Computer Vision

18AI742

Dr. rer. nat. D. Antony Louis Piriyakumar,
Dean (R&D),
Cambridge Institute of Technology.

BRIDGE TO THE FUTURE

CAMBRIDGE
INSTITUTE OF TECHNOLOGY

Segmentation by clustering

Dr. D. Antony Louis Piriyakumar,
*Dean (*Research & Development)
Registered Indian patent agent (IN/PA 3041)

www.cambridge.edu.in

Segmentation by clustering

Contents

$$\nabla \cdot \mathbf{E} = \rho/\varepsilon_0$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \boldsymbol{j}_c$$

where

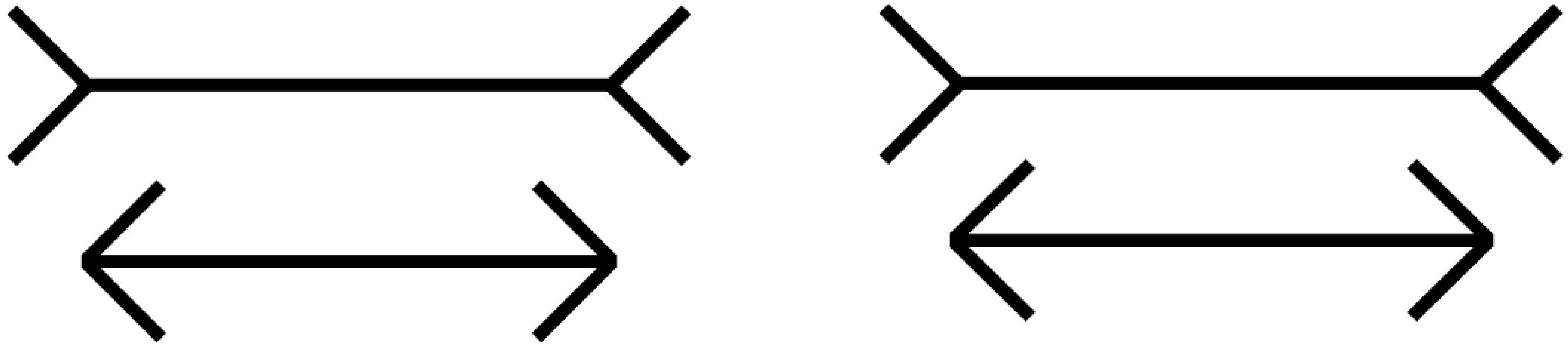
$$\nabla = \hat{\mathbf{i}} \frac{\partial}{\partial x} + \hat{\mathbf{j}} \frac{\partial}{\partial y} + \hat{\mathbf{k}} \frac{\partial}{\partial z}$$

Human vision: grouping and gestalt

- constructing groups of pixels that all have the same color or texture
- collecting together pixels or pattern elements into summary representations
- early vision produces vast quantities of information
- That emphasize important, interesting, or distinctive properties.
- Obtaining such representation is known variously as segmentation

How sharp is your eye?

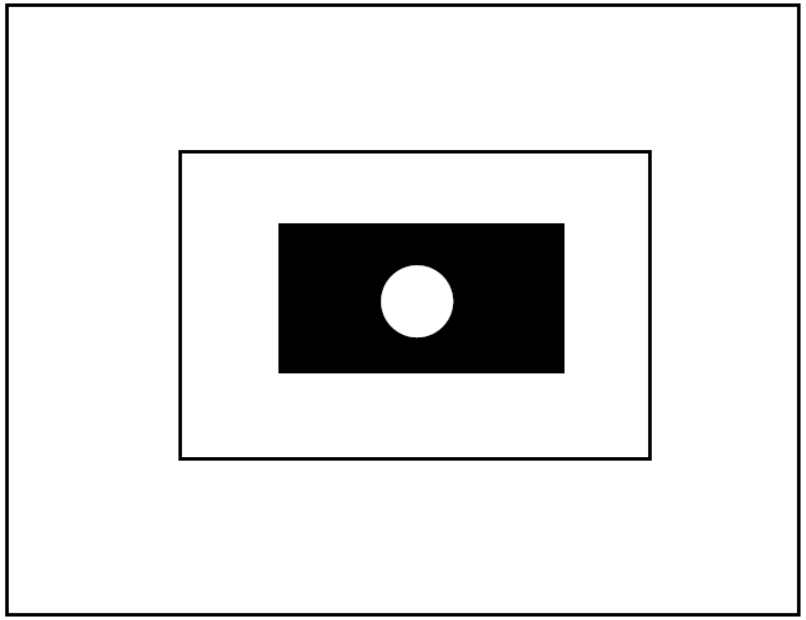
Mueller-Lyer illusion

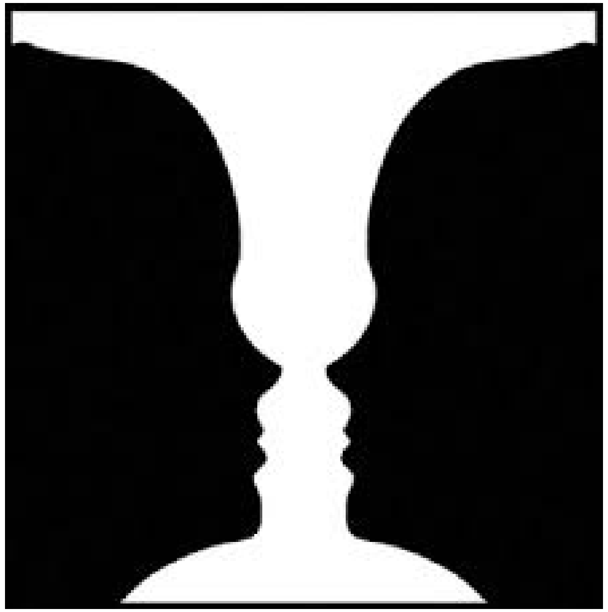


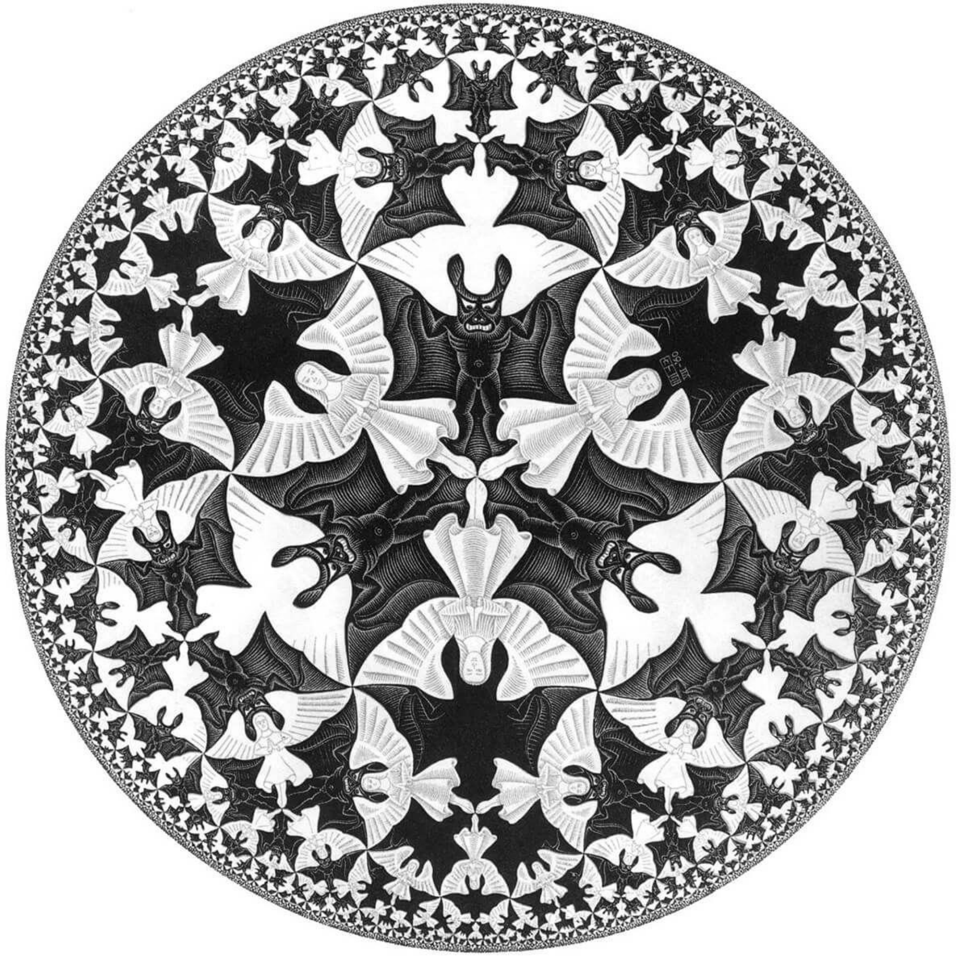
Two important threads in segmentation

- clustering methods that focus on local relations between items
- assembling together clumps (regions) of pixels that look similar
- assembling together items based on global relations
- Eg - all items that lie on a straight line
- emphasizing methods that can identify parametric models in pools of data

