CS 664 Slides #11
Image Segmentation

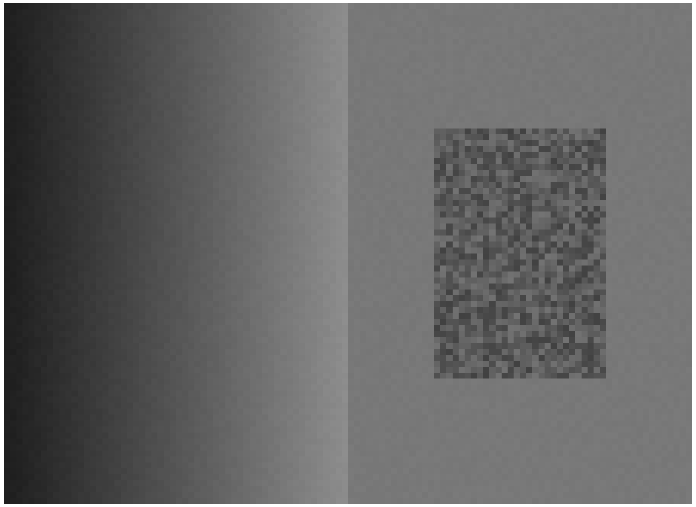
Prof. Dan Huttenlocher
Fall 2003

Image Segmentation

- Find regions of image that are “coherent”
- “Dual” of edge detection
  - Regions vs. boundaries
- Related to clustering problems
  - Early work in image processing and clustering
- Many approaches
  - Graph-based
    - Cuts, spanning trees, MRF methods
  - Feature space clustering
  - Mean shift

A Motivating Example

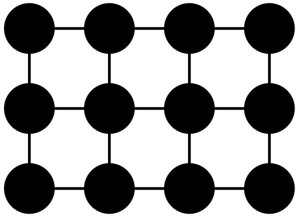
- Image segmentation plays a powerful role in human visual perception
  - Independent of particular objects or recognition



This image has three perceptually distinct regions

Graph Based Formulation

- $G=(V,E)$ with vertices corresponding to pixels and edges connecting neighboring pixels



4-connected or 8-conneted

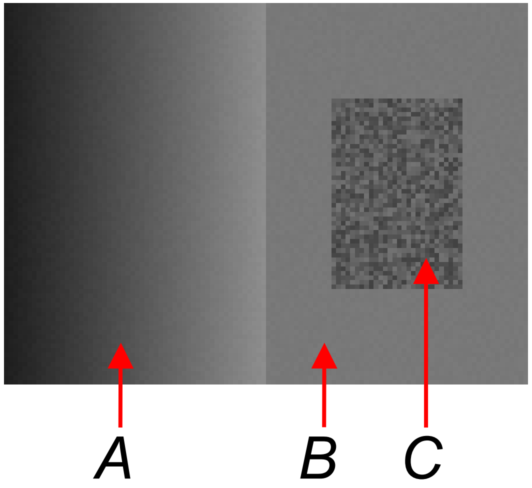
- Weight of edge is magnitude of intensity difference between connected pixels
- A *segmentation*, S, is a partition of V such that each $C \in S$ is connected

Important Characteristics

- Efficiency
  - Run in time essentially linear in the number of image pixels
    - With low constant factors
    - E.g., compared to edge detection
- Understandable output
  - Way to describe what algorithm does
    - E.g., Canny edge operator and step edge plus noise
- Not purely local
  - Perceptually important

Motivating Example

- Purely local criteria are inadequate
  - Difference along border between A and B is less than differences within C
- Criteria based on piecewise constant regions are inadequate (e.g., Potts MRF)
  - Will arbitrarily split A into subparts



MST Based Approaches

- Graph-based representation
  - Nodes corresponding to pixels, edge weights are intensity difference between connected pixels
- Compute minimum spanning tree (MST)
  - Cheapest way to connect all pixels into single component or "region"
- Selection criterion
  - Remove certain MST edges to form components
    - Fixed threshold
    - Threshold based on neighborhood
      - How to find neighborhood

Component Measure

- Instead of constructing MST based on just the edge weights
  - Consider properties of two components being merged when adding an edge
- Recall Kruskal’s MST algorithm adds edges from lowest to highest weight
  - Only when connect distinct components
- Apply criterion based on components to further filter added edges
  - Form of criterion limited by considering edges weight ordered

Measuring Component Difference

- Let *internal difference* of a component be maximum edge weight in its MST

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

  - Smallest weight such that all pixels of C are connected by edges of at most that weight
- Let *difference* between two components be minimum edge weight connecting them

$$Dif(C_1, C_2) = \min_{v_i \in C_1,\, v_j \in C_2} w((v_i, v_j))$$

  - Note: infinite if there is no such edge

Region Comparison Function

- Two components judged to be distinct when $Dif(C_1,C_2)$ large relative to $Int(C_1)$ or $Int(C_2)$
  - Require that it be *sufficiently* larger
  - Controlled by (non-negative) threshold function τ
- Region comparison function $g(C_1,C_2)$ is true when regions should be distinct, i.e., when

$$Dif(C_1,C_2) > MInt(C_1,C_2)$$

where $MInt(C_1,C_2)$

$$= \min(Int(C_1)+\tau(C_1),\ Int(C_2)+\tau(C_2))$$

About the Threshold Function τ

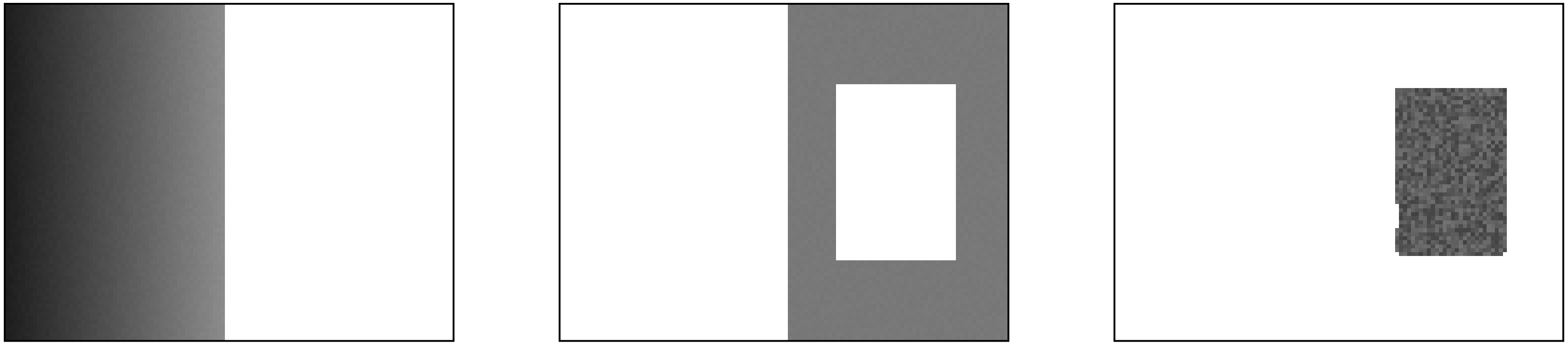
- Intuitively $Int(C)$ estimates local differences over component
  - Small components give underestimate of local difference – neighboring pixels tend to be similar
    - Thus τ should be large in this case
- Use a function inversely proportional to component size $\tau(C) = k / |C|$
  - k is a parameter of the method that captures "scale of observation"
    - Larger k means prefer larger components
  - Other functions possible, e.g., based on shape

