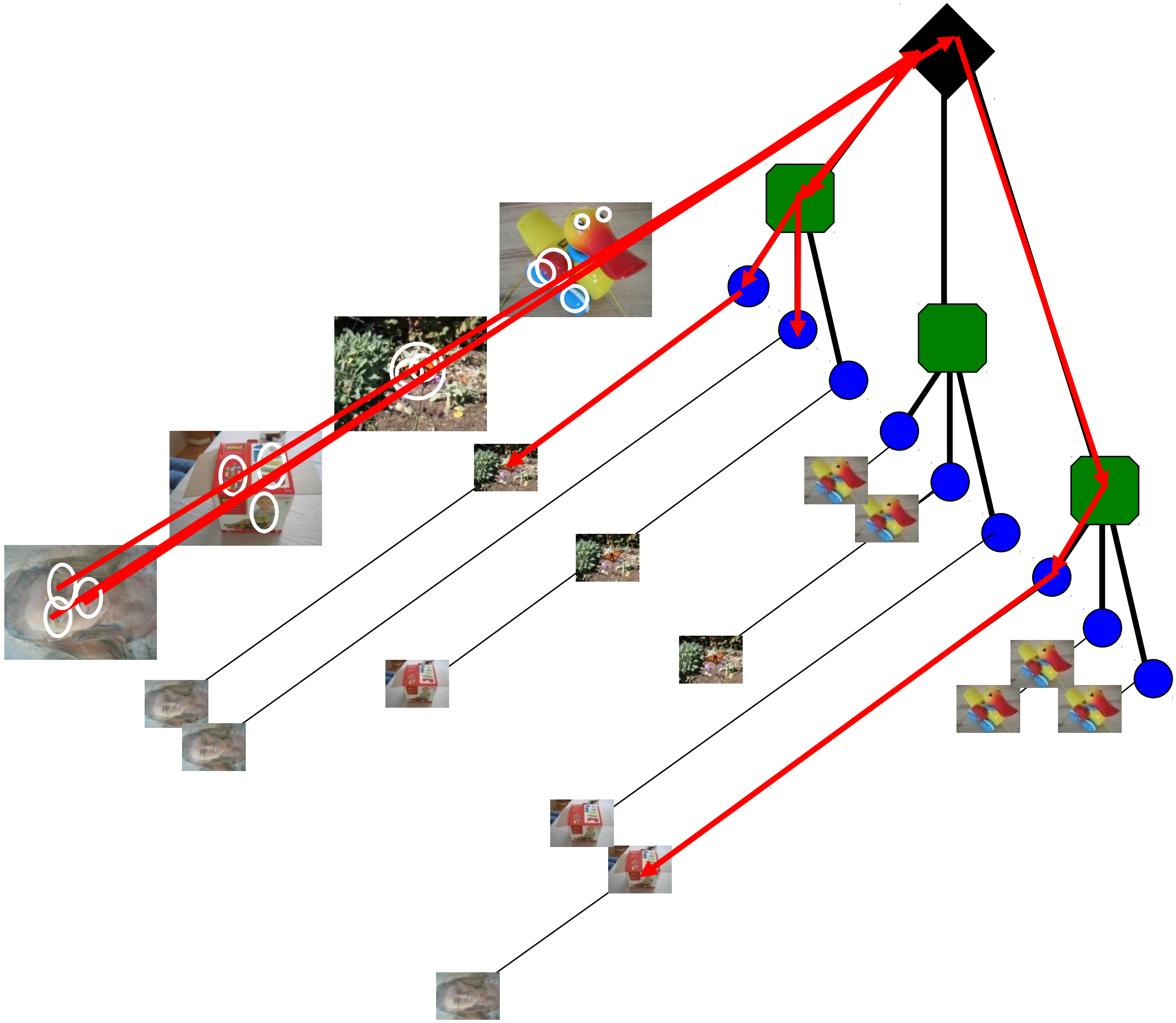
110,000,000 Images in 5.8 Seconds

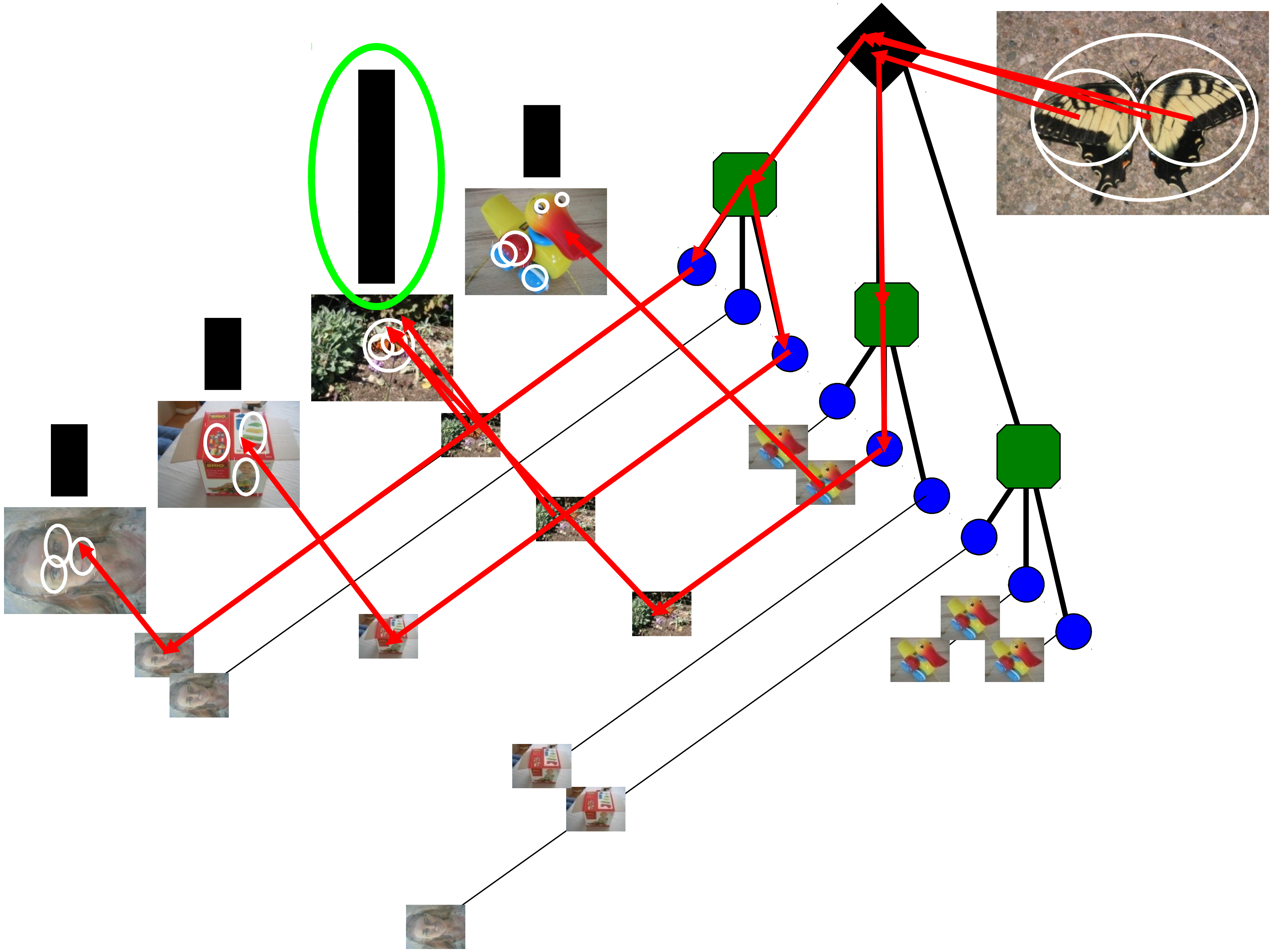
This slide and following by David Nister