Figure and ground - ambiguity







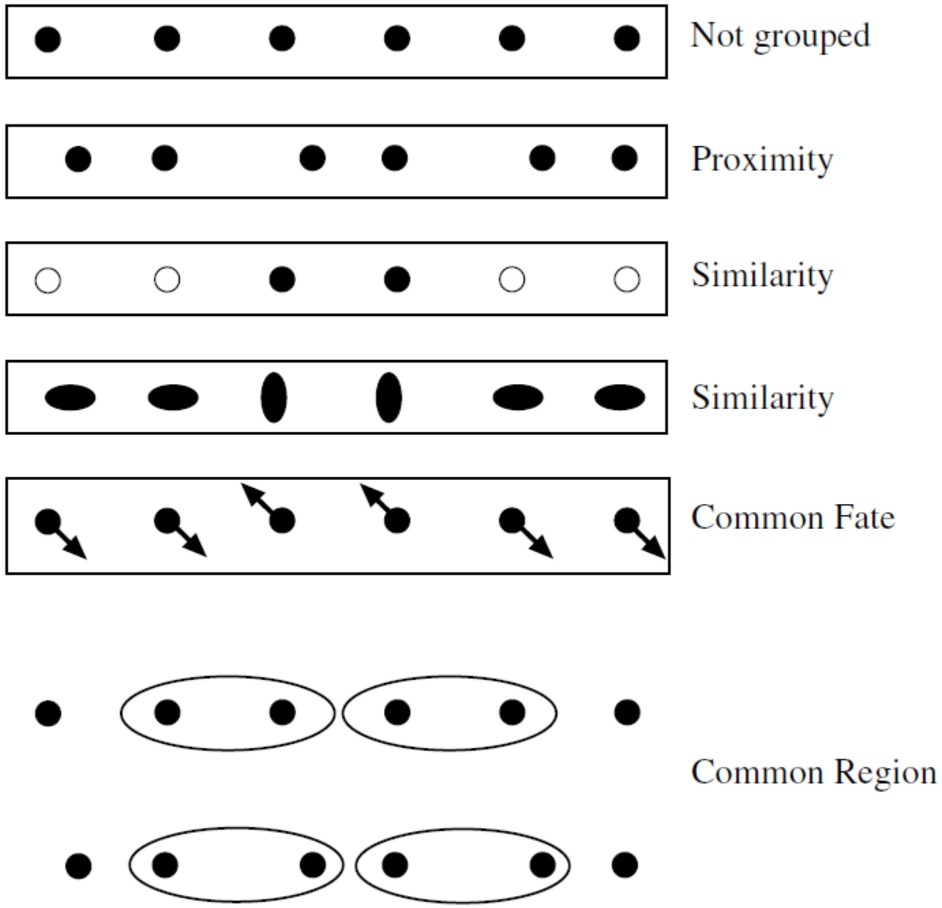
Gestalt/group set of internal relationships

- Proximity: Tokens that are nearby tend to be grouped.
- Similarity: Similar tokens tend to be grouped together.
- Common fate: Tokens that have coherent motion tend to be grouped
- Common region: Tokens that lie inside the same closed region tend
- Parallelism: Parallel curves or tokens tend to be grouped together

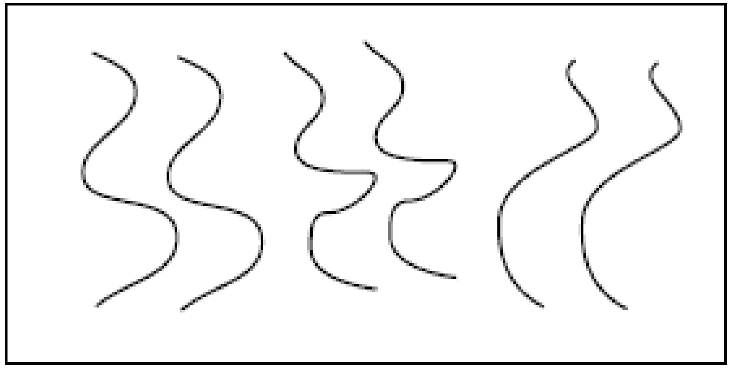
Gestalt/group set of internal relationships

- Closure: Tokens or curves that tend to lead to closed curves tend
- Symmetry: Curves that lead to symmetric groups are grouped together
- Continuity: Tokens that lead to continuous—as in joining up nicely
- Familiar configuration: Tokens that, when grouped, lead to a familiarobject tend
- Parallelism: Parallel curves or tokens tend to be grouped together

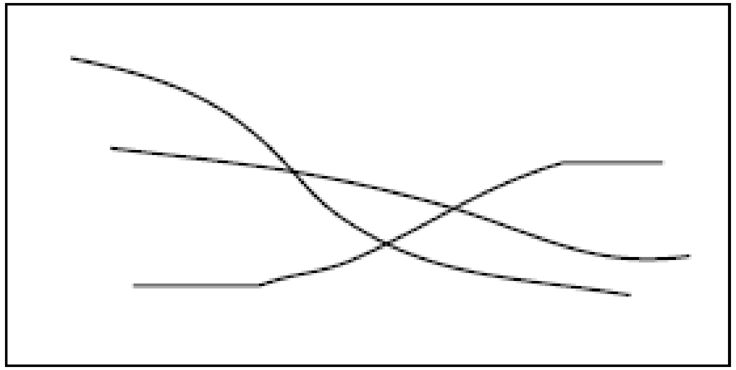
Examples of Gestalt factors that lead to grouping



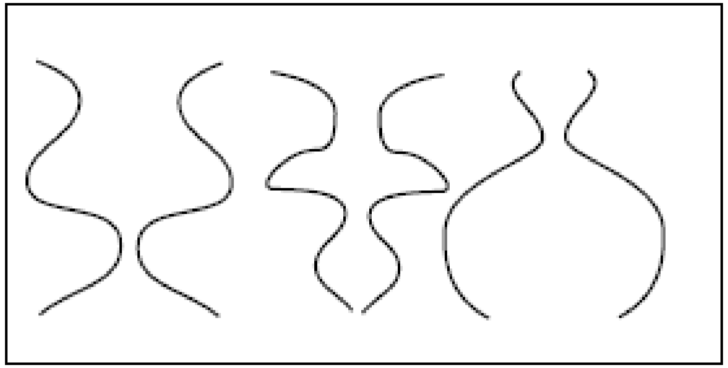
Examples of Gestalt factors that lead to grouping