The Algorithm

0. Sort edges of E into $(e_1, \ldots, e_n)$, in order of non-decreasing edge weight
1. Initialize S with one component per pixel
2. For each e_q in $(e_1, \ldots, e_n)$ do step 3
3. If weight of e_q small relative to internal difference of components it connects then merge components, otherwise do nothing

   I.e., if $w(e_q) \leq MInt(C_i, C_j)$, where $C_i, C_j \in S$ are distinct components connected by e_q, then update S by merging C_i and C_j

Regions Found by the Algorithm



A B C

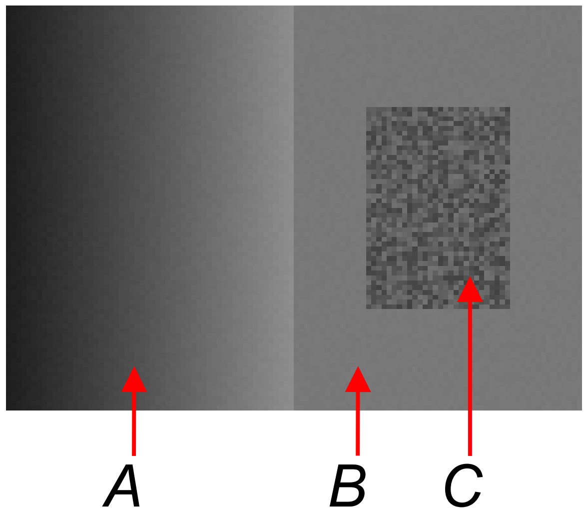
- Three main regions plus a few small ones
- Why the algorithm stops growing these
  - Weight of edges between A and B large wrt max weight MST edges of A and of B
  - Weight of edges between B and C large wrt max weight MST edge of B (but not of C)

Criteria for a Good Segmentation

- Some predicate for comparing two regions
  - Intuitively, evaluates whether there is evidence for a boundary between two regions
- A segmentation is *too fine* when predicate says no evidence for a boundary
  - Some pair of neighboring regions where predicate false
- A segmentation is *too coarse* when there is some refinement that is not too fine
  - A *refinement* is obtained by splitting one or more regions of a segmentation

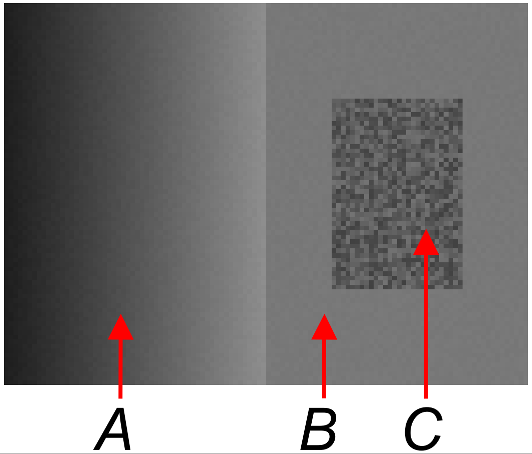
Good Segmentations and the Example

- Splitting A, B or C would be too fine
- Not splitting A from B or B from C would be too coarse



Other Algorithms and the Criteria

- Piecewise constant regions (or compact clusters in a color-based feature space)
  - Too fine: arbitrarily split ramp in A into pieces
- Breaking high cost edges in the MST of a graph corresponding to the image
  - Both: merge A with B or split C into multiple pieces



Properties of the Algorithm

- It is fast, $O(n \log n)$ for sorting in step 0 and $O(n\alpha(n))$ for the remaining steps
  - Using union-find with path compression to represent the partition, S
- It produces good segmentations
  - Neither too coarse nor too fine according to the above definitions
    - Despite being a greedy algorithm
- It yields the same results regardless of the order that equal-weight edges are considered
  - Proof a bit involved, won't discuss here

Components “Freeze”

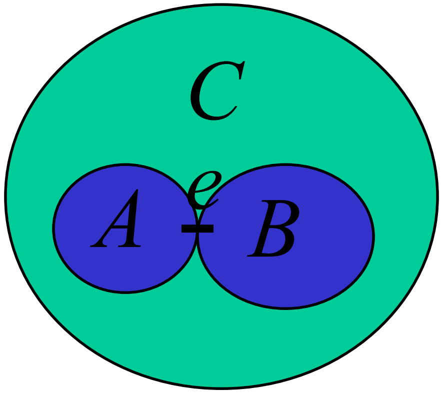
- When two components do not merge, one will be a component of the final segmentation
  - A merge decision is made for an edge e_q and the two components that it connects C_i, C_j
  - Say the merge does not occur because $w(e_q) > Int(C_i)+\tau(C_i)$
    - Then any subsequent merge involving C_i will also not occur, because edges are considered in non-decreasing weight order
  - Analogous for C_j, so when a merge fails one or both of the components involved “freeze”

Segmentation Not Too Fine

- Follows readily from fact that components "freeze"
  - An edge between two components in final segmentation implies the algorithm decided not to merge when considering this edge
    - Component that caused this decision is frozen, so appears in the final segmentation
- Thus the decision that was true when the edge was considered remains true for the final segmentation

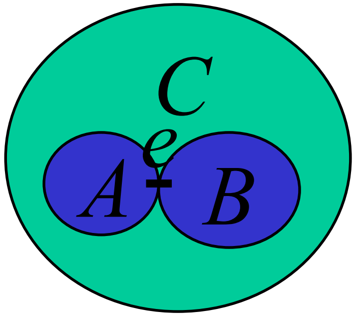
Segmentation Not Too Coarse

- Means any proper refinement is too fine
- Suppose was a proper refinement, T, of the final segmentation, S, that is not too fine
  - Consider the minimum weight edge, e, that is between two components A,B of T but is within a single component C of S



Sketch Continued

- All edges in MST of either A or B have weights smaller than $w(e)$, say it is A
  - Definition of not too fine, and predicate
- Thus algorithm creates A before considering e
  - Because all edges on boundary of A, but internal to C, have weight larger than $w(e)$
- Since T not too fine, the decision criterion implies the algorithm would freeze A when considering e