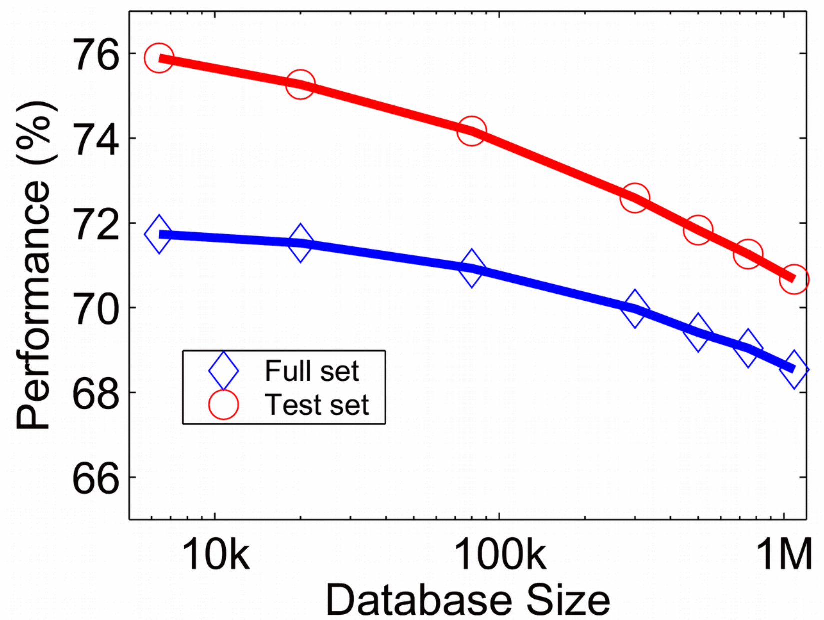








Performance

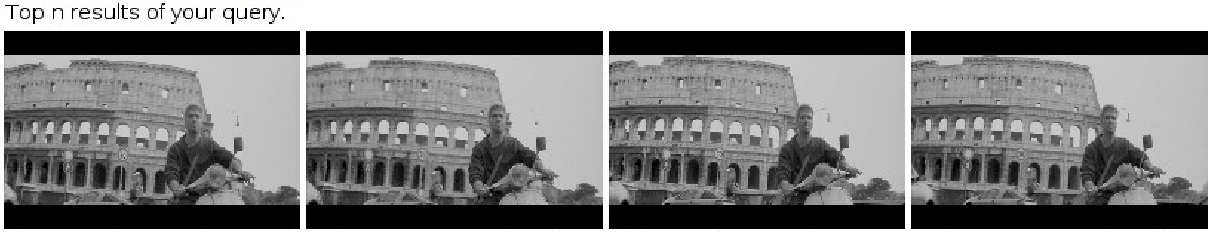


ImageSearch at the VizCentre

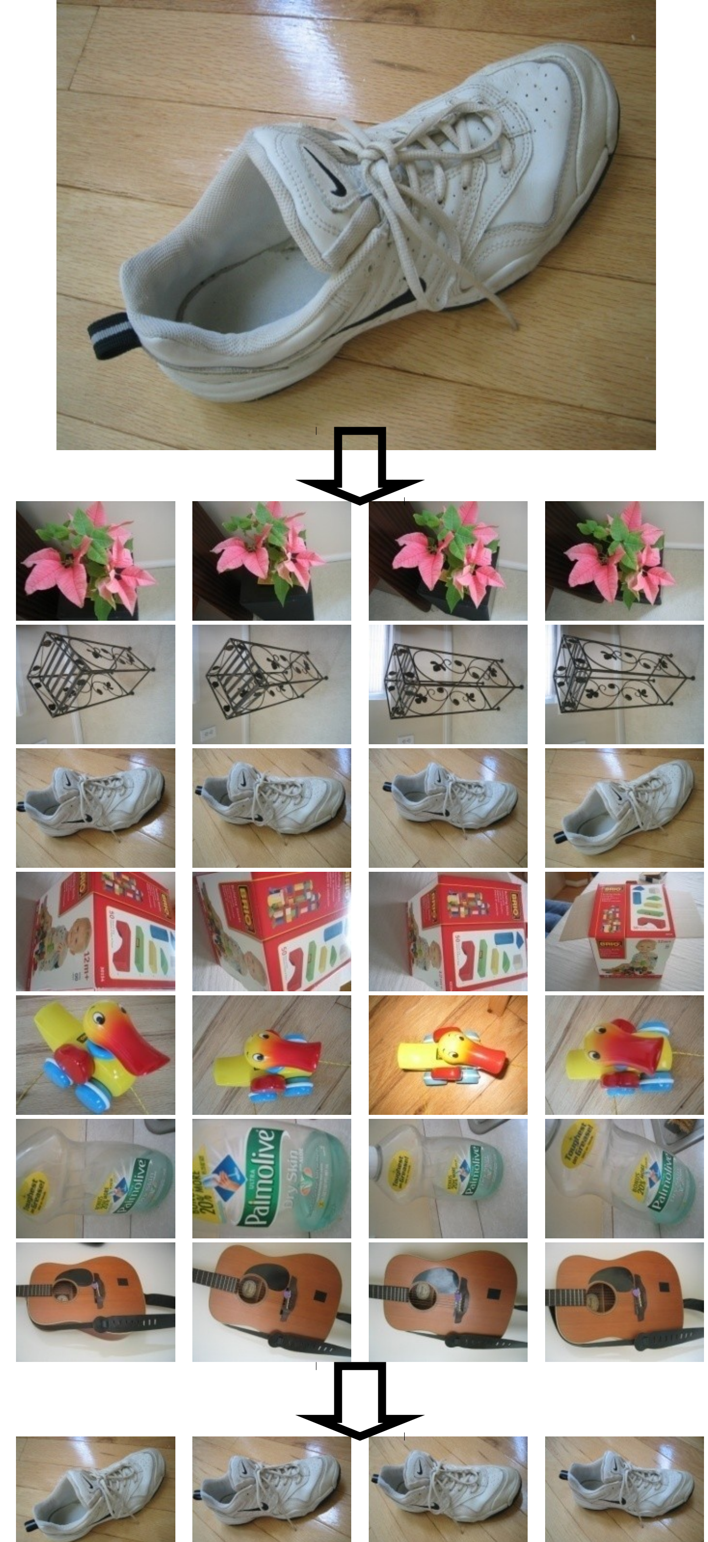
New query: Browse... Send File

File is 500x320

Top n results of your query.



