CS 664
Segmentation

Daniel Huttenlocher

Cornell University
Faculty of Computing and Information Science

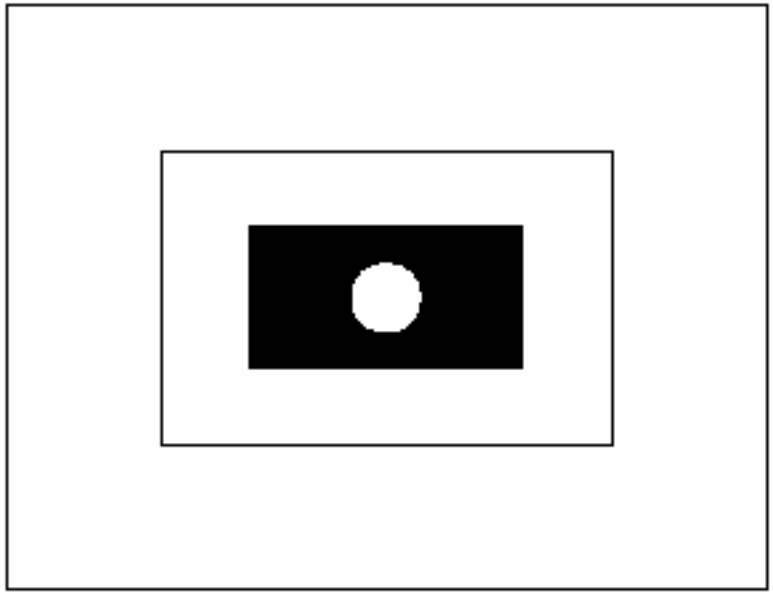
Perceptual Organization

- Grouping
  - Structural relationships between “tokens”
    - Parallelism, symmetry, alignment
    - Similarity of token properties
    - Often strong psychophysical cues
- Segmentation
  - Clustering pixels into regions
    - Generally contiguous in image, not always
  - Over-segmentation has proven useful, commonly termed super-pixels

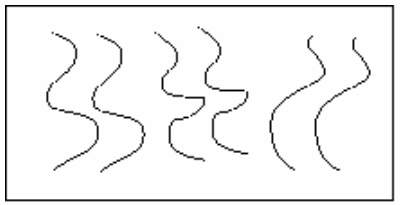
What's This Image?

Beyond "Figure/Ground"

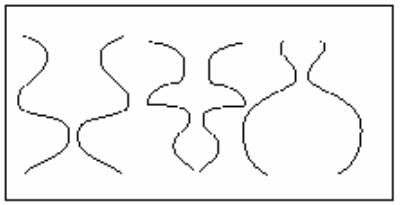
- Gestalt movement, properties and relations that form "percept as a whole"