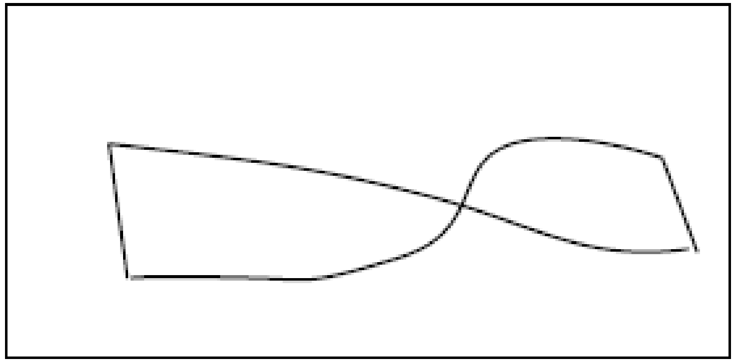
Parallelism



Continuity

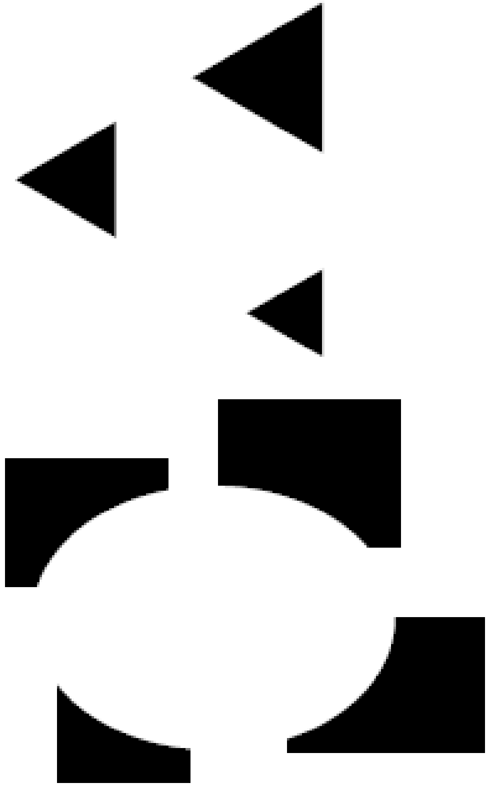


Symmetry

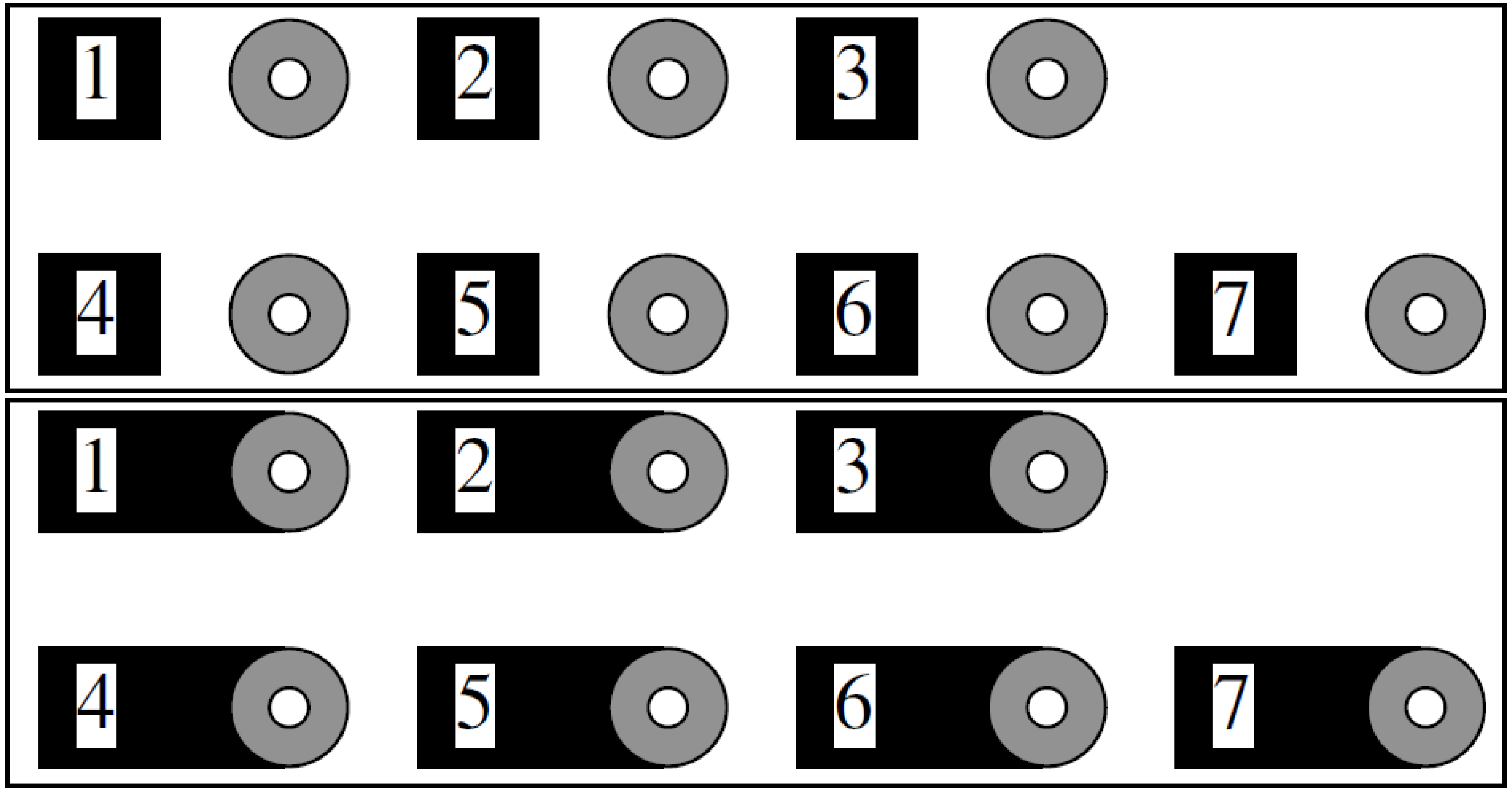


Closure

Occlusion appears to be an important cue in grouping



An example of grouping phenomena in real life



Gestalt rules

- rules help solve problems posed by visual effects that arise commonly in real world
- ecologically valid
- continuity may represent a solution to problems posed by occlusion
- tendency to prefer interpretations explained by occlusion leads to interesting effects.
- illusory contour (one could fill in the no-contrast regions of contour)

Gestalt rules

- Common fate can be seen as a consequence of the fact
- that components of objects tend to move together
- symmetry is a useful grouping cue because
- lot of real objects that have symmetric or close to symmetric contours.
- should be seen as the consequences of a larger grouping process

Segmentation by clustering

Contents

1) Human vision: grouping and gestalt
2) Important applications
3) Image segmentation by clustering pixels
4) Segmentation, clustering, and graphs
5) Conclusion
6) Q&A

$$\nabla \cdot \mathbf{E} = \rho/\varepsilon_0$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \boldsymbol{j}_c$$

where

$$\nabla = \hat{\mathbf{i}} \frac{\partial}{\partial x} + \hat{\mathbf{j}} \frac{\partial}{\partial y} + \hat{\mathbf{k}} \frac{\partial}{\partial z}$$

Simple segmentation algorithms

- work best when it is easy to tell what a useful decomposition is.
- background subtraction
- where anything that doesn’t look like a known background is interesting
- shot boundary detection
- Where substantial changes in a video are interesting

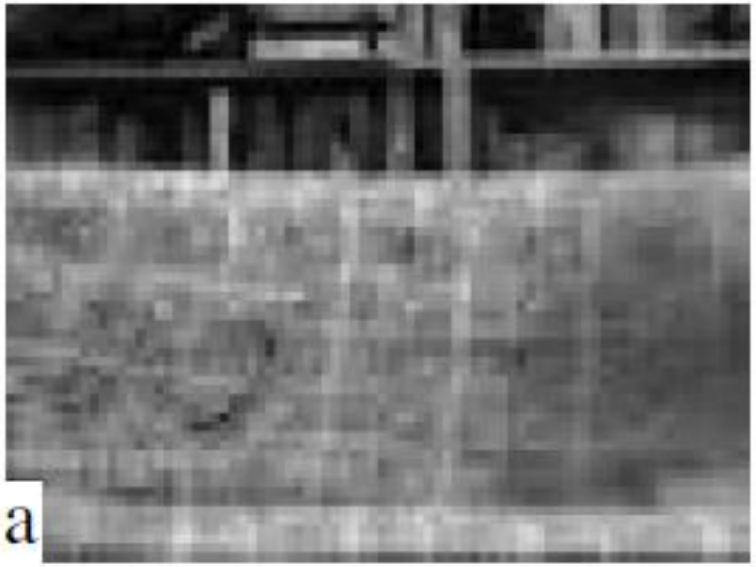
CAMBRIDGE
INSTITUTE OF TECHNOLOGY

Background subtraction

- objects appear on a largely stable background
- Counting vehicles on road
- Cine green screen
- estimate of the appearance of the background from the image
- works rather poorly because the background typically changes slowly over time

Every fifth frame from a sequence of 120 frames of a child playing on a patterned sofa.

Background subtraction results for the sequence



The average of all 120 frames.

Pixels whose difference from the average exceeds a small threshold

Those whose difference from the average exceeds
a somewhat larger threshold.