Closely Related Problems Hard

- What appears to be a slight change
  - Make Dif be quantile instead of min

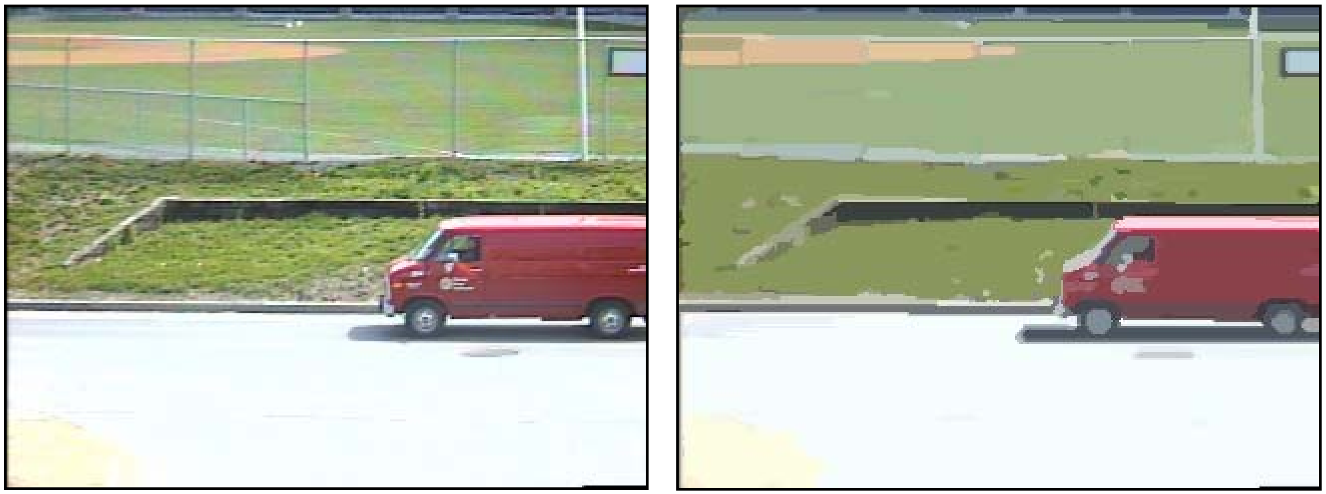
$$\text{k-th}_{\,v_i \in C_1,\, v_j \in C_2}\; w((v_i, v_j))$$

  - Desirable for addressing "cheap path" problem of merging based on one low cost edge
- Makes problem NP hard
  - Reduction from min ratio cut
    - Ratio of "capacity" to "demand" between nodes
- Other methods that we will see are also NP hard and approximated in various ways

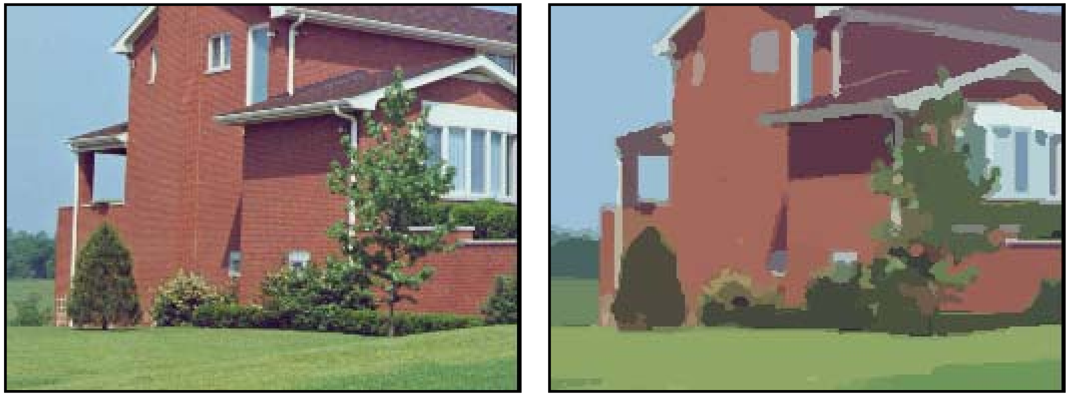
Some Implementation Issues

- Smooth images slightly before processing
  - Remove high variation due to digitization artifacts
- Sorting is dominant time in processing
  - For known edge distribution can in principle do better by binning
- Treat color images as three separate images
  - Components of segmentation are “intersection” of components from each of the three color planes
    - Motivation: significant change in any color channel should result in a region boundary