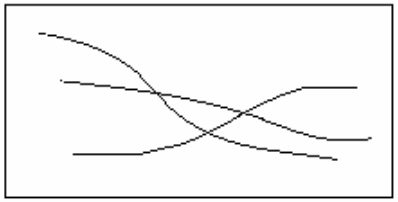
Important Structural Relations



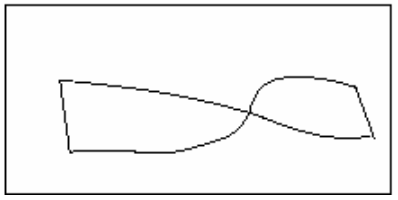
Parallelism



Symmetry



Continuity

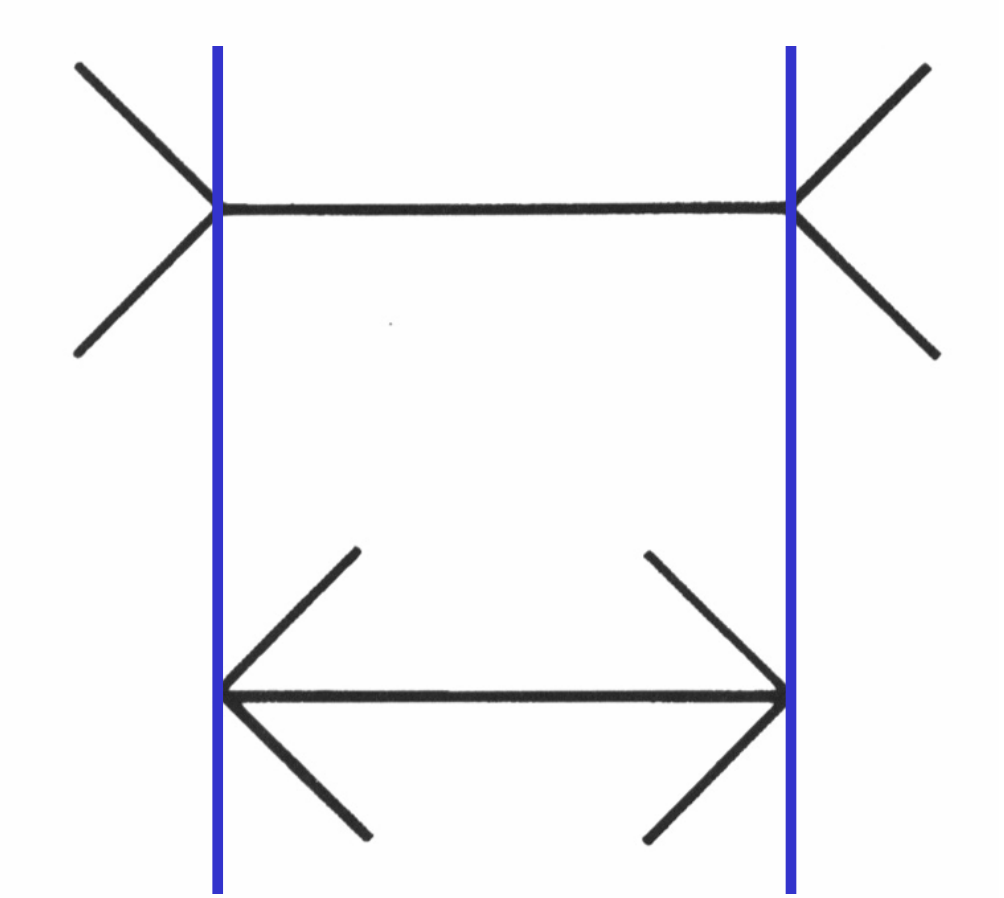


Closure

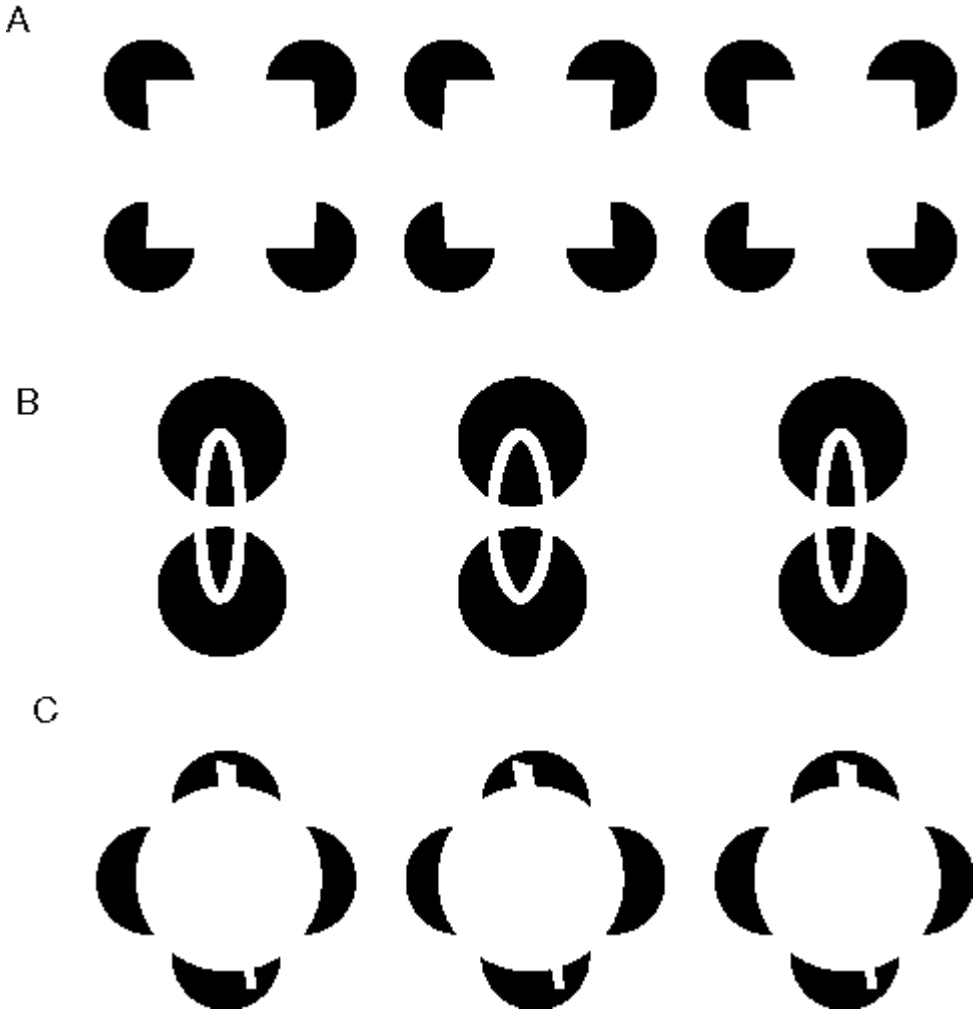
Occluding Contours and Percept

Importance of Context to Percept

- Famous Muller-Lyer Illusion



Grouping: Illusory Contours

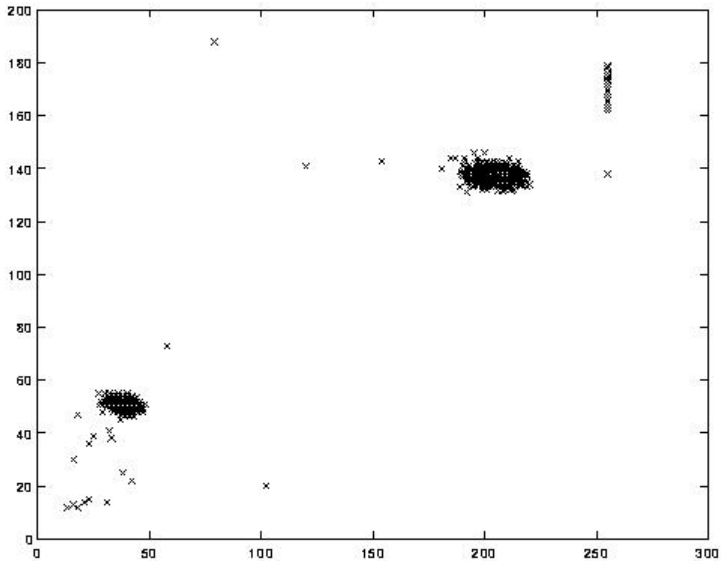


Figure/Ground Separation

- Find foreground by subtracting out previously obtained background image