Estimate the value of background pixels using a moving average

Form a background estimate $\mathcal{B}^{(0)}$. At each frame $\mathcal{F}$

Update the background estimate, typically by

forming $\mathcal{B}^{(n+1)} = \frac{w_a \mathcal{F} + \sum_i w_i \mathcal{B}^{(n-i)}}{w_c}$

for a choice of weights w_a, w_i and w_c.

Subtract the background estimate from the

frame, and report the value of each pixel where

the magnitude of the difference is greater than some

threshold.

end

Algorithm 9.1: Background Subtraction.

Estimate the value of background pixels using a moving average

Estimate the value of background pixels using a moving average

- Estimate particular background pixel as a weighted average of the previous values
- pixels in the distant past should be weighted at zero
- weights increase smoothly
- moving average should track the changes in the background
- suppress frequencies that are larger than the typical frequency of change in the background

Shot boundary detection

```
For each frame in an image sequence
    Compute a distance between this frame and the
      previous frame
    If the distance is larger than some threshold,
      classify the frame as a shot boundary.
end
```

Algorithm 9.2: Shot Boundary Detection Using Interframe Differences.

Standard techniques for computing a distance

- Frame differencing algorithms take pixel-by-pixel differences
- Histogram-based algorithms compute color histograms for each frame
- Block comparison algorithms compare frames by cutting them into a grid
- Edge differencing algorithms compute edge maps for each frame
- few potentially corresponding edges,there is a shot boundary

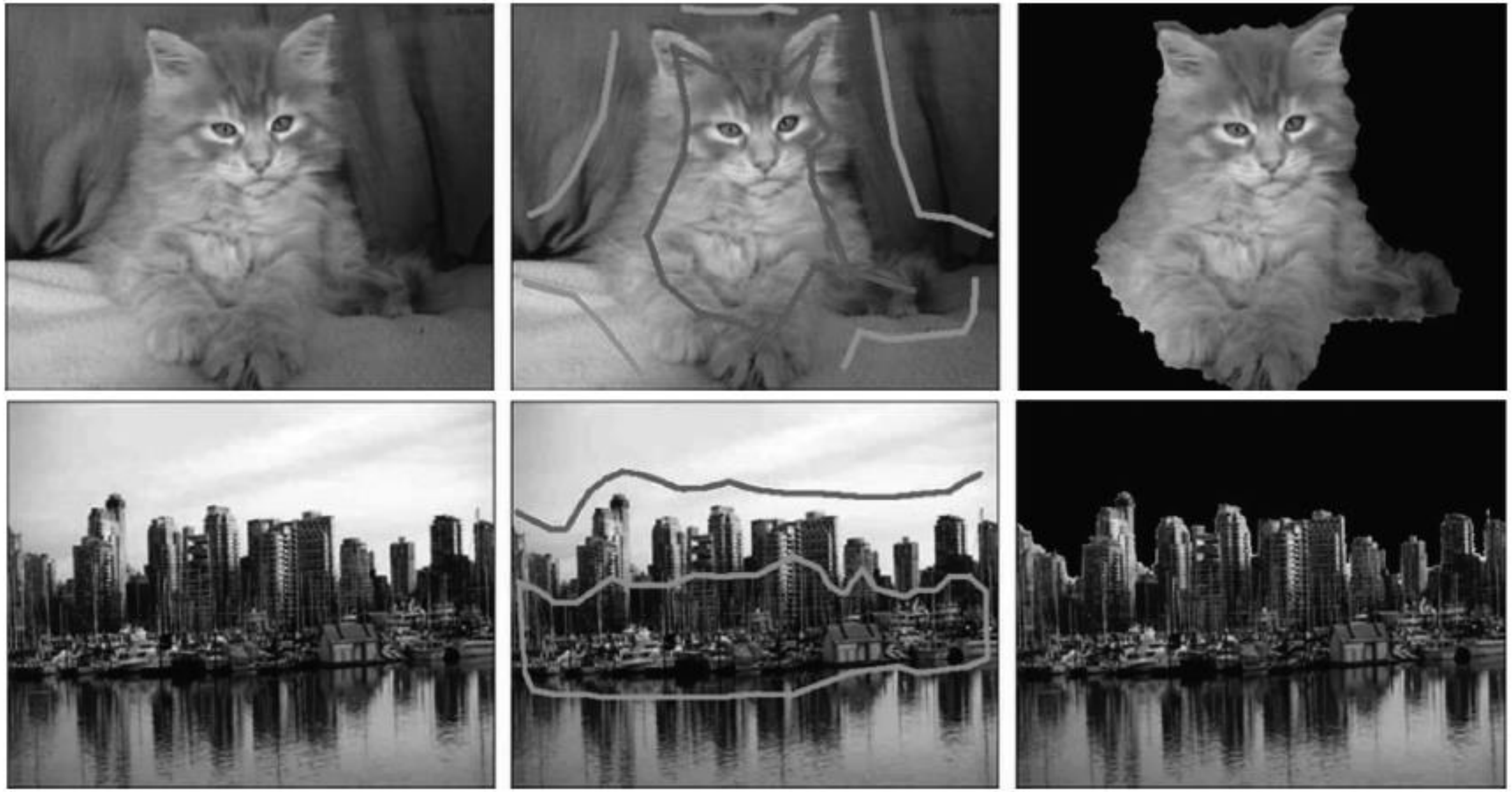
Interactive segmentation

- two segments, foreground (coherent) and background,
- intelligent scissors interface
- user sketches a curve fairly close to the boundary of the object
- curve is then moved to the boundary using local information, image gradient cues
- need good ways to select the object we want to cut out

Interactive segmentation

- painting interface
- user paints some pixels with a foreground or background brush
- to produce an appearance model of the foreground and of the background.
- models are fed into a fast graph-based segmenter

Interactive Segmentation (painting interface)



Interactive segmentation (grabcut interface)

- grabcut interface
- user draws a box around the object
- box yield an initial estimate of foreground and background pixels,.
- If this segmentation isn’t satisfactory
- user has the option of painting foreground and background strokes on pixels

Interactive segmentation (grabcut interface)

FIGURE 9.12: In a grabcut interface for interactive segmentation, a user marks a box around the object of interest; foreground and background models are then inferred by a clustering method, and the object is segmented. If this segmentation isn't satisfactory, the user has the option of painting foreground and background strokes on pixels to help

pixels are neither pure background or pure foreground

- prepare a matte, a mask of values in the range [0–1].
- ith pixel value is that it is $\alpha f + (1 - \alpha)b$,
- Rotoscoping is a process like matting, but applied to video;
- recovers a set of segments, one per frame, corresponding to a moving object.
- segments could then becomposited onto a new background

Matting methods produce a real-valued mask

FIGURE 9.13: Matting methods produce a real-valued mask (rather than a foreground-background mask) to try and compensate for effects in hair, at occluding boundaries, and so on, where some pixels consist of an average of foreground and background values. The matte is bright for foreground pixels and dark for background pixels; for some pixels in the hair, it is gray, meaning that when the foreground is transferred to a new image, these pixels should become a weighted sum of foreground and background. The gray value indicates the weight. *This figure was originally published as Figure 6 of "Spectral*