Some Example Segmentations



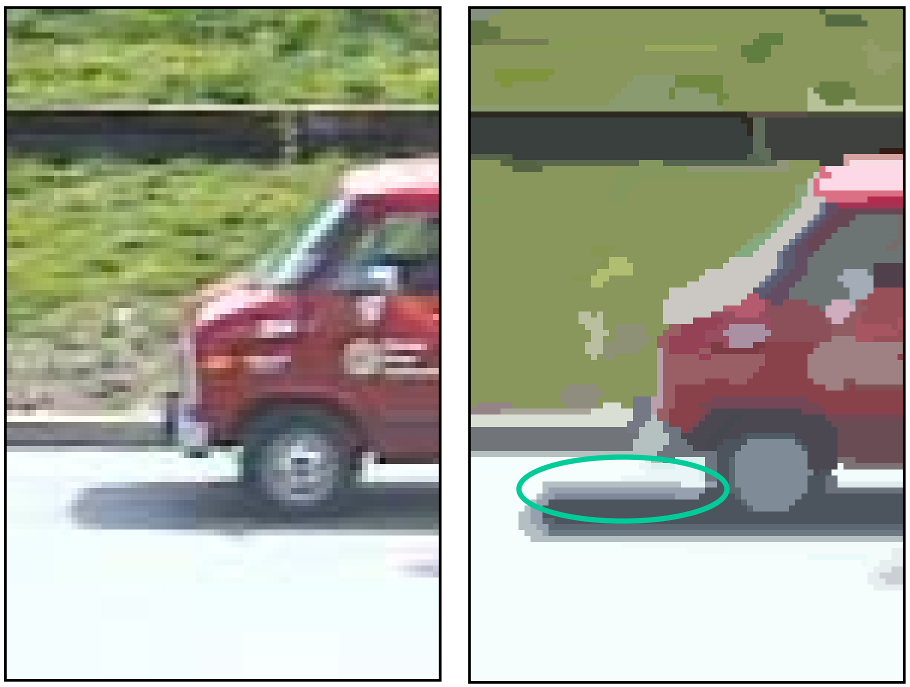
k=300
320 components
larger than 10



k=200
323 components
larger than 10

Some Shortcomings

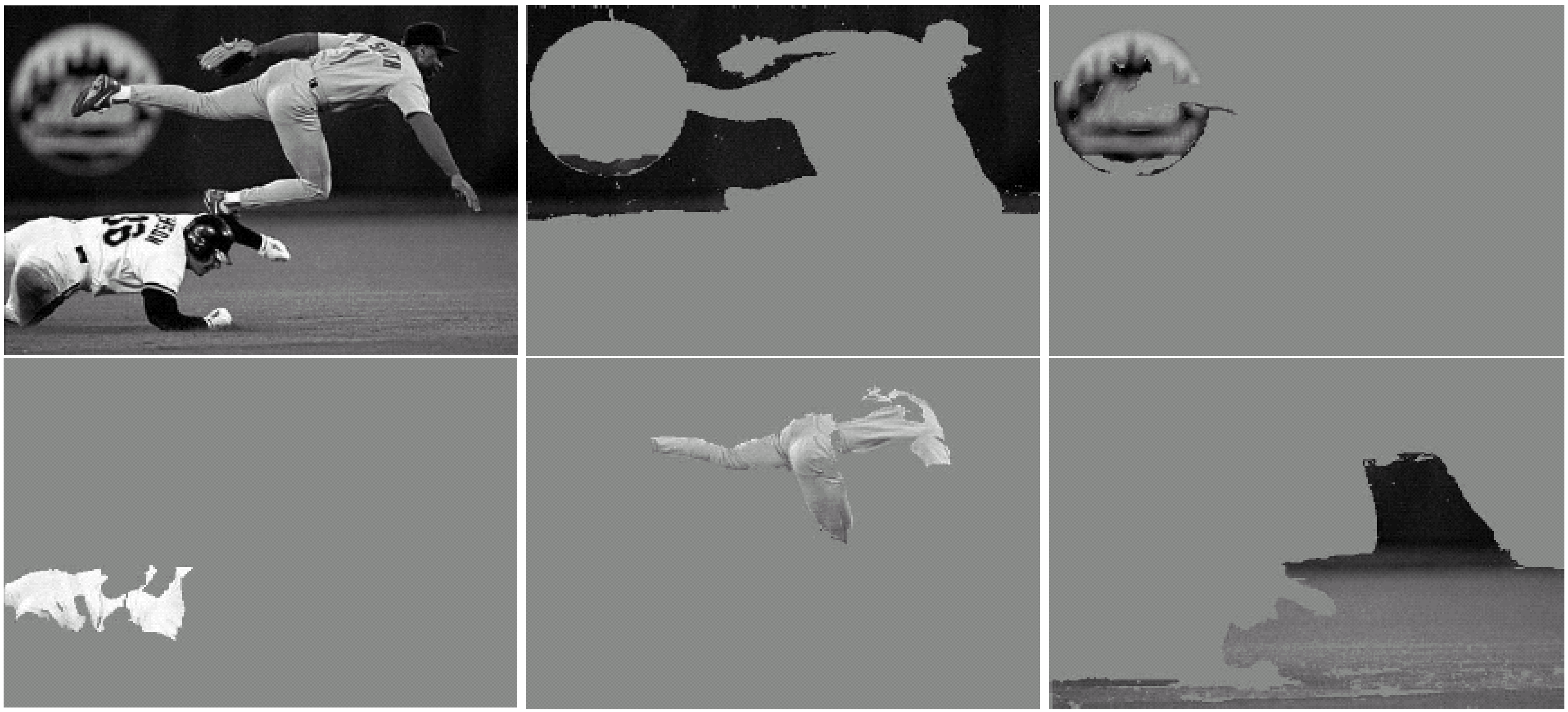
- Smoothing can introduce problems
  - “Extra regions” at boundaries
  - Creates “ramps” between regions, thus merge



Simple Object Examples

Monochrome Example

- Components locally connected (grid graph)
  - Sometimes not desirable

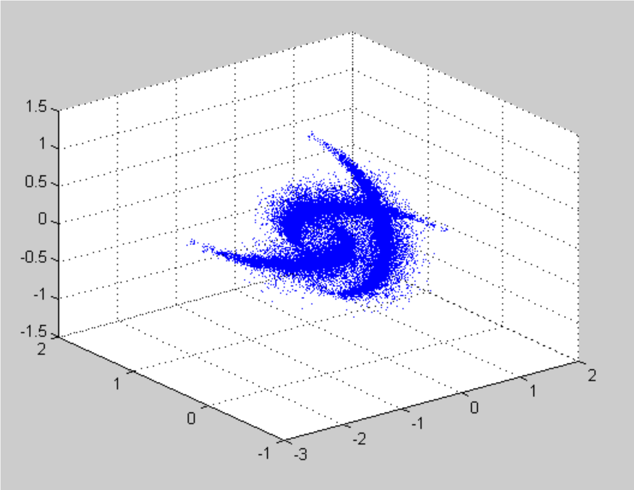


Clustering: Non-Local Components

- Points in d-dimensional space
  - Vertex for each point, edge weights based on distance in this space
- Intuitively, Int measures "density" of clusters
  - Smallest dilation radius such that all points in the cluster are connected
  - When clusters separated by nearly same distance as their "densities" then segmentation is too fine
- For efficiency use a graph with $O(|V|)$ edges
  - Use Mount's approximate nearest neighbor algorithm to find nearest neighbors