- In practice background usually not stationary, so simple subtraction not good
- Model background
  - For fixed camera, Gaussian intensity model for each pixel has proven effective [Stauffer&Grimson]
    - Test fit of observed pixel value to distribution
    - Update background distribution at some slow rate, for inliers

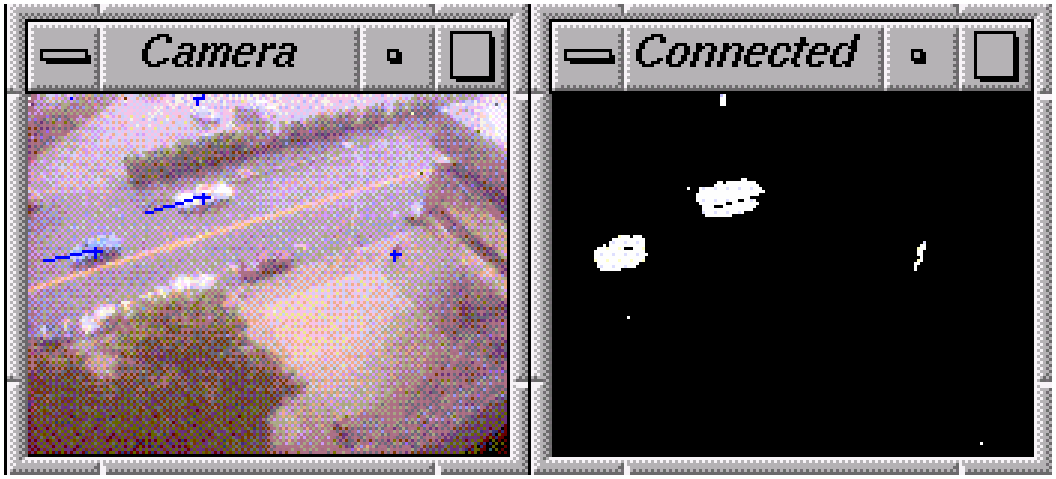
Non-Stationary Pixels

- Trees, bushes, water, flags, …
- Changes in lighting over time
- E.g., bimodal distribution of intensity at a given “water” pixel over a few minutes