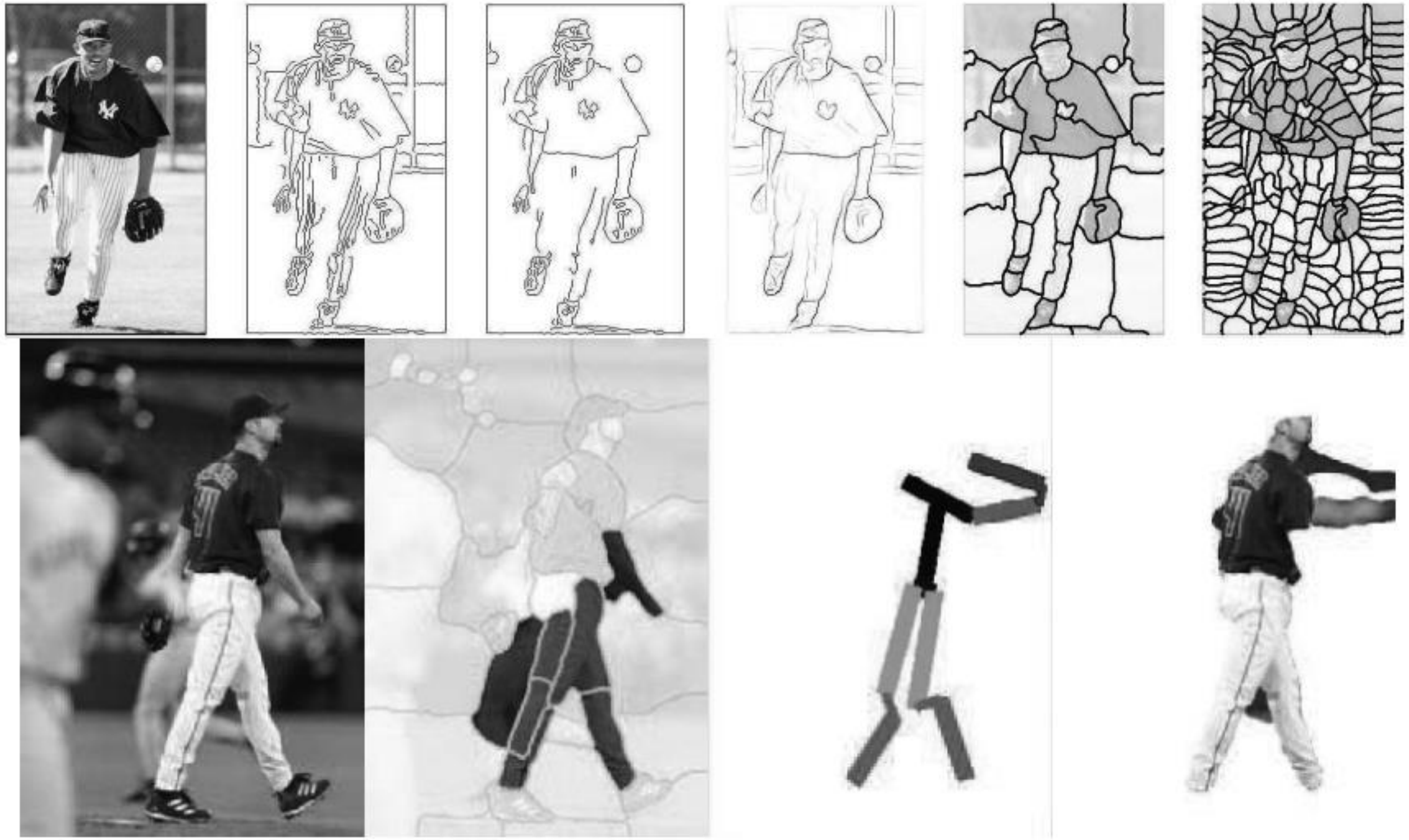
Forming Image Regions

- decompose an image into regions that have roughly coherent color and texture
- Because each region has coherent appearance, able to compress the image
- describing the shape and appearance of regions separately (asopposed to describing each pixel)
- correspondences between two images—to compute optic flow or to register parts of the images
- windows on the facade of a building—not quite a texture, but still repetitious.

Forming Image Regions (superpixels)

- more useful to have small, compact regions – Superpixels
- Need representation that is small compared to the pixel grid, but still very rich
- representation is sometimes called an oversegmentation
- one shading value per superpixel, and smooth the result.
- human arms and legs tend to be long and straight;

Forming Image Regions (superpixels)



Segmentation by clustering

Contents

1) Human vision: grouping and gestalt
2) Important applications
3) Image segmentation by clustering pixels
4) Segmentation, clustering, and graphs
5) Conclusion
6) Q&A

$$\nabla \cdot \mathbf{E} = \rho/\varepsilon_0$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \boldsymbol{j}_c$$

where

$$\nabla = \hat{\mathbf{i}} \frac{\partial}{\partial x} + \hat{\mathbf{j}} \frac{\partial}{\partial y} + \hat{\mathbf{k}} \frac{\partial}{\partial z}$$

General recipe for image segmentation by clustering

- represent each image pixel with a feature vector (FV)
- FV contains all measurements that may be relevant in describing a pixel
- Intensity, colour, texture, filter output
- Cluster these FVs. Every FV belongs to exactly one cluster
- each cluster represents an image segment

CAMBRIDGE
INSTITUTE OF TECHNOLOGY

General recipe for image segmentation by clustering

- replace the FV at each pixel with the number of that feature vector’s cluster center
- No guarantee that segments are connected,
- If the feature vector contains a representation of the position of the pixel,
- segments that result tend to be “blobby,”
- pixels very far from the center of a segment wil tend to belong to other clusters

Two natural algorithms for clustering

- divisive clustering
- Entire data set is regarded as a cluster
- clusters are recursively split to yield a good clustering
- agglomerative clustering
- each data item is regarded as a cluster
- clusters are recursively merged to yield a good clustering

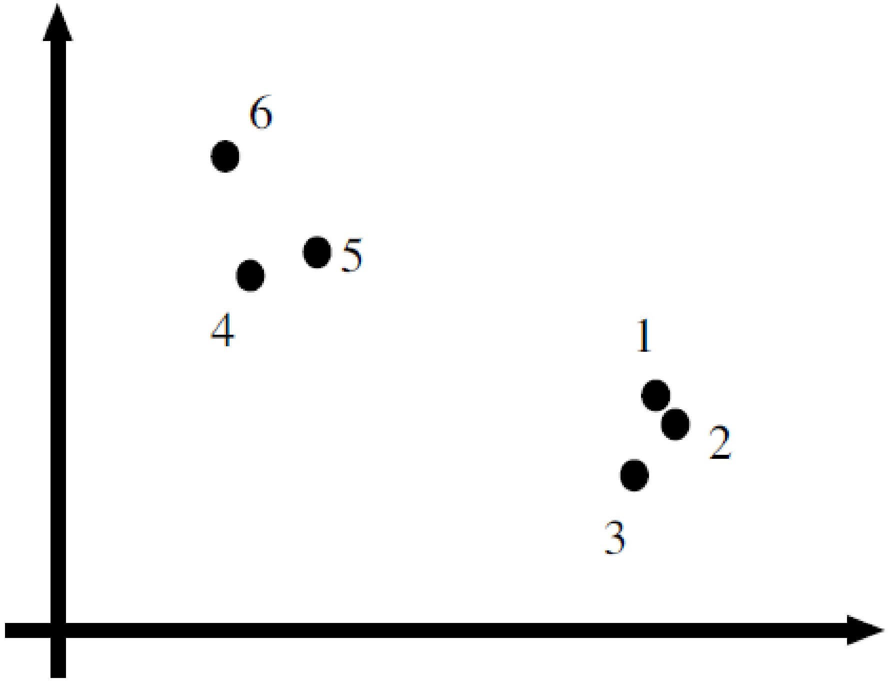
Divisive Clustering, or Clustering by Splitting.

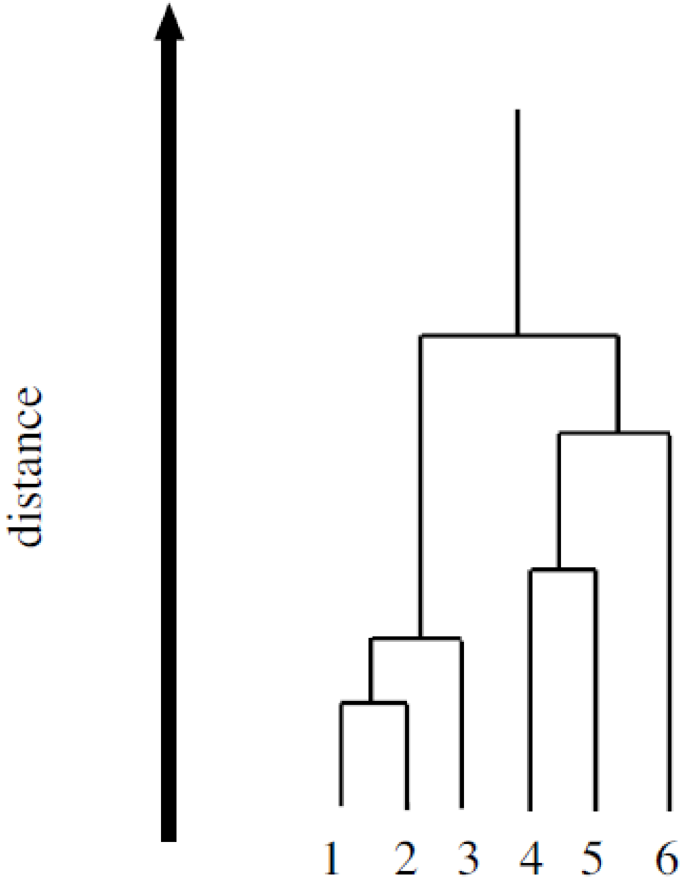
```
Construct a single cluster containing all points
Until the clustering is satisfactory
      Split the cluster that yields the two
         components with the largest inter-cluster distance
end
```

Agglomerative Clustering or Clustering by Merging.

```
Make each point a separate cluster
Until the clustering is satisfactory
      Merge the two clusters with the
         smallest inter-cluster distance
end
```

Dendrogram obtained by agglomerative clustering using single-link clustering.