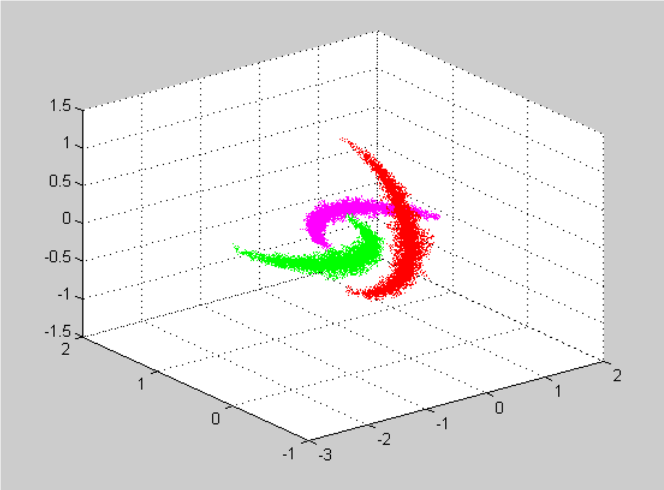
Clustering Gaussian Point Data

Note: Gaussian not constant density

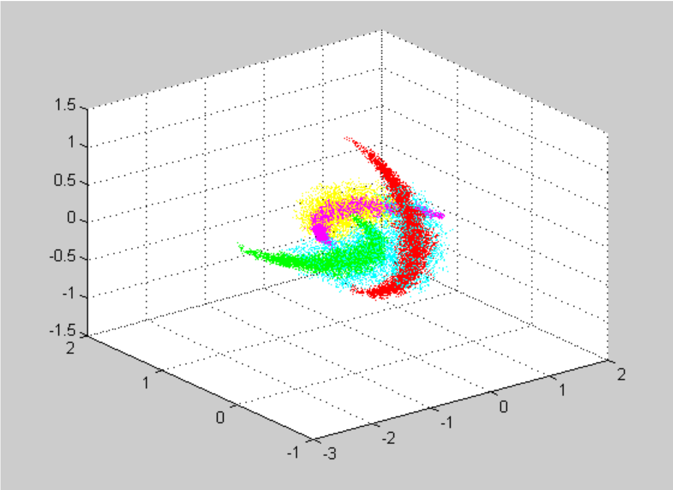


Graph connecting four nearest neighbors to each vertex

$k = 1$



3 largest clusters, 75% classified



5 largest clusters, 95% classified

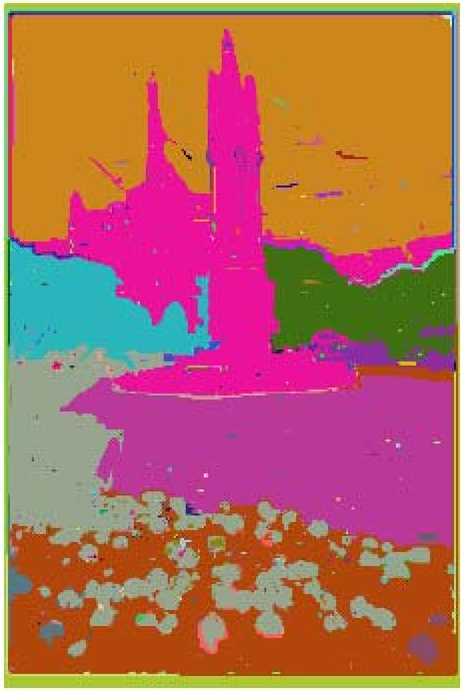
Clustering for Image Segmentation

- Treat each pixel as a point in a feature space
  - More than just local intensity or color, incorporate spatial, texture, motion or other differences
- Now regions of segmentation need not be connected in image
- Practical issue, relatively expensive to find nearest neighbors for graph
  - Can use neighbors in some fixed distance, but restricts regions that can be found
  - In examples here use 4 nearest neighbors

Example Clustering of Image Data

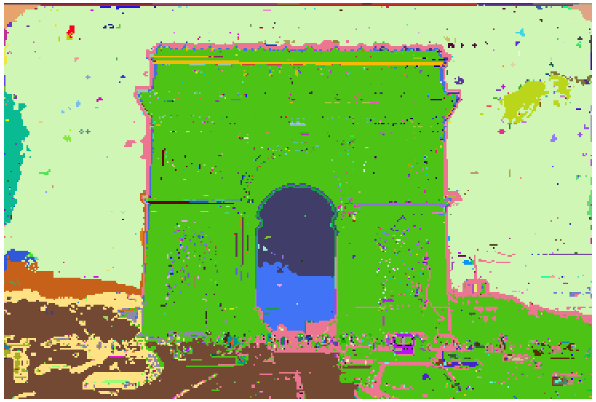
- Segmentation using difference in R,G,B values and in position
  - Distance of 5 pixels same as 1 intensity unit

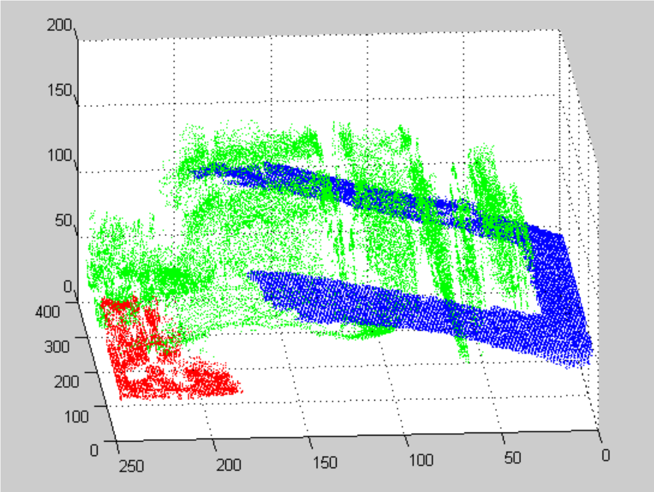
Non-Local Component



About Clustering for Image Data

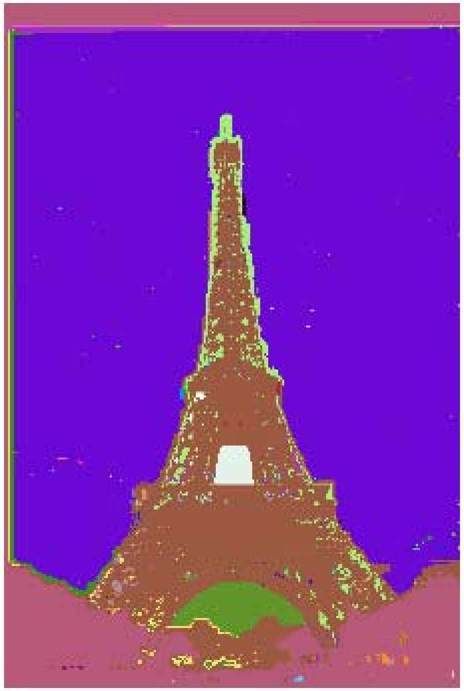
- Meaningful regions in image are not necessarily compact in feature space
- Cheap path in feature space not always apparent in image





Additional Example

- High variability in illuminated tower pixels



Beyond Grid Graphs

- Image segmentation methods using affinity (or cost) matrices
  - For each pair of vertices v_i, v_j an associated weight w_{ij}
    - Affinity if larger when vertices more related
    - Cost if larger when vertices less related
  - Matrix $W=[\ w_{ij}\ ]$ of affinities or costs
    - W is large, avoid constructing explicitly
    - For images affinities tend to be near zero except for pixels that are nearby
      - E.g., decrease exponentially with distance
    - W is sparse

Cut Based Techniques

- For costs, natural to consider minimum cost cuts
  - Removing edges with smallest total cost, that cut graph in two parts
  - Graph only has non-infinite-weight edges
- For segmentation, recursively cut resulting components
  - Question of when to stop
- Problem is that cuts tend to split off small components
  - Few edges

Normalized Cuts

- A number of normalization criteria have been proposed
- One that is commonly used

$$\text{Ncut}(A,B) = \frac{\text{cut}(A,B)}{\text{assoc}(A,V)} + \frac{\text{cut}(A,B)}{\text{assoc}(B,V)}$$