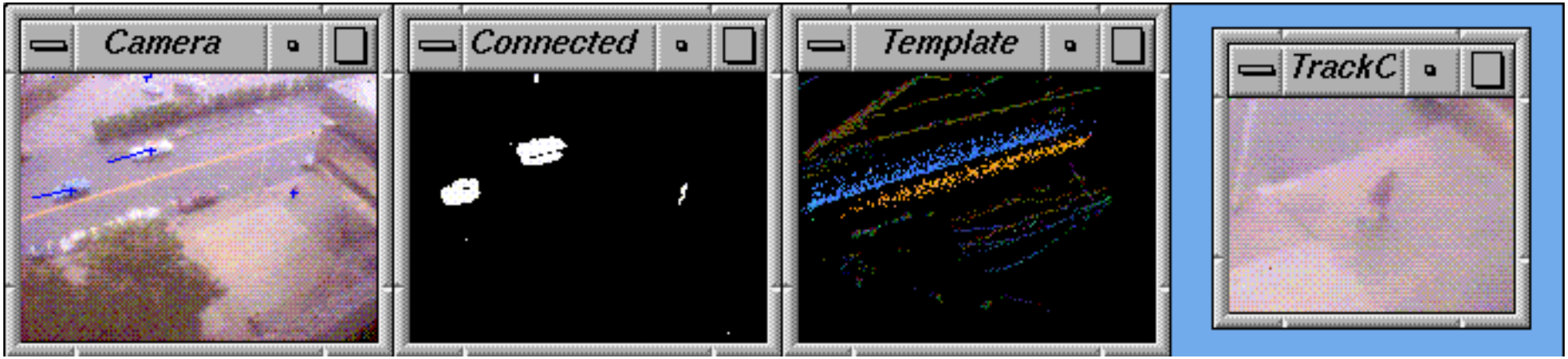
Background Modeling

- Binary image of outlier pixels from Gaussian background models
  - Small differences reliable enough to often correspond to actual objects
    - E.g., pedestrians in this scene



Simple Tracker

- Determination of foreground/background in this manner yields data good for simple tracking
  - E.g., Kalman filter for each object estimating position and velocity vector

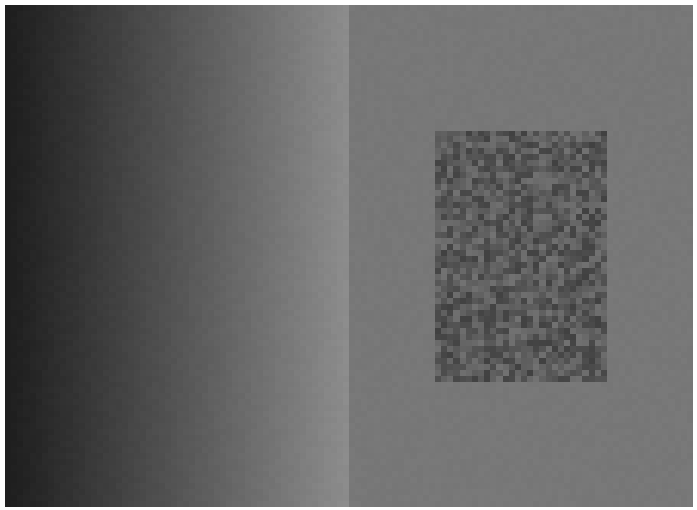


Segmenting Single Images

- 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

Motivating Example

- Coherent regions independent of particular objects or recognition

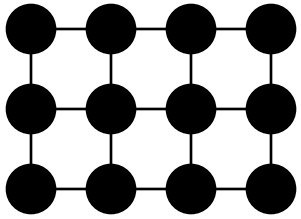


This image has three perceptually distinct regions

Where are largest intensity differences?

Graph Based Formulations

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



4-connected or 8-conneted

- Weight of edge is measure of difference (or affinity) between connected pixels
- A *segmentation*, S, is a partition of V such that each $C \in S$ is connected

Forms of Affinity Measure

- Intensity

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_i^2\right)\left(\|I(x) - I(y)\|^2\right)\right\}$$

- Distance

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x - y\|^2\right)\right\}$$

- Texture

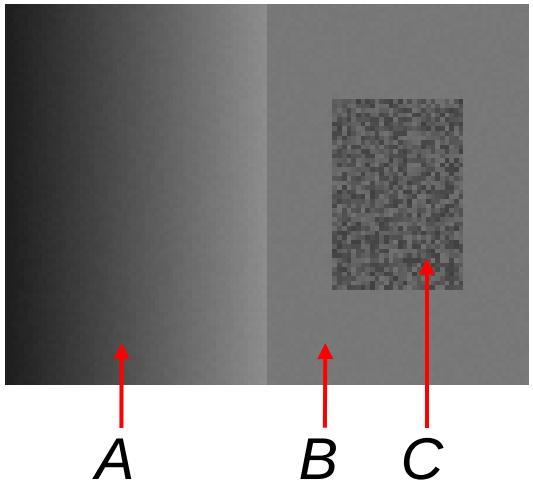
$$aff(x,y) = \exp\left\{-\left(1/2\sigma_t^2\right)\left(\|c(x) - c(y)\|^2\right)\right\}$$

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

Measure Whole Components

- Consider properties of two components being merged when adding an edge [Felzenszwalb 04]
  - Rather than MST based on local edge weights
- 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