Good inter-cluster distance

- single-link clustering
  - Distance between the closest elements as the inter-cluster distance
- complete-link clustering
  - maximum distance between an element of the first cluster and one of the second
- group average clustering
  - average of distances between elements in the cluster, “rounded” clusters

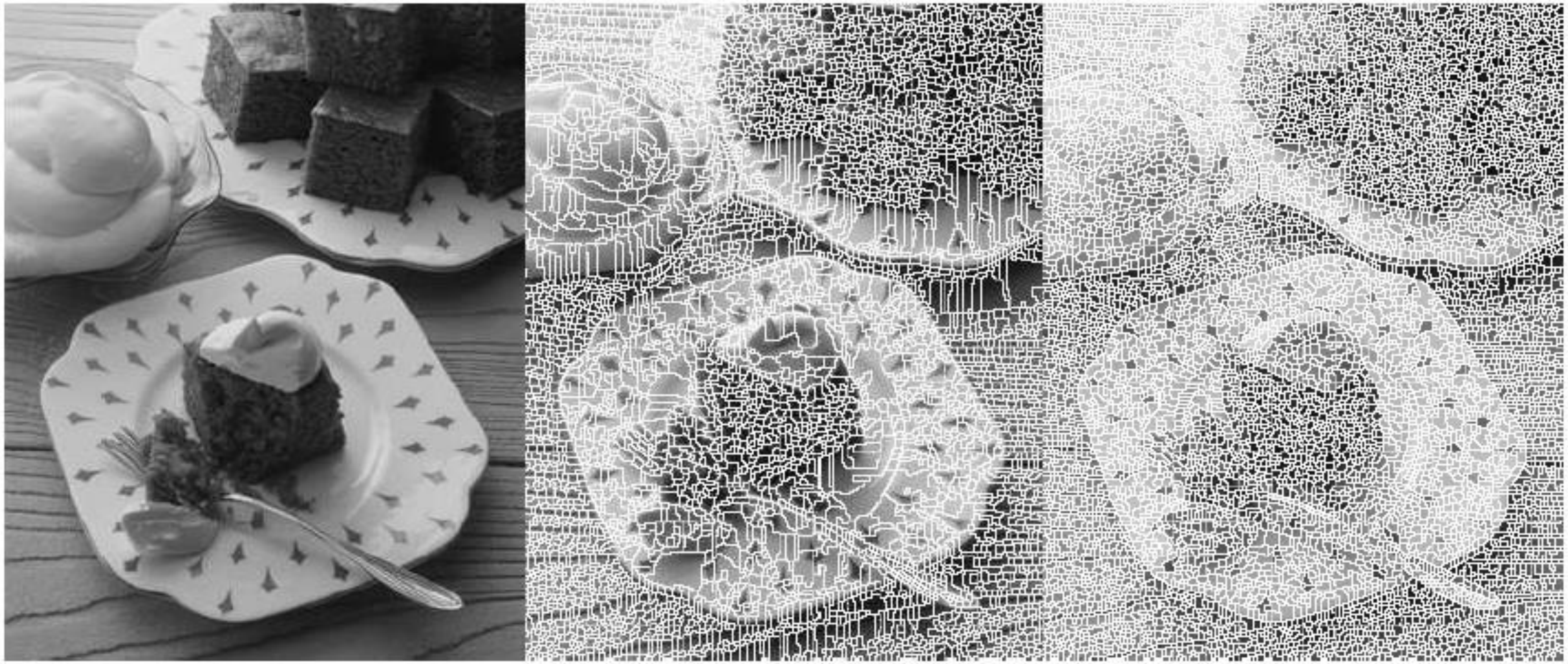
Main difficulty in using either agglomerative or divisive clustering

- an awful lot of pixels in an image
- segmenters decide when to stop splitting or merging by using a set of threshold
- agglomerative segmenter might stop merging when the distance between
- Clusters is sufficiently low or when the number of clusters reaches some value.
- stop splitting when the resulting clusters meet some similarity test

Watershed Algorithm

- to segment image I. compute a map of the image gradient magnitude, $||\nabla I||$
- Zeros of this map are locally extreme intensity values
- take each as a seed for a segment and give each seed a unique label
- filling a height map with water
- starting at pixel (i, j); travel backward down the gradient of $||\nabla I||$,
- Each pixel gets the label of the seed that is hit

Segmentation results from the watershed algorithm

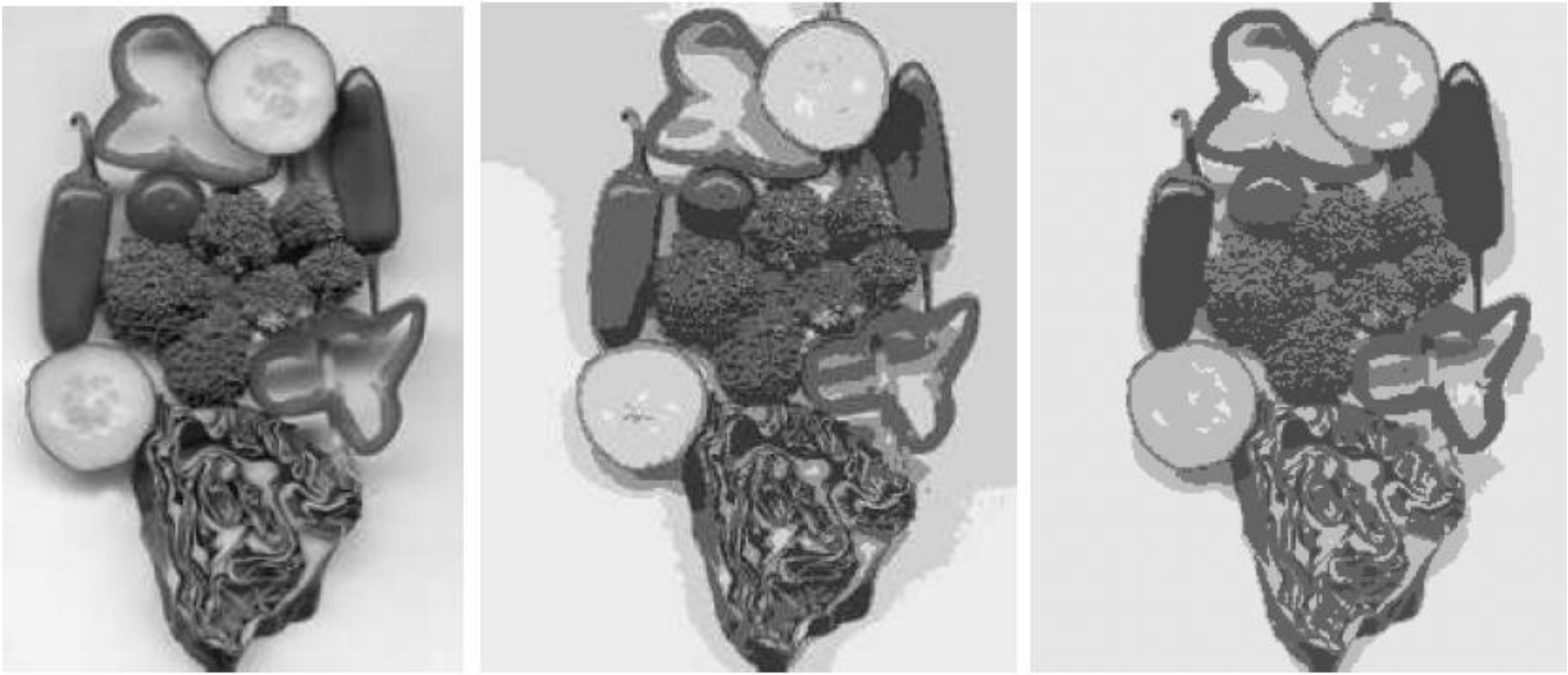


CAMBRIDGE INSTITUTE OF TECHNOLOGY

Segmentation Using K-means

- compute a feature vector representing each pixel
- each pixel goes to the segment represented by the cluster center
- that claims its feature vector.
- we know how many segments there will be
- stop splitting when the resulting clusters meet some similarity test