- Where cut(A,B) is standard definition

$$\sum_{i \in A, j \in B} w_{ij}$$

- And $\text{assoc}(A,V) = \sum_j \sum_{i \in A} w_{ij}$

Computing Normalized Cuts

- Has been shown this is equivalent to an integer programming problem, minimize

$$\frac{y^T (D-W) y}{y^T D\, y}$$

- Subject to the constraint that $y_i \in \{1,b\}$ and $y^T D 1 = 0$
  - Where 1 vector of all 1's
- W is the affinity matrix
- D is the degree matrix (diagonal)

$$D(i,i) = \sum_j w_{ij}$$

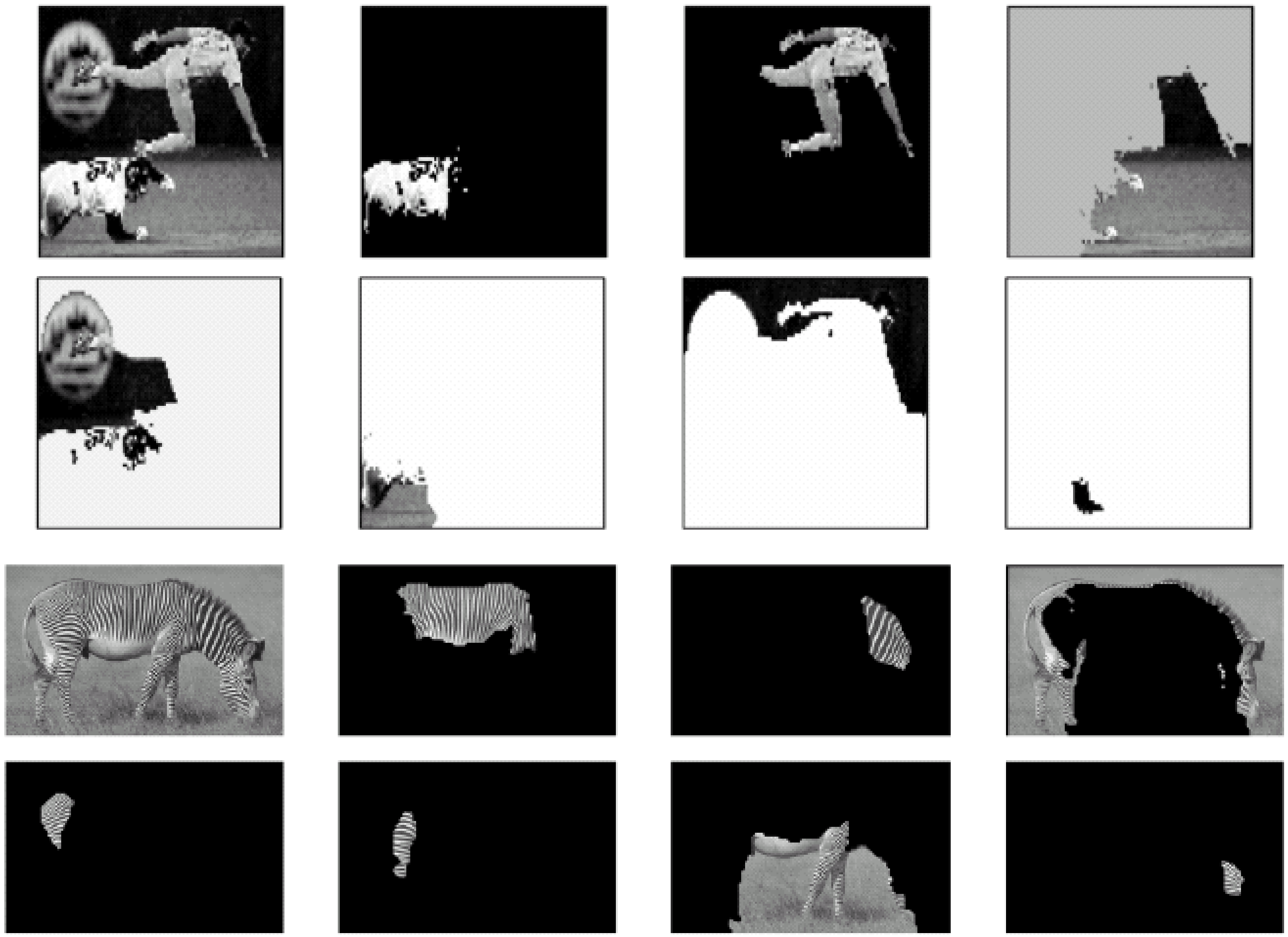
Approximating Normalized Cuts

- Integer programming problem NP hard
  - Instead simply solve continuous (real-valued) version
  - This corresponds to finding second smallest eigenvector of

$$(D\text{-}W)y_i = \lambda_i D y_i$$

- Widely used method
  - Works well in practice
    - Large eigenvector problem, but sparse matrices
    - Often resolution reduce images, e.g, 100x100
  - But no longer clearly related to cut problem

Normalized Cut Examples



Another Look at the Problem

- Consider eigen analysis of affinity matrix

$$W = [\ w_{ij}\ ]$$

  - Note W is symmetric; for images $w_{\mathbf{ij}}=w_{\mathbf{ji}}$
  - W also essentially block diagonal
    - With suitable rearrangement of rows/cols so that vertices with higher affinity have nearer indices
    - Entries far from diagonal are small (though not quite zero)
- Eigenvectors of W
  - Recall for real, symmetric matrix forms an orthogonal basis
    - Axes of decreasing “importance”

Structure of W

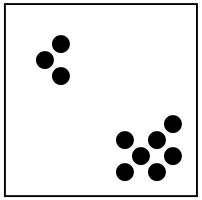
- Eigenvectors of block diagonal matrix consist of eigenvectors of the blocks
  - Padded with zeroes
- Note rearrangement so that clusters lie near diagonal only conceptual
  - Eigenvectors of permuted matrix are permutation of original eigenvectors
- Can think of eigenvectors as being associated with high affinity “clusters”
  - Eigenvectors with large eigenvalues
  - Approximately the case

Structure of W

- Consider case of point set where affinities

$$w_{ij}=\exp(-(y_i-y_j)^2/\sigma^2)$$

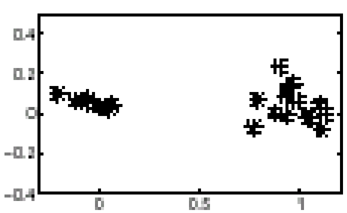
- With two clusters
  - Points indexed to respect clusters for clarity
- Block diagonal form of W
  - Within cluster affinities A, B for clusters
  - Between cluster affinity C



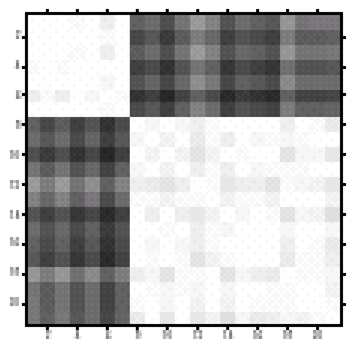
$$W=\begin{pmatrix} A & C \\ C^T & B \end{pmatrix}$$