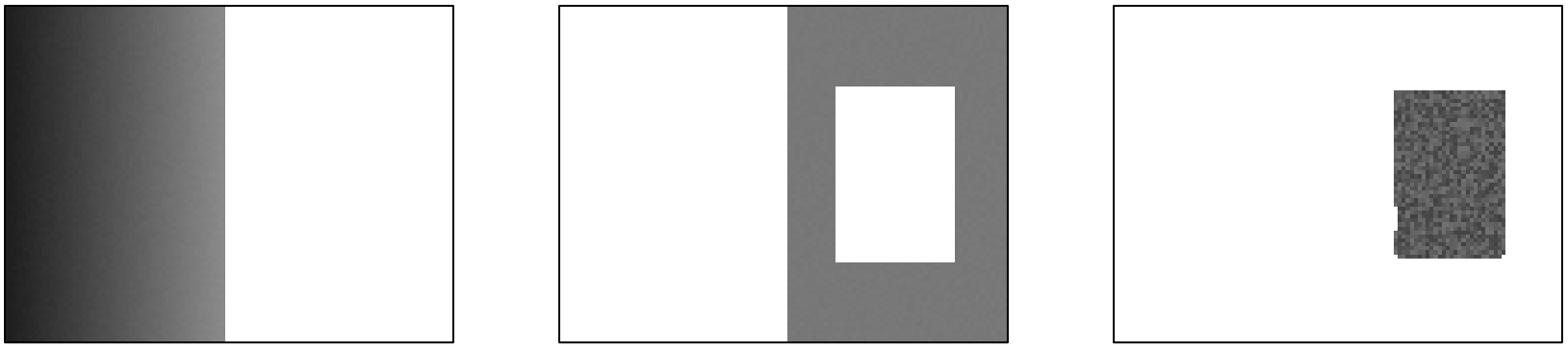
- Intuitively $Int(C)$ estimates local differences over component
  - Small components give underestimate of local difference – neighboring pixels tend to be similar
    - Thus τ should be large in this case
- Use a function inversely proportional to component size $\tau(C) = k / |C|$
  - k is a parameter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

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*

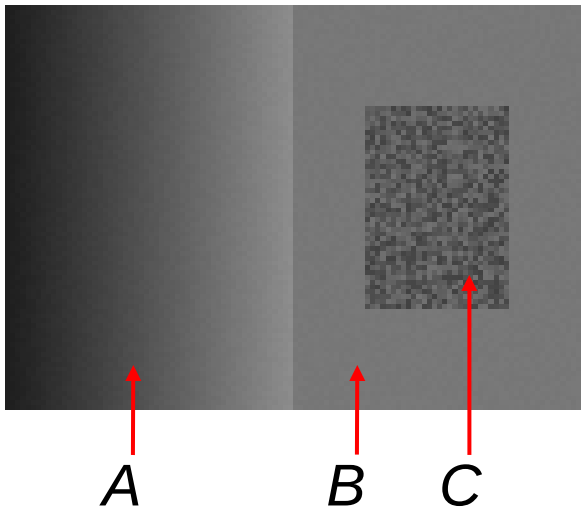
- Three main regions plus a few small ones
- Why the algorithm stops growing these
  - Weight of edges between A and B large wrt max weight MST edges of A and of B
  - Weight of edges between B and C large wrt max weight MST edge of B (but not of C)

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

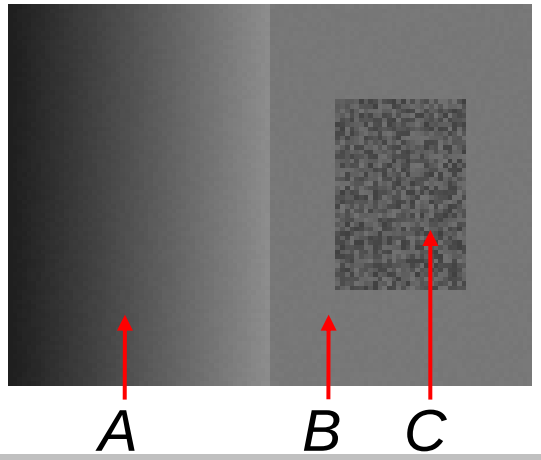
Good Segmentations and the Example

- 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"

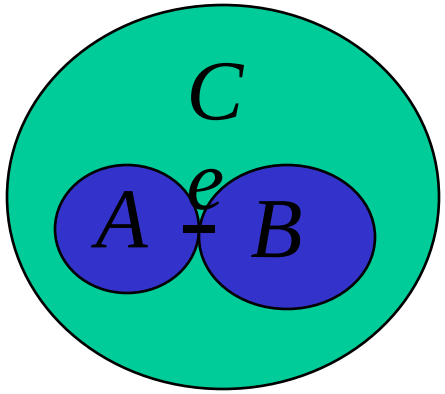
- When two components do not merge, one will be a component of the final segmentation
  - A merge decision is made for an edge e_q and the two components that it connects C_i, C_j
  - Say the merge does not occur because $w(e_q) > Int(C_i) + \tau(C_i)$
    - Then any subsequent merge involving C_i will 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