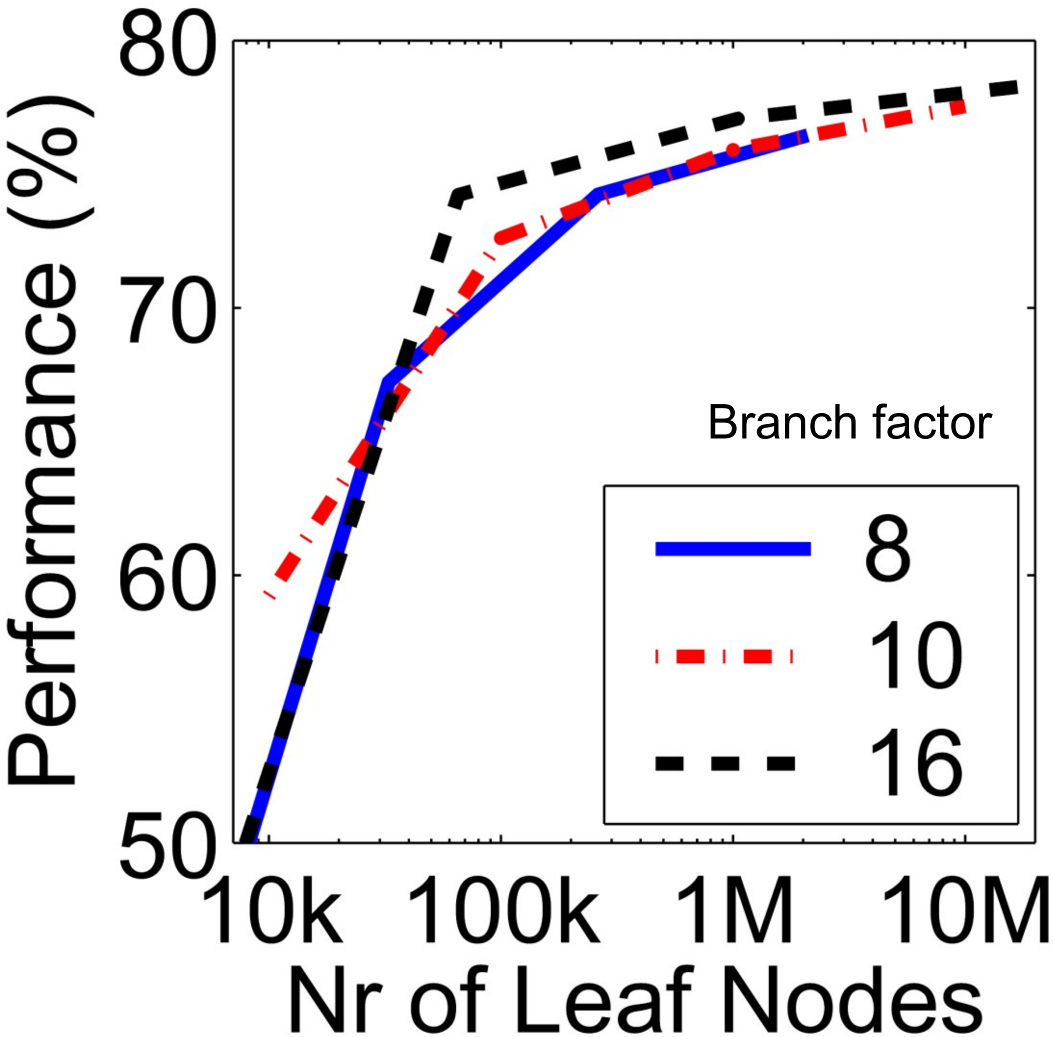
bourne/im1000043322.pgm bourne/im1000043323.pgm bourne/im1000043326.pgm bourne/im1000043327.pgm



UK

Center for Visualization & Virtual Environments

More words is better

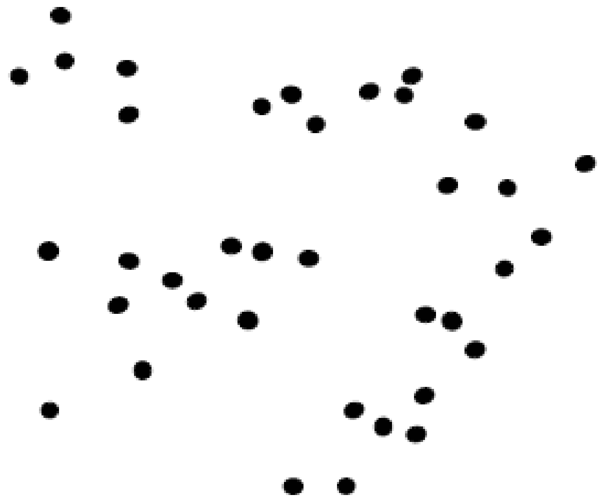


Improves Retrieval

Improves Speed

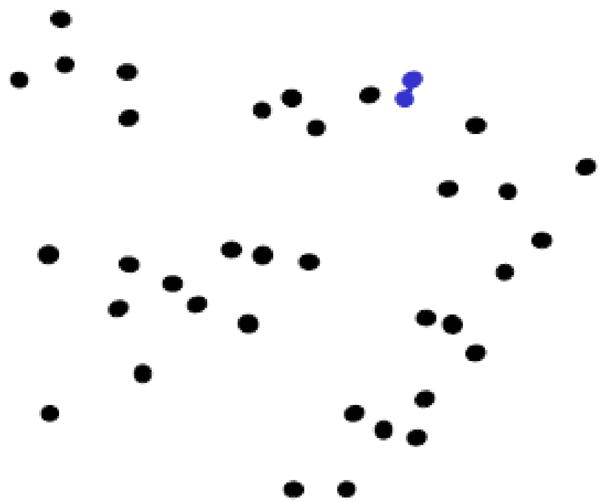
UK Center for Visualization & Virtual Environments

Agglomerative clustering



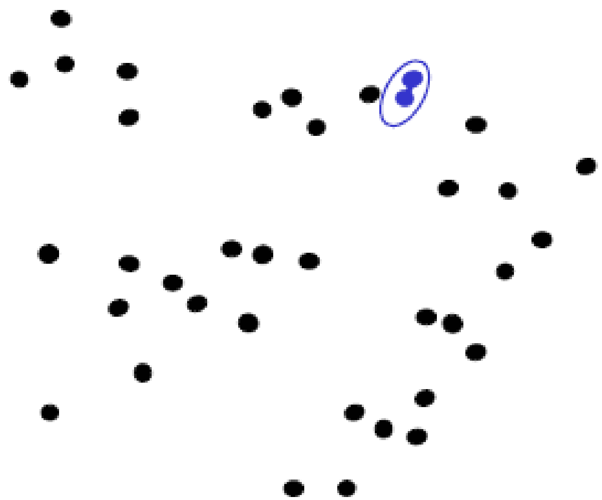
1. Say "Every point is its own cluster"