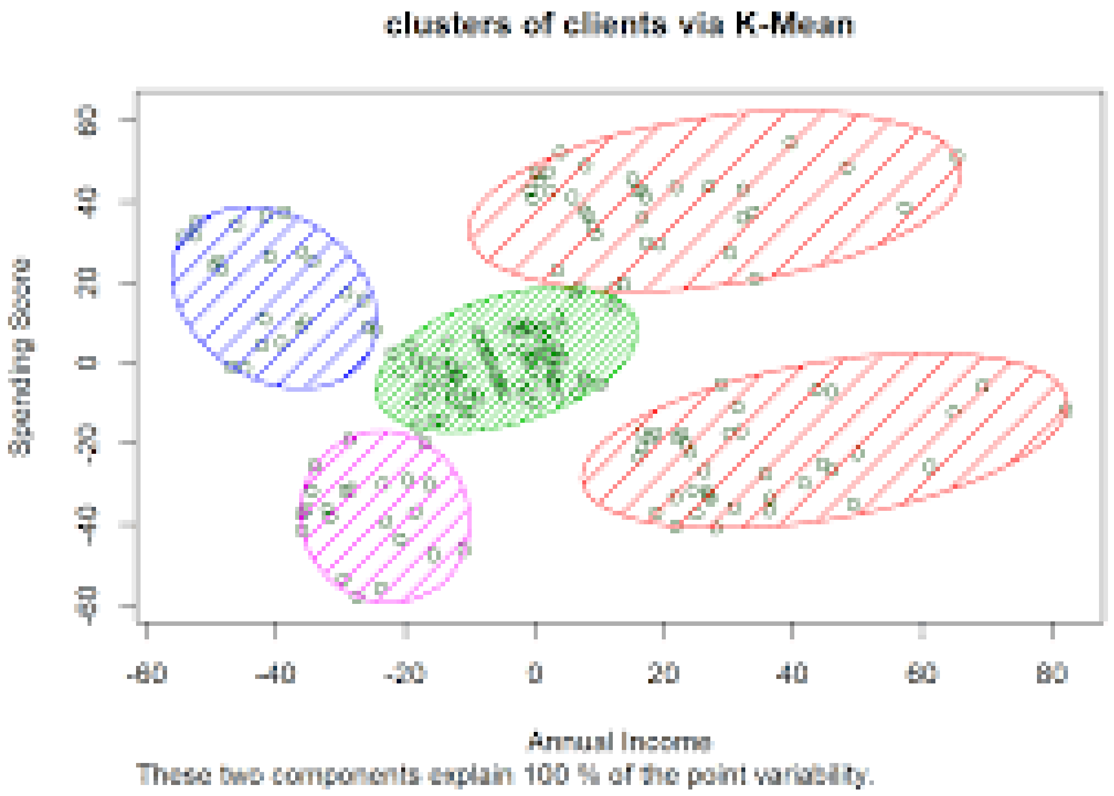
segmented using k-means



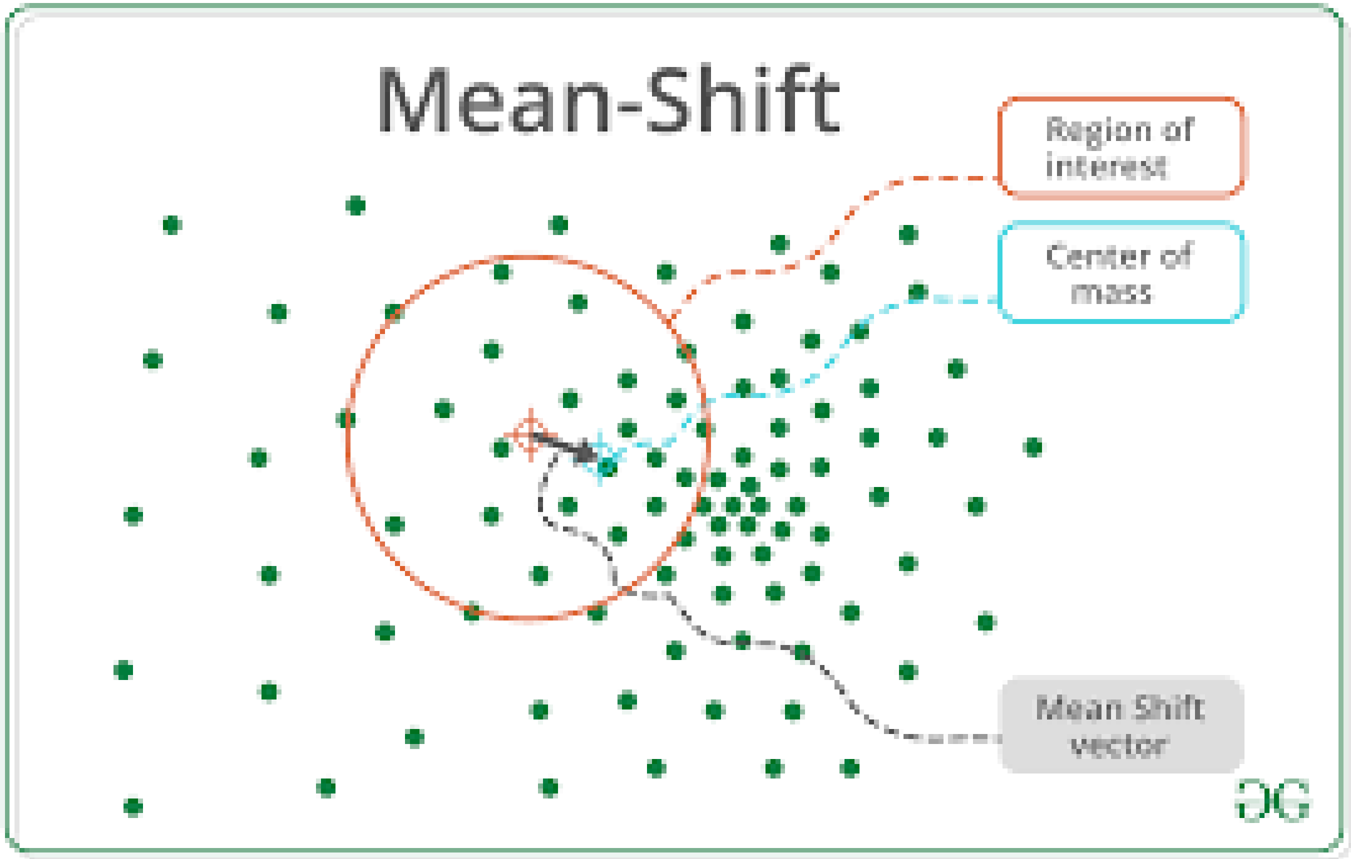
Mean Shift: Finding Local Modes in Data

- compute a feature vector representing each pixel
- each pixel goes to the segment represented by the cluster center
- that claims its feature vector.
- we know how many segments there will be
- stop splitting when the resulting clusters meet some similarity test

segmented using k-means



Mean Shift: Finding Local Modes in Data



Mean Shift: Finding Local Modes in Data

Mean-shift offset

Spatial kernel

Color weight

Local pixel offset

$$\Delta x = \frac{\sum_i K(x_i - x)\, w(x_i)\, (x_i - x)}{\underbrace{\sum_i K(x_i - x)\, w(x_i)}_{\text{Normalization factor}}}$$

Mean Shift: Finding Local Modes in Data

Mean Shift: Finding Local Modes in Data

- set of sample points in some feature space, with underlying probability density
- clusters as local maxima (local modes) in this density
- need an approximate representation of the density
- build an approximation is to use kernel smoothing
- What is needed exactly?