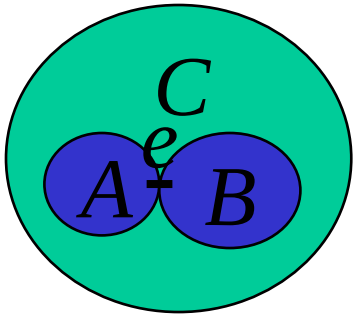
Segmentation Not Too Coarse

- 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

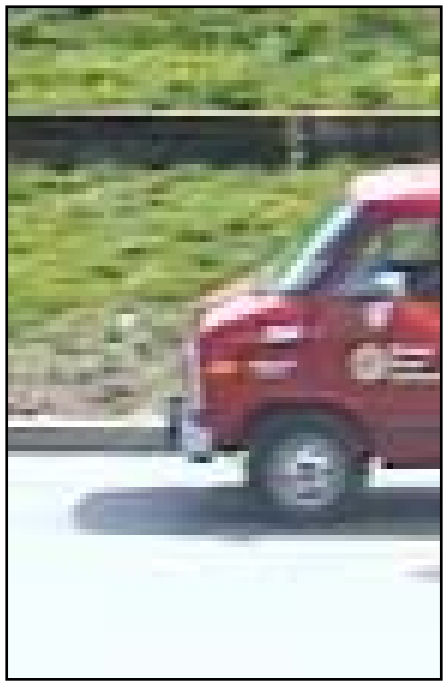
Some Example Segmentations

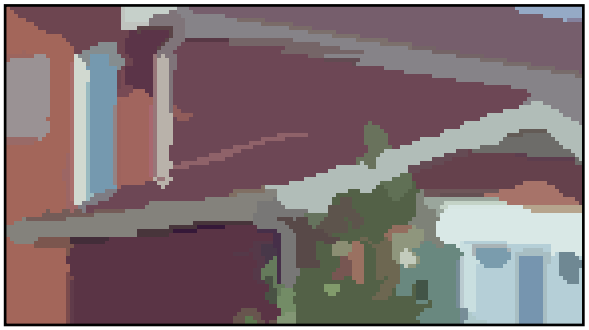
k=300
320 components
larger than 10

k=200
323 components
larger than 10

Some Shortcomings

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

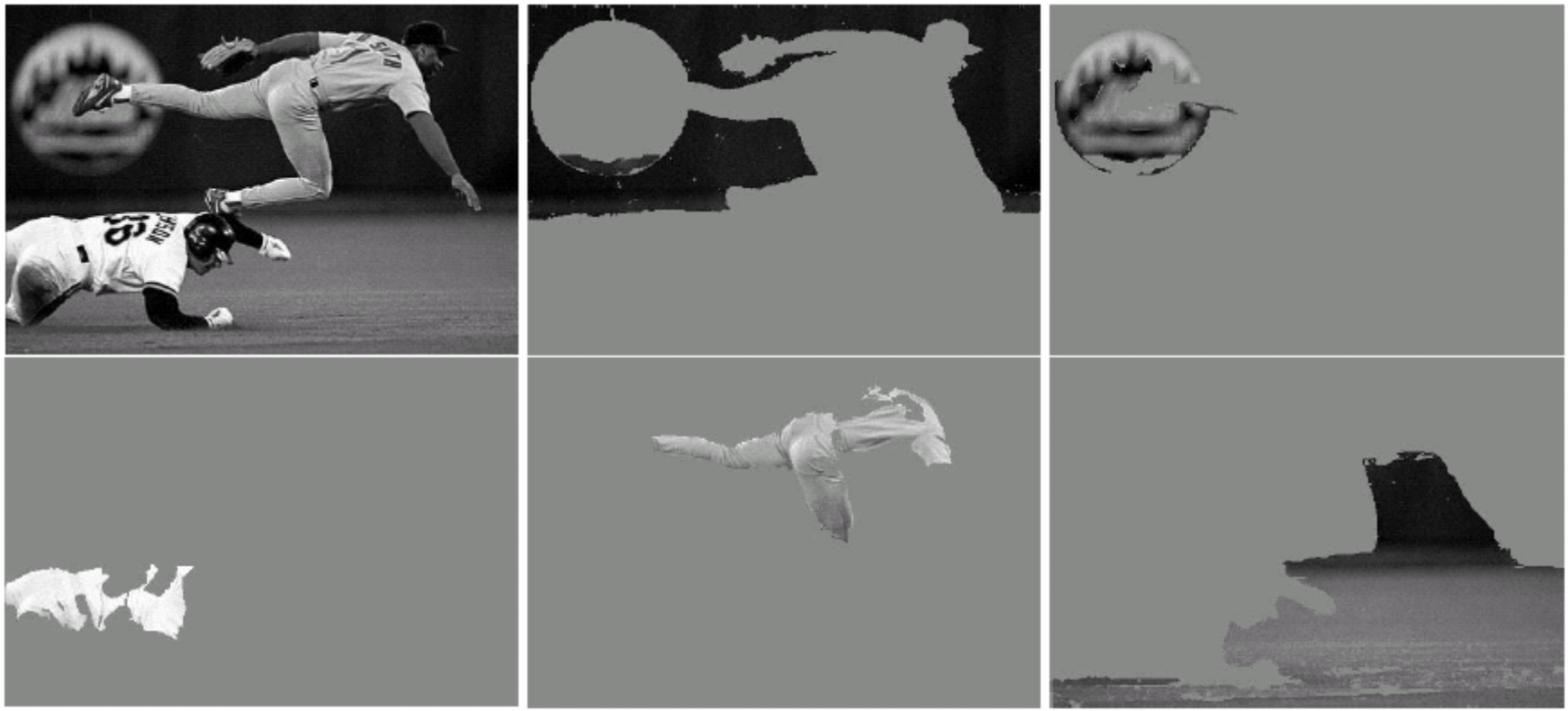


Simple Object Examples

Monochrome Example