Agglomerative clustering

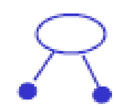


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

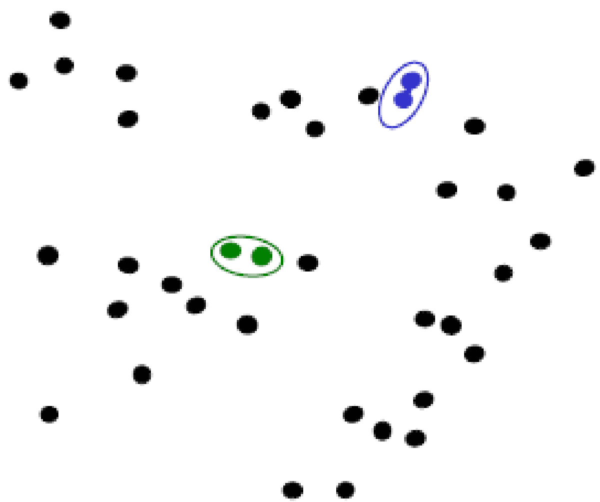
Agglomerative clustering



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

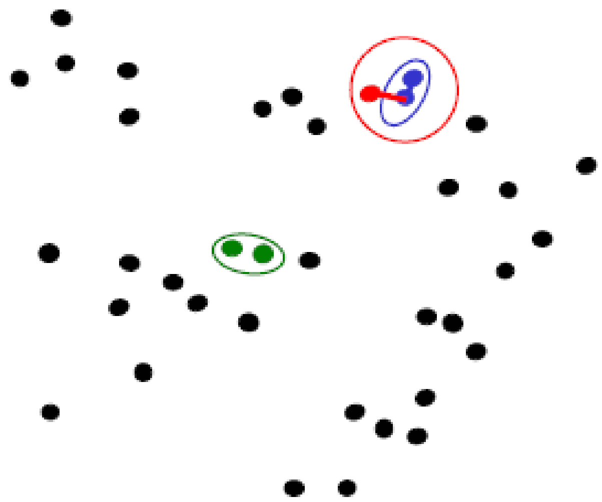


Agglomerative clustering

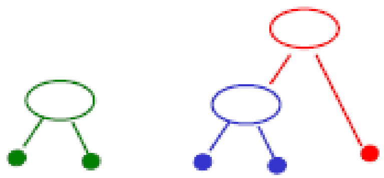


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering



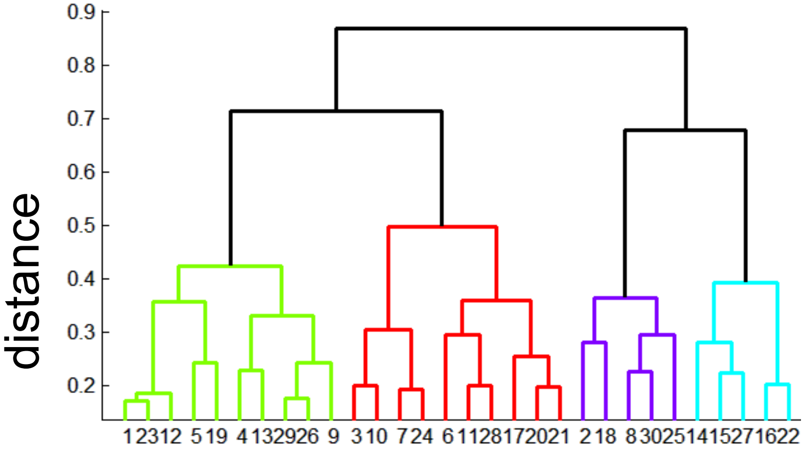
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



Agglomerative clustering

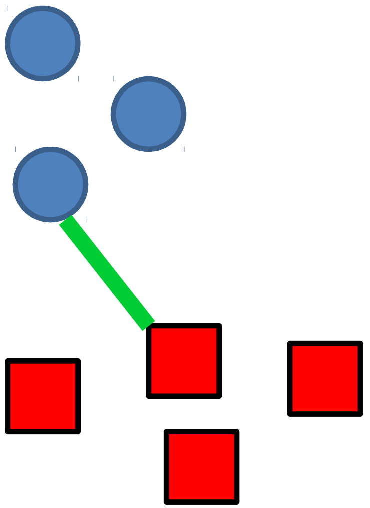
How many clusters?

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



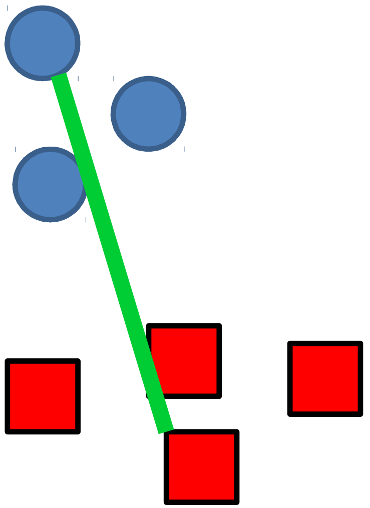
Agglomerative clustering

- How to define cluster similarity?
  - Single linkage: closest pair of points



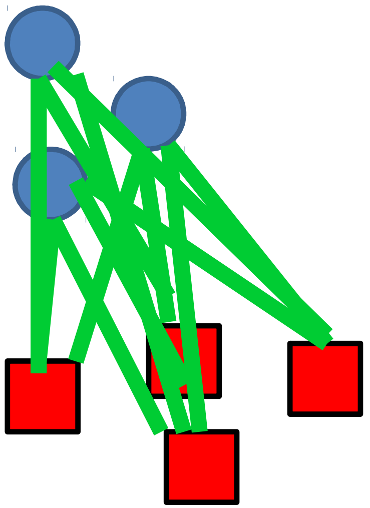
Agglomerative clustering

- How to define cluster similarity?
  - Single linkage: closest pair of points
  - Complete linkage: furthest pair of points

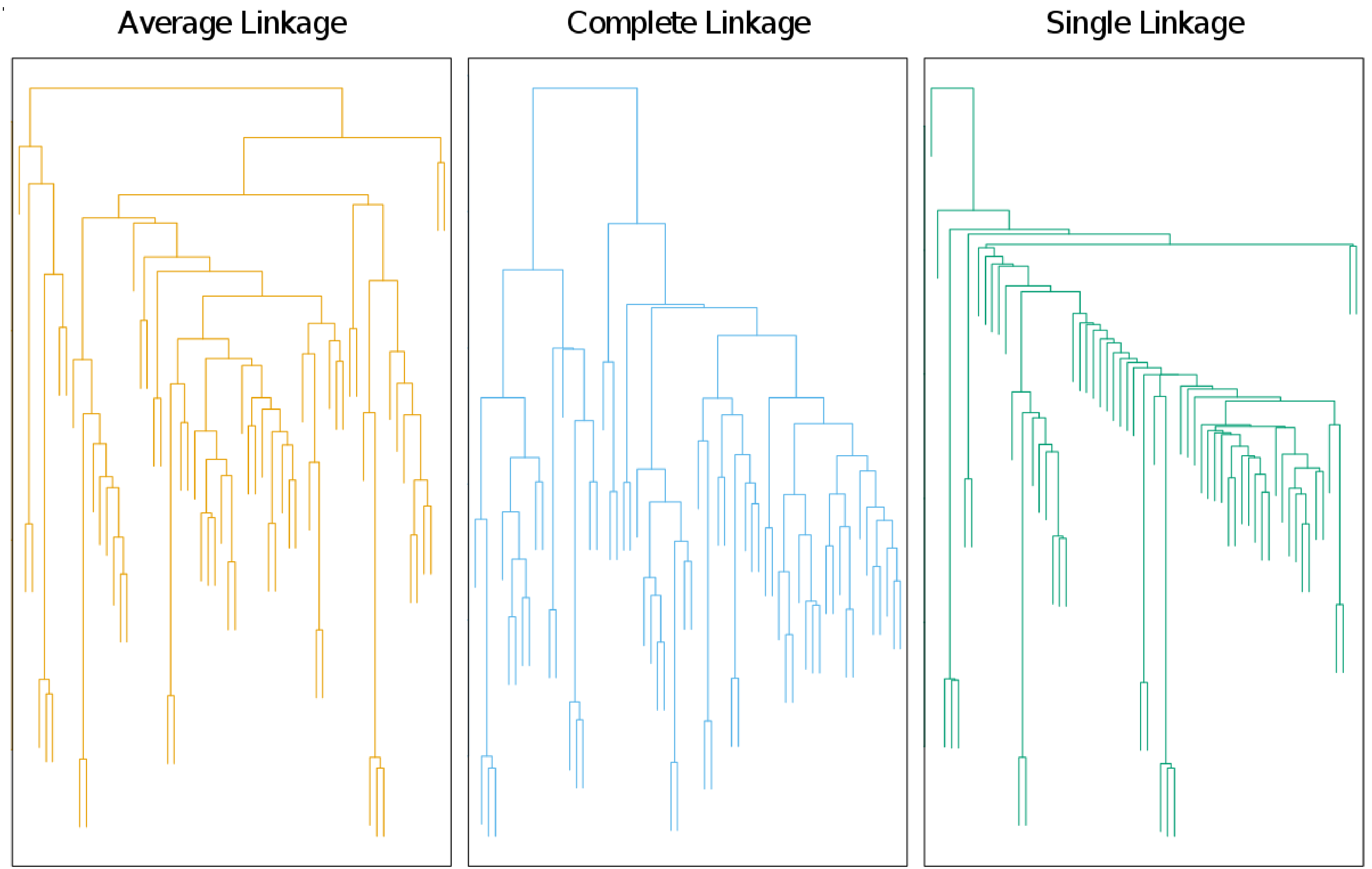


Agglomerative clustering

- How to define cluster similarity?
  - Single linkage: closest pair of points
  - Complete linkage: furthest pair of points
  - Average linkage: average over all pairs



Agglomerative clustering



Source: The Elements of Statistical Learning, Hastie et al.