First Eigenvector of W

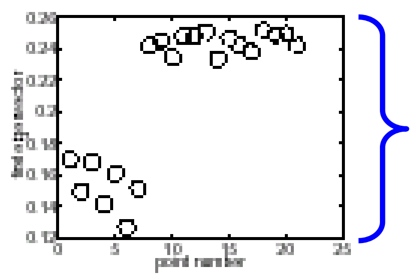
- Recall, vectors x_i satisfying $Wx_i=\lambda_i x_i$
- Consider ordered by eigenvalues λ_i
  - First eigenvector $x_{\mathbf{1}}$ has largest eigenvalue $\lambda_{\mathbf{1}}$
- Elements of first eigenvector serve as "index vector"
  - Selecting elements of highest affinity cluster



Points in plane



W

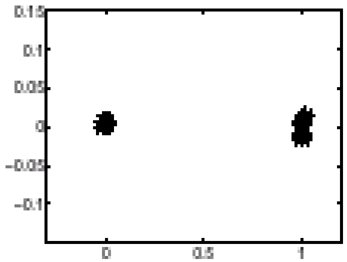


Magnitude of elements

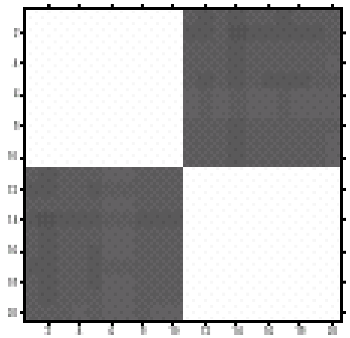
Elements of $x_{\mathbf{1}}$

Clustering

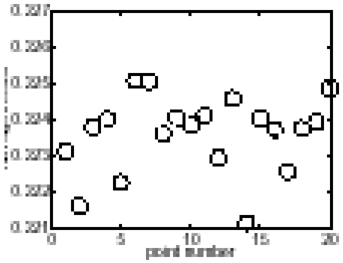
- First eigenvector of W has been suggested as clustering or segmentation criterion
  - For selecting most significant segment
  - Then recursively segment remainder
- Problematic when similar affinity clusters (regions)



Points in plane



W



Elements of x_1

Understanding Normalized Cuts

- Intractable discrete graph problem used to motivate continuous (real valued) problem
  - Find second *smallest* “generalized eigenvector”

$$(D-W)x_i = \lambda_i D x_i$$

  - Where D is (diagonal) degree matrix $d_{ii} = \sum_j w_{ij}$
- Can be viewed in terms of first two eigenvectors of normalized affinity matrix
  - Let $N=D^{-1/2}WD^{-1/2}$
  - Note $n_{ij}=w_{ij}/(\sqrt{d_{ii}}\,\sqrt{d_{jj}})$
    - Affinity normalized by degree of the two nodes

Normalized Affinities

- Can be shown that
  - If x is an eigenvector of N with eigenvalue λ then $D^{-1/2}x$ is a generalized eigenvector of W with eigenvalue $1-\lambda$
  - The vector $D^{-1/2}1$ is an eigenvector of N with eigenvalue 1
- It follows that
  - Second smallest generalized eigenvector of W is ratio of first two eigenvectors of N
  - So ncut uses normalized affinity matrix N and first two eigenvectors rather than affinity matrix W and first eigenvector

Contrasting W and N

- Three simple point clustering examples
  - W, first eigenvector of W, ratio of first two eigenvectors of N (generalized eigenvector of W)

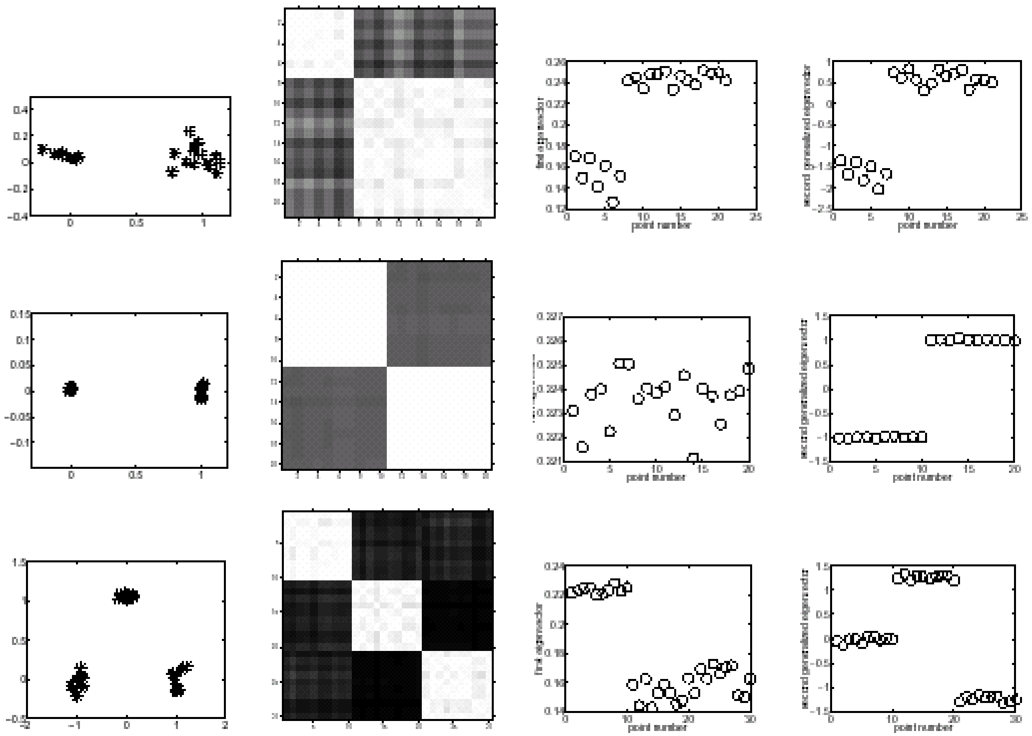
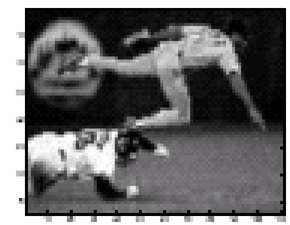