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

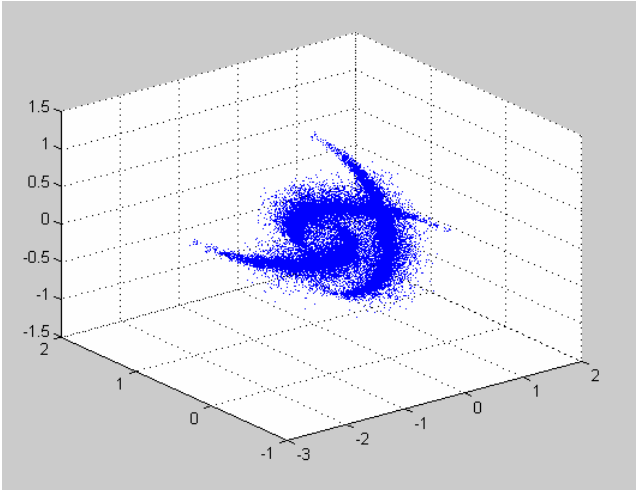


Clustering: Non-Local Components

- 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

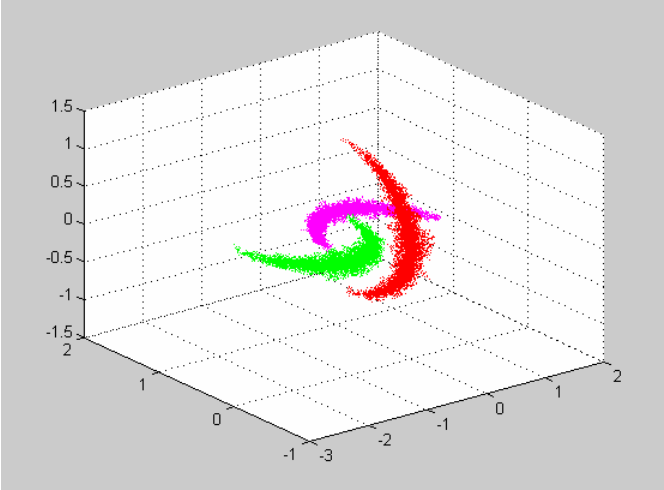
Clustering Gaussian Point Data

Note: Gaussian not constant density

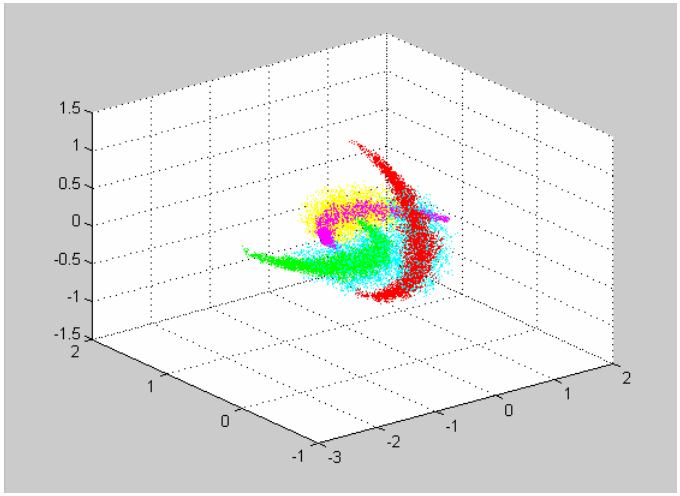


Graph connecting four nearest neighbors to each vertex

$k = 1$



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

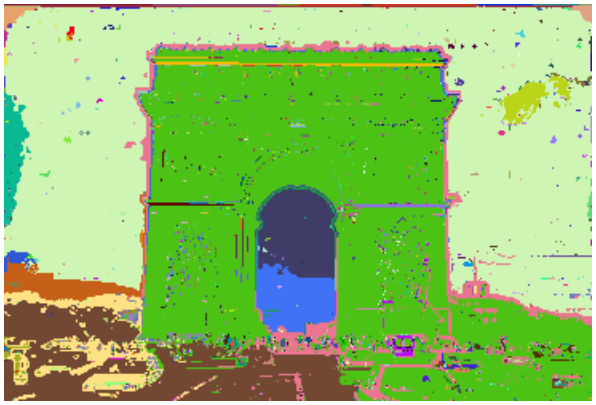