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

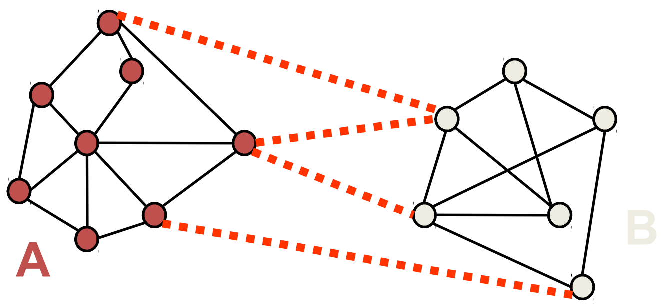
- Resulting hierarchy is sensitive to choice of similarity metric
- Still have to choose number of clusters or threshold

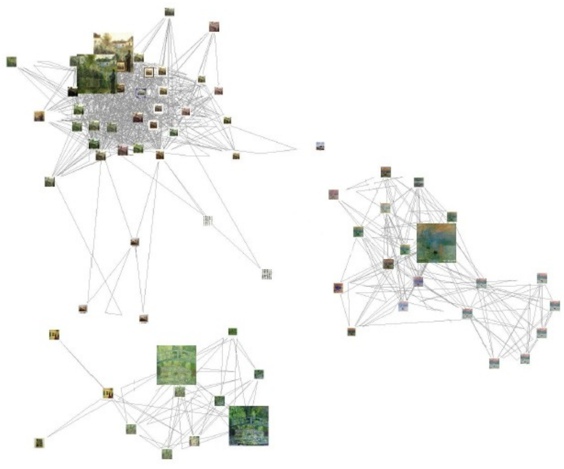
Divisive Clustering

- Top down hierarchical clustering
- Start with one large group and recursively split
- Possible splitting criteria:
  - Kmeans with K=2

Spectral Clustering

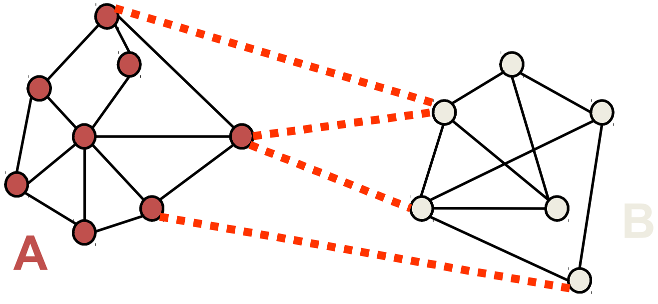
- Groups points based on pairwise affinities
- Use spectral techniques to determine a partitioning





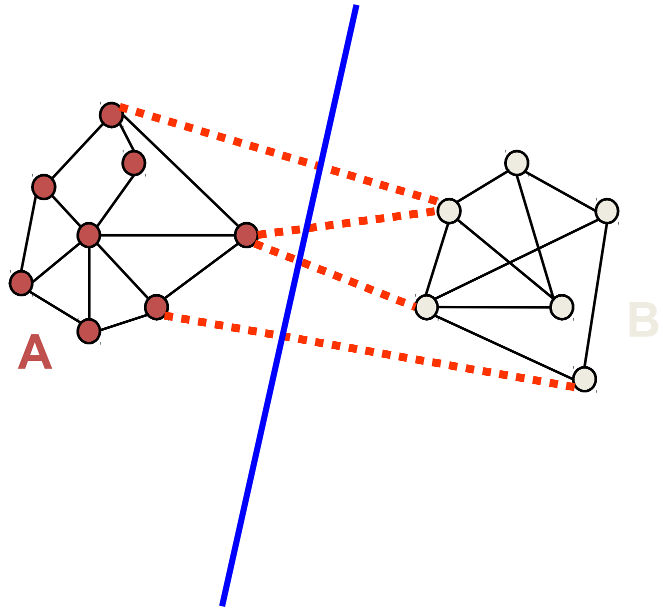
Spectral Clustering

- Build similarity graph

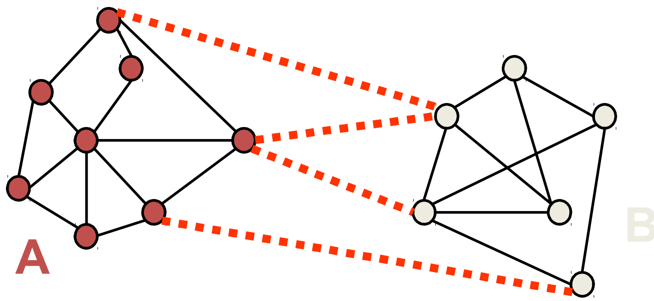


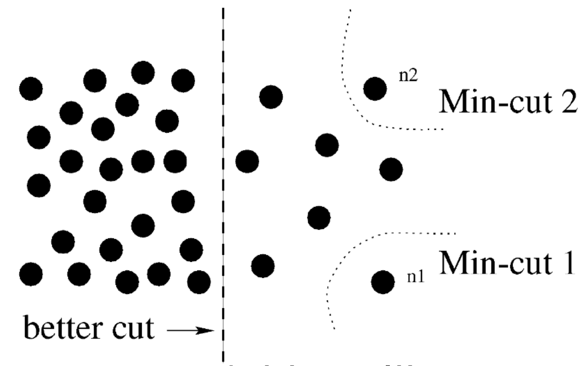
Spectral Clustering

- Build similarity graph
- Find a cut through the graph



Normalized Cuts





(Shi, Malik, TPAMI 2000)

- Partition the graph G = (V, E) into two disjoint subsets A and B to:
  - Minimize the weights of the cut edges (dashed red)