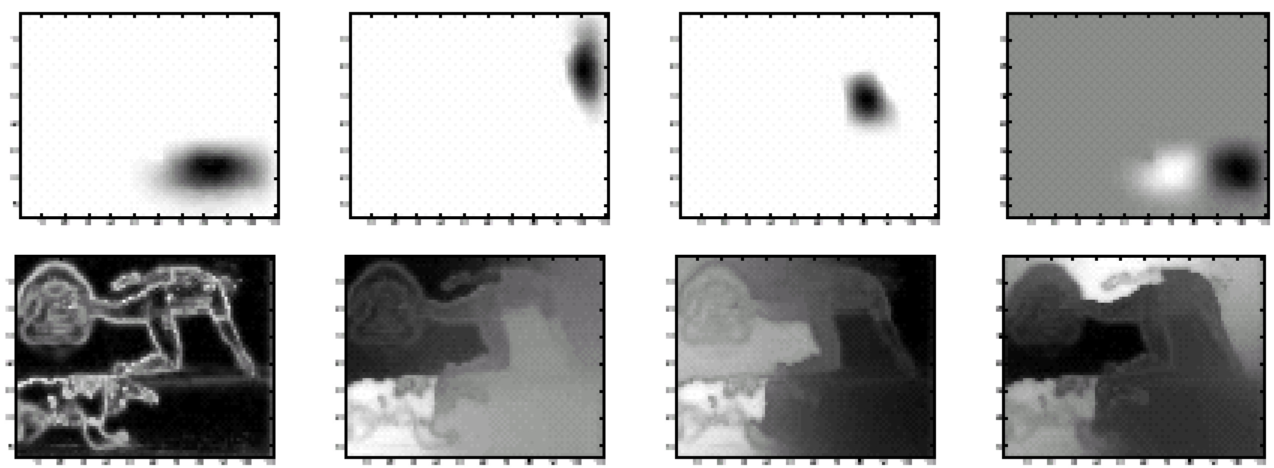
Image Segmentation

- Considering W and N for segmentation
  - Affinity a negative exponential based on distance in x,y,b space
- Eigenvectors of N more correlated with regions



First 4 eigenvectors of W

First 4 eigenvectors of N

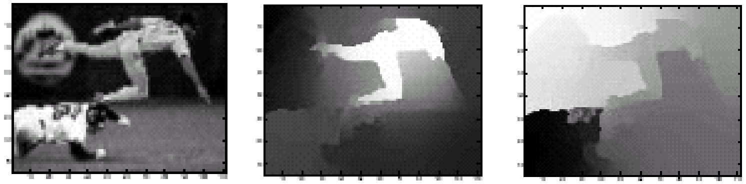


Using More Eigenvectors

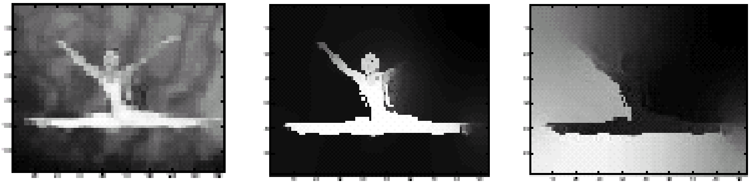
- Based on k largest eigenvectors
  - Construct matrix Q such that (ideally) $q_{ij}=1$ if i and j in same cluster, 0 otherwise
- Let V be matrix whose columns are first k eigenvectors of W
- Normalize rows of V to have unit Euclidean norm
  - Ideally each node (row) in one cluster (col)
- Let $Q=VV^T$
  - Each entry product of two unit vectors

Normalization and k Eigenvectors

- Normalized affinities help correct for variations in overall degree of affinity
  - So compute Q for N instead of W
- Contrasting Q with ratio of first two eigenvectors of N (ncut criterion)
  - More clearly selects most significant region
    - Using k=6 eigenvectors
  - Row of Q matrix vs. ratio of eigenvectors of N



Q N



Q N

Spectral Methods

- Eigenvectors of affinity and normalized affinity matrices
- Widely used outside computer vision for graph-based clustering
  - Link structure of web pages, citation structure of scientific papers
  - Often directed rather than undirected graphs

Mean Shift

- Used both for segmentation and for edge preserving filtering
- Operates on collection of points $X=\{x_1, \ldots, x_n\}$ in R^d
- Replace each point with value derived from mean shift procedure
  - Searches for a local density maximum by repeatedly shifting a d-dimensional hyper-sphere of fixed radius h
  - Differs from most hyper-sphere based clustering in that no fixed number of clusters

Mean Shift Procedure

- For given point $x \in X$ let $y_1, \ldots, y_T$ denote successive locations of that point

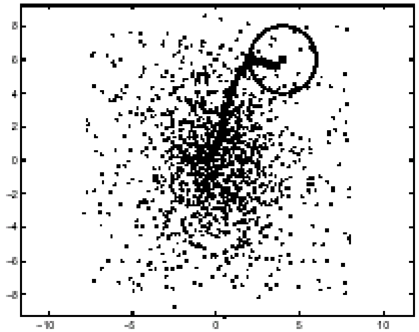
$$y_1 = x$$

$$y_{k+1} = 1/|S(y_k)| \sum_{x \in S(y_\mathbf{k})} x$$

  - Where $S(y_\mathbf{k})$ is the subset of X contained in a hyper-sphere of radius h centered at $y_\mathbf{k}$
    - The radius h is a fixed parameter of the method
- For a point set X, the mean shift procedure is applied separately to all the points

Illustration of Mean Shift

- Path of successive values of y_k for given starting point x



- Can be shown that converges to local density maximum

Mean Shift Image Filtering

- Map each image pixel to point in u,v,b space

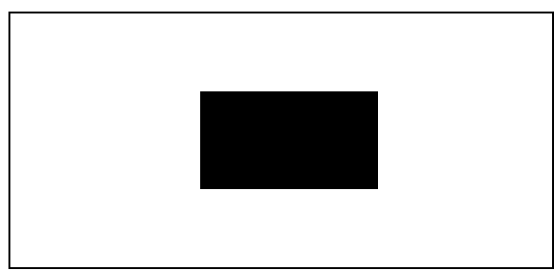
$$x_{\mathbf{i}}=(u_{\mathbf{i}},v_{\mathbf{i}},b_{\mathbf{i}}/\sigma)$$

  - Analogous for color images, with three intensity values instead of one
  - Scale factor σ normalizes intensity vs. spatial dimensions
- Perform mean shift for each point
  - Let $Y_{\mathbf{i}}=(U_{\mathbf{i}},Vi,B_{\mathbf{i}})$ denote mean shifted value
- Assign result $z_i=(u_i,v_i,B_i)$
  - Original spatial coords, mean shifted intensity

Mean Shift Example

Edge Preserving Filtering

- Mean shift tends to preserve edges
- Edges are where intensity is changing rapidly
- Rapid changes in intensity will result in lower density regions in joint spatial-intensity space
- Mean shift finds local density maxima

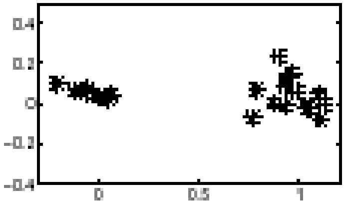


Mean Shift Clustering

- Run mean shift procedure for each point
- Cluster resulting convergence points that closer than some small constant
- Assign each point label of its cluster
- Analogous to filtering, but with added step of merging cluster that are nearby in the joint spatial-intensity domain

About Mean Shift

- Convergence to local density maximum
  - Where "local" determined by sphere radius
- Consider simple point set



- Over wide range of sphere radii end up with two clusters
  - Relationship to MST