Example Clustering of Image Data

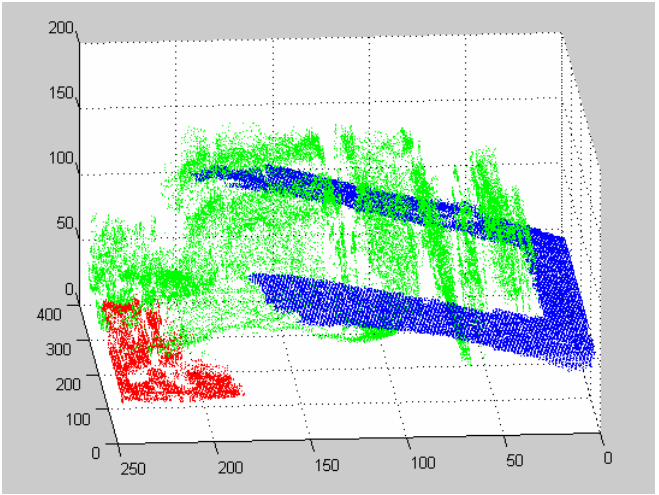
- Segmentation using difference in R,G,B values and in position
  - Distance of 5 pixels same as 1 intensity unit

Non-Local Component

About Clustering for Image Data

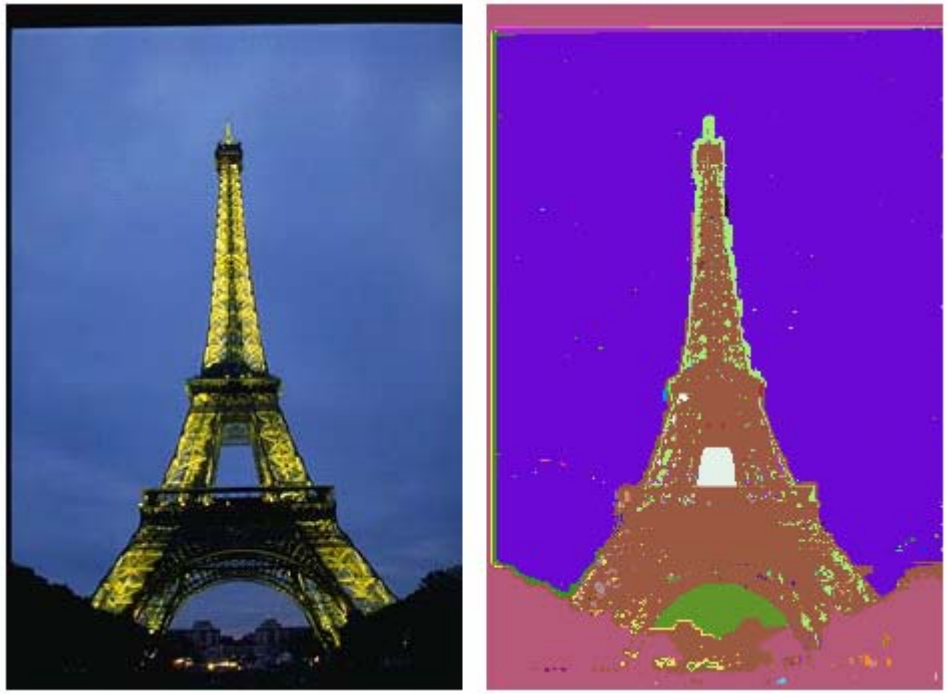
- Meaningful regions in image are not necessarily compact in feature space
- Cheap path in feature space not always apparent in image





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