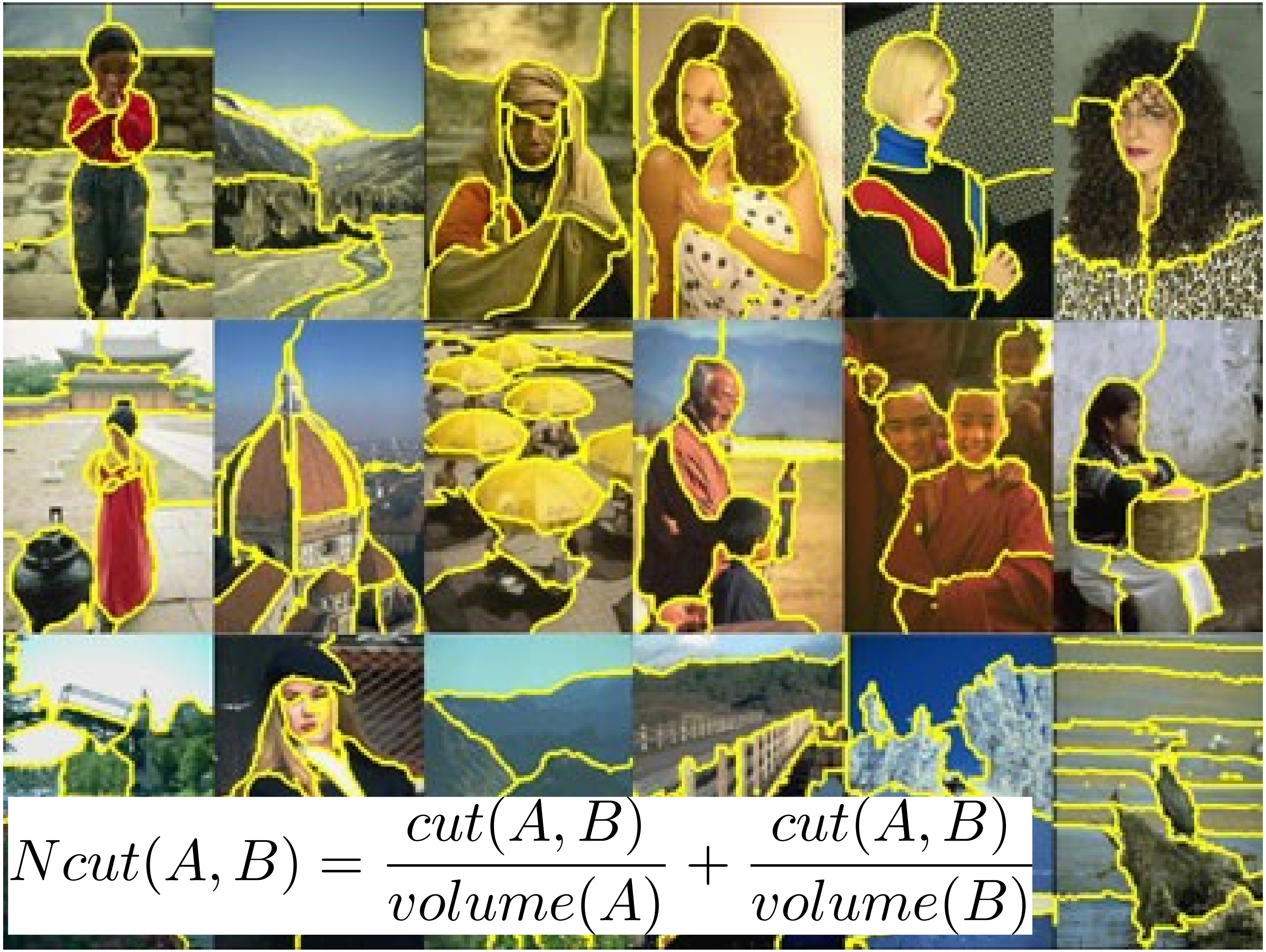
$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

- Introduce a normalization factor to avoid tiny partitions

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

Adapted from: Seitz

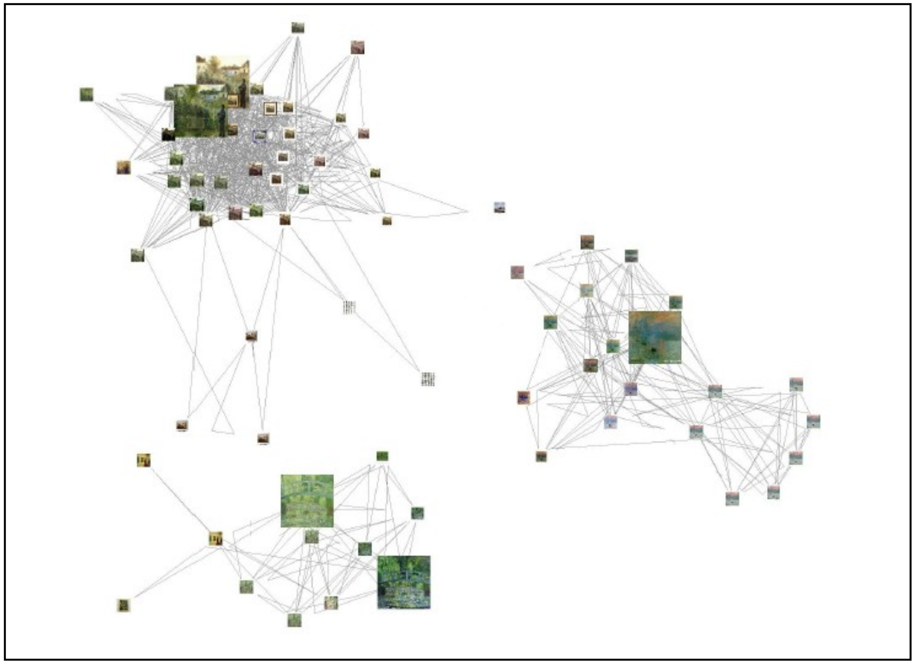
Normalized cuts for segmentation



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

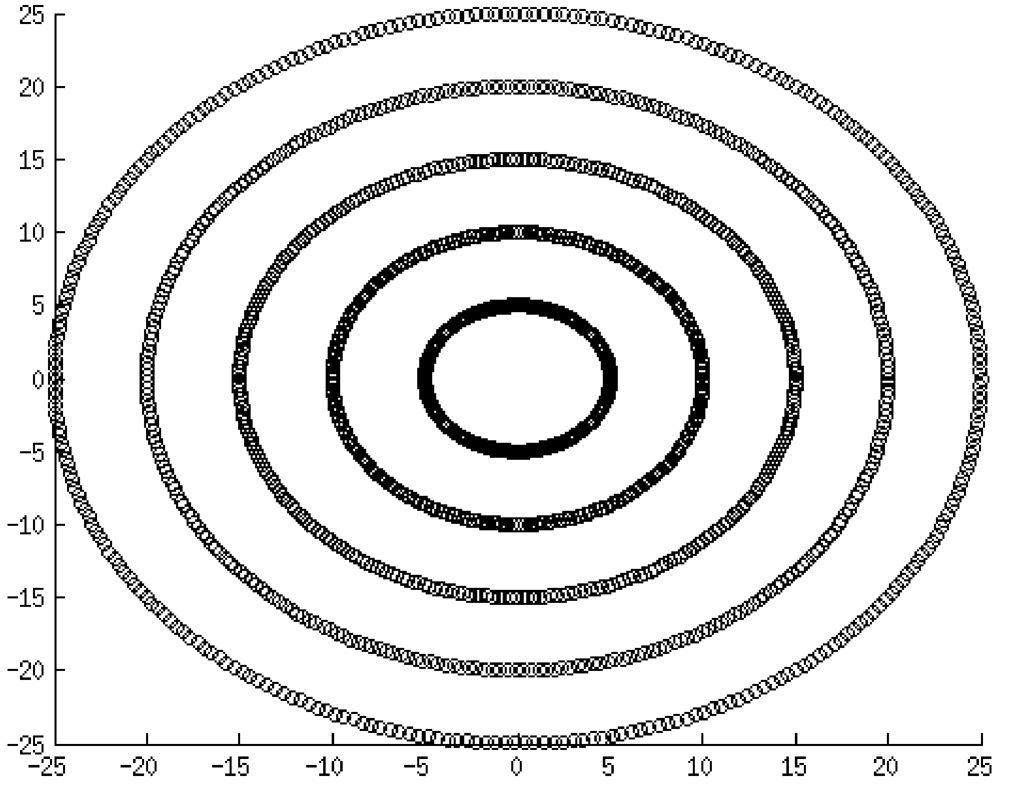
Visual PageRank

- 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
    - Rank by image search results by stationary distribution



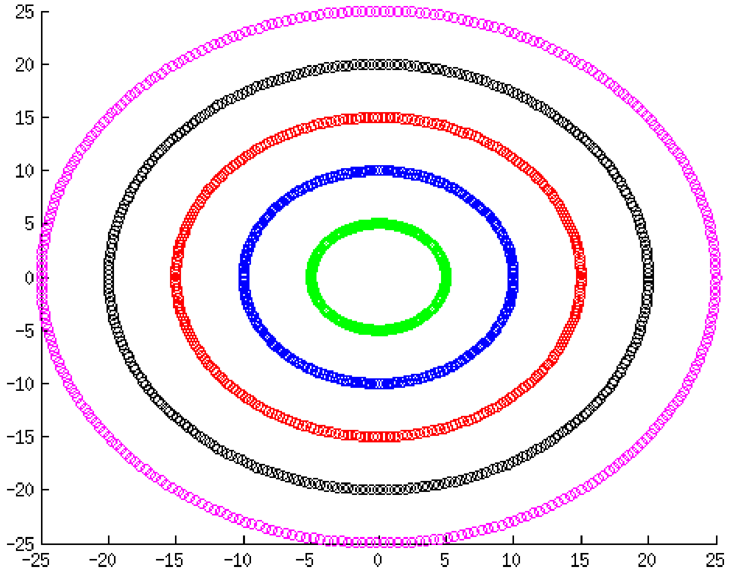
Jing Baluja 2008

Spectral Clustering

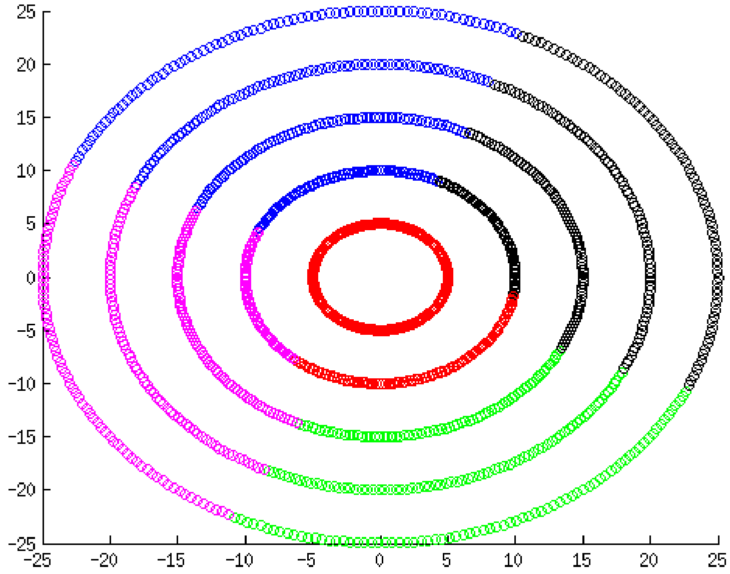


Goal: Find 5 clusters

Spectral Clustering



Spectral



K-means

Spectral Clustering

```
function[memb] = spectral_cluster(data, c, num_clusters)
% a simple implementation for unnormalized spectral clustering
% each row of data should be a data point

%% setup affinity matrix W
num_data = size(data,1);
D = zeros(num_data, num_data);
for i = 1:size(data,1)
    dists = data - repmat(data(i,:), [num_data, 1]);
    D(:,i) = sqrt(sum(dists.*dists, 2));
end
W = exp(-(D.*D)/c);
W = W-diag(diag(W)); % remove diagonal

%% setup degree matrix G
gs = sum(W, 2);
G = diag(gs);

%% compute laplacian
L = G - W;
[V, D] = eigs(L, num_clusters, 'sm');
memb = kmeans(V, num_clusters);
end
```

Spectral Clustering

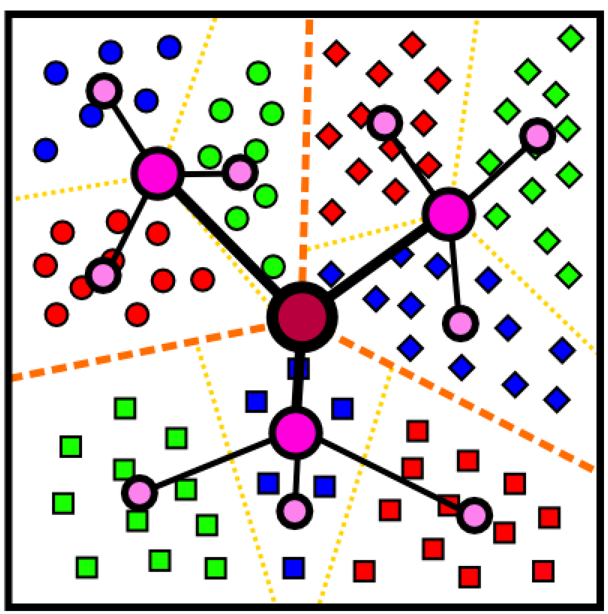
- Pros:
  - Fast for sparse datasets
  - Can output partitions with complex shapes
- Cons:
  - Hard to determine membership of unseen samples
  - Computationally expensive for large, dense datasets

How do we cluster?

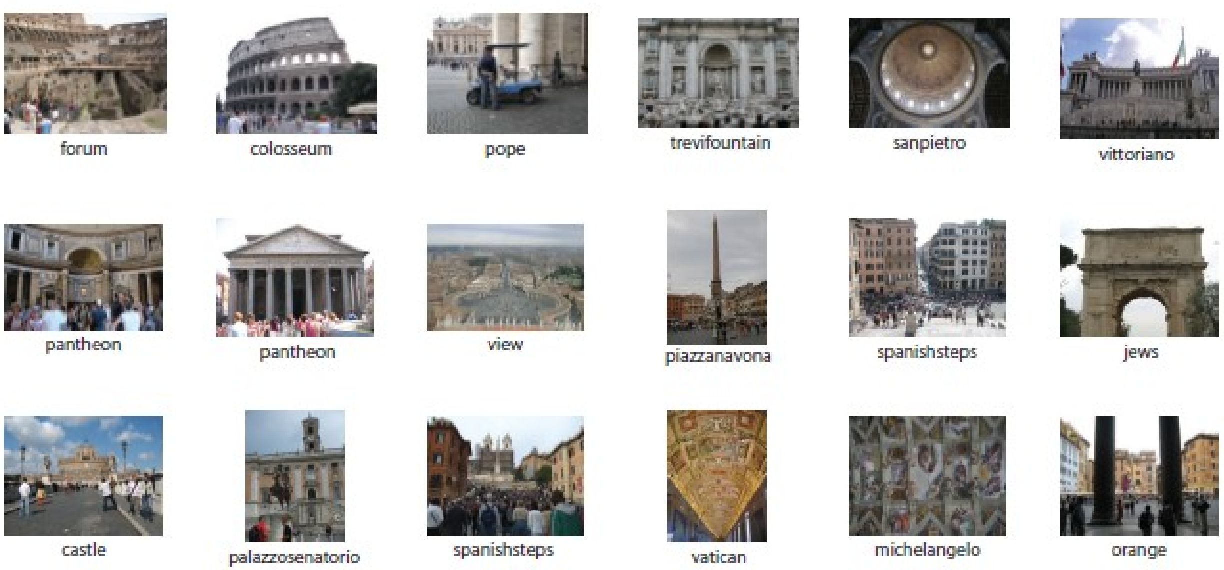
- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Graph-based clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Which algorithm to use?

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



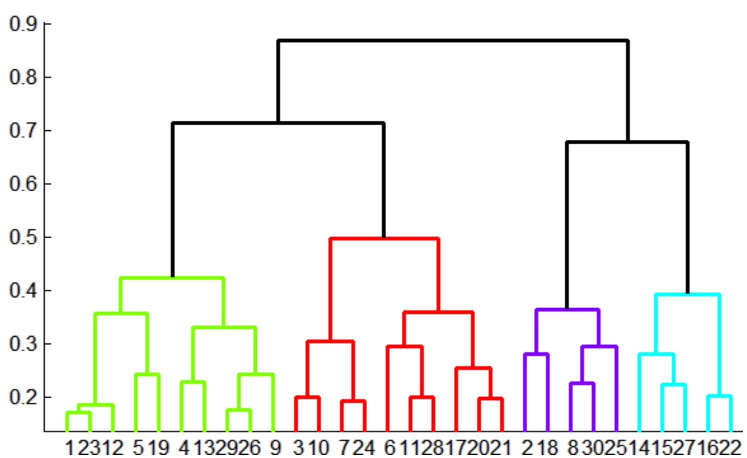
Quantization for computing histograms

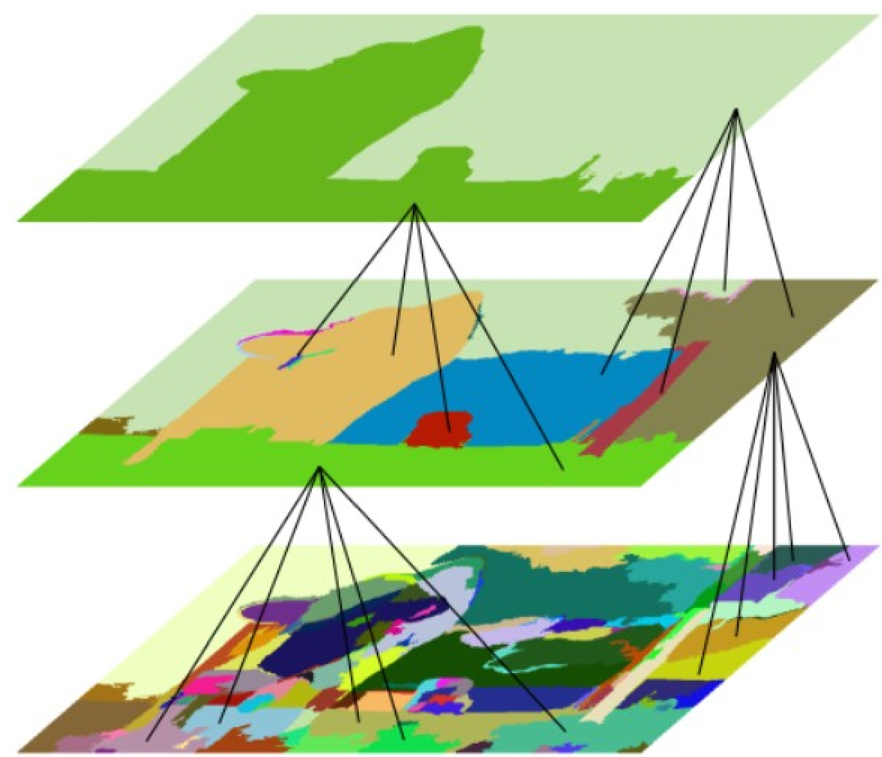


Summary of 20,000 photos of Rome using "greedy k-means"
http://grail.cs.washington.edu/projects/canonview/

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

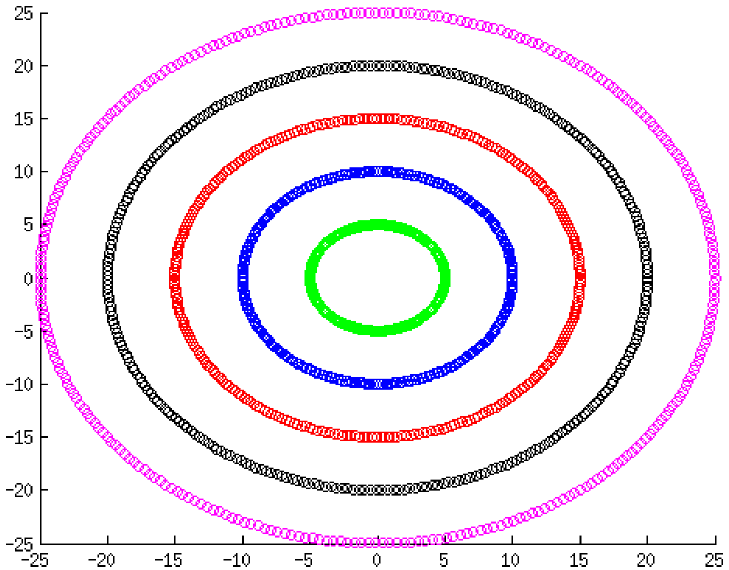




van de Sande et. al

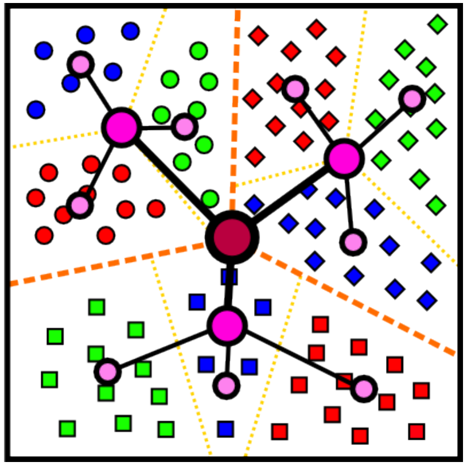
Which algorithm to use?

- Image segmentation: spectral clustering
  - Captures pairwise connectivities
  - Allows for variances within a partition

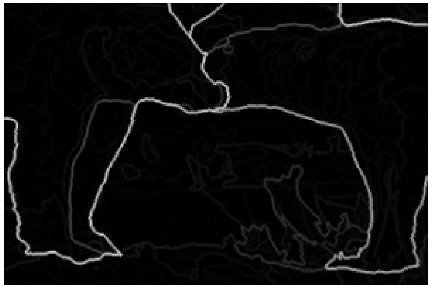


Things to remember

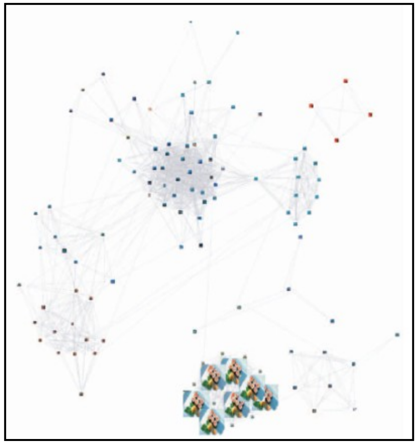
- K-means useful for summarization, building dictionaries of patches, general clustering
  - Fast object retrieval using visual words and inverse index table



- Agglomerative clustering useful for segmentation, general clustering



- Spectral clustering useful for determining relevance, general clustering, segmentation



Next class

- Gestalt grouping
- Image segmentation
  - Mean-shift segmentation
  - Watershed segmentation