Mean Shift: Finding Local Modes in Data

- take a set of functions that look like “blobs” or “bumps,”
- place one over each datapoint, and so produce a smooth function
- is large when there are many data points close together
- small when the data points are widely separated
- specific kernel smoother

Specific kernel smoother

$$K(x;h) = \frac{(2\pi)^{(-d/2)}}{h^d} \exp\left(-\frac{1}{2}\frac{\| x \|^2}{h}\right)$$

$$f(x) = \left(\frac{1}{n}\right)\sum_{i=1}^{n} K(x_i - x; h)$$

$$; \; k(u) = \exp\left(-\tfrac{1}{2}u\right)$$

$$C = \frac{(2\pi)^{(-d/2)}}{nh^d}$$

$$f(x) = C\sum_{i=1}^{n} k\left(\left\| \frac{x - x_i}{h} \right\|^2\right)$$

Specific kernel smoother

$$
\begin{aligned}
\nabla f(x)\,|_{x=y} &= 0 \\
&= C \sum_i \nabla k\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \\
&= C \frac{2}{h} \sum_i [\boldsymbol{x}_i - \boldsymbol{y}] \left[g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \right] \\
&= C \frac{2}{h} \left[\frac{\sum_i \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)}{\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)} - \boldsymbol{y} \right] \times \left[\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \right].
\end{aligned}
$$

$$g = \frac{d}{du} k(u).$$

We expect that $\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)$ is nonzero, so that the maximum occurs when

$$\left[\frac{\sum_i \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)}{\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)} - \boldsymbol{y} \right] = 0,$$

Specific kernel smoother

or equivalently, when

$$\boldsymbol{y} = \frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)}.$$

The mean shift procedure involves producing a series of estimates $\boldsymbol{y}^{(j)}$ where

$$\boldsymbol{y}^{(j+1)} = \frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}.$$

The procedure gets its name from the fact that we are shifting to a point which has the form of a weighted mean (see Algorithm 9.5).

Finding a Mode with Mean Shift.

Start with an estimate of the mode $\boldsymbol{y}^{(0)}$ and a set of n data vectors $\boldsymbol{x}_i$ of dimension d, a scaling constant h, and g the derivative of the kernel profile

Until the update is tiny

Form the new estimate

$$\boldsymbol{y}^{(j+1)} = \frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}$$

Mean Shift Clustering.

For each data point $\boldsymbol{x}_i$

Apply the mean shift procedure (Algorithm 9.5), starting with $\boldsymbol{y}^{(0)} = \boldsymbol{x}_i$

Record the resulting mode as $\boldsymbol{y}_i$

Cluster the $\boldsymbol{y}_i$, which should form small tight clusters.

A good choice is an agglomerative clusterer with group average distance, stopping clustering when the group average distance exceeds a small threshold

The data point $\boldsymbol{x}_i$ belongs to the cluster that its mode $\boldsymbol{y}_i$ belongs to.

Mean Shift Segmentation.

For each pixel, p_i, compute a feature vector $\boldsymbol{x}_i = (\boldsymbol{x}_i^s, \boldsymbol{x}_i^r)$ representing spatial and appearance components, respectively.

Choose h_s, h_r the spatial (resp. appearance) scale of the smoothing kernel.

Cluster the $\boldsymbol{x}_i$ using this data and mean shift clustering (Algorithm 9.6).

(Optional) Merge clusters with fewer than t_{min} pixels with a neighbor; the choice of neighbor is not significant, because the cluster is tiny.

The i'th pixel belongs to the segment corresponding to its cluster center (for example, one could label the cluster centers $1 \dots r$, and then identify segments by computing a map of the labels corresponding to pixels).

Specific kernel smoother

$$K(x;h) = \frac{(2\pi)^{(-d/2)}}{h^d} \exp\left(-\frac{1}{2}\frac{\|x\|^2}{h}\right)$$

$$K(x;h_s,h_r) = \left[\frac{(2\pi)^{(-d_s/2)}}{h_s^{d_s}} k\left(\frac{x^s}{h_s}\right)\right] \left[\frac{(2\pi)^{(-d_r/2)}}{h_r^{d_r}} k\left(\frac{x^r}{h_r}\right)\right].$$

Mean Shift Clustering.

Segmentation by Clustering

Contents

1) Human vision: grouping and gestalt
2) Important applications
3) Image segmentation by clustering pixels
4) Segmentation, clustering, and graphs
5) Conclusion
6) Q&A

$$\nabla \cdot \mathbf{E} = \rho/\varepsilon_0$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \boldsymbol{j}_c$$

where

$$\nabla = \hat{\mathbf{i}}\frac{\partial}{\partial x} + \hat{\mathbf{j}}\frac{\partial}{\partial y} + \hat{\mathbf{k}}\frac{\partial}{\partial z}$$

Why graph?

- summarize data items (the cluster centers in the k-means algorithm) - vertices
- similarity between data items – weighted edges
- not be useful to compare all pairs of data items
- process of clustering the data then becomes
- one of segmenting the graph into connected components

Terminology and Facts for Graphs

- $G = \{V,E\}$, $E \subset V \times V$.
- The degree of a vertex is the number of edges incident on that vertex
- A directed graph is one in which edges (a, b) and (b, a) are distinct
- An undirected graph no distinction is drawn between edges (a, b) and (b, a).
- A weighted graph is one in which a weight is associated with each edge.

Terminology and Facts for Graphs

- Two edges are consecutive if they have a vertex in common
- A path is a sequence of consecutive edges.
- A circuit is a path which ends at the vertex at which it begins.
- Two vertices are said to be connected when there is a sequence of edges
- Starting at the one and ending at the other;

Terminology and Facts for Graphs

- A connected graph is one where every pair of vertices is connected
- A tree is a connected graph with no circuits
- spanning tree is a tree with vertices V and edges a subset of E
- Every graph consists of a disjoint set of connected components
- G ={V1 ∪ V2 . . . Vn,E1 ∪E2 . . .En}, where {Vi,Ei} are all connected graphs
- no edge in E that connects an element of Vi with one of Vj for i != j.
- A forest is a graph whose connected components are trees

CAMBRIDGE INSTITUTE OF TECHNOLOGY

Terminology and Facts for Graphs

- in a directed graph identify one vertex as a source s and another as a target t
- Associate with each directed edge e a capacity,c(e), non-negative number.
- A flow is a non-negative value f(e) associated with each edge with the following properties.
- First, 0 ≤ f(e) ≤ c(e). Second, at any vertex v ∈ {V −s−t},
- $$\sum_{e \text{arriving at } v} f(e) - \sum_{e \text{leaving from} v} f(e) = 0$$

Terminology and Facts for Graphs

- Decompose the vertices into two disjoint sets S and T , such that $s \in S$ and $t \in T$.
- This represents a cut. Consider $W \in E$, the set of directed edges from S to T.
- The value of the cut is $\sum_{e \in W} c(e)$.
- value of the cut can again be minimized efficiently
- efficient algorithms to maximize the flow in

Agglomerative Clustering with a Graph

Start with a set of clusters $\mathcal{C}_i$, one cluster per pixel.
Sort the edges in order of non-decreasing edge weight, so that
$w(e_1) \geq w(e_2) \geq \ldots \geq w(e_r)$.

For $i = 1$ to r
    If the edge e_i lies inside a cluster
        do nothing
    Else
        One end is in cluster $\mathcal{C}_l$ and the other is in cluster $\mathcal{C}_m$
        If $diff(\mathcal{C}_l, \mathcal{C}_m) \leq MInt(\mathcal{C}_l, \mathcal{C}_m)$
            Merge $\mathcal{C}_l$ and $\mathcal{C}_m$ to produce a new set of clusters.

Report the remaining set of clusters.

Algorithm 9.8: Agglomerative Clustering with Graphs.

Agglomerative Clustering with a Graph

stop clustering. We define the internal difference of a component to be the largest weight in the minimum spanning tree of the component. Write $M(\mathcal{C}) = \{V_\mathcal{C}, E_M\}$ for the minimum spanning tree of $\mathcal{C}$. Then, we have

$$\text{int}(\mathcal{C}) = \max_{e \in M(\mathcal{C})} w(e).$$

cluster. Then the difference between two components is the minimum weight edge connecting two components. Write $\mathcal{C}_1$, $\mathcal{C}_2$ for the two components, $\mathcal{E}$ for the edges, and $w(v_1, v_2)$ for the weight of the edge joining v_1 and v_2. Then, we have

$$\text{diff}(\mathcal{C}_1, \mathcal{C}_2) = \min_{v_1 \in \mathcal{C}_1, v_2 \in \mathcal{C}_2, (v_1, v_2) \in \mathcal{E}} w(v1, v2).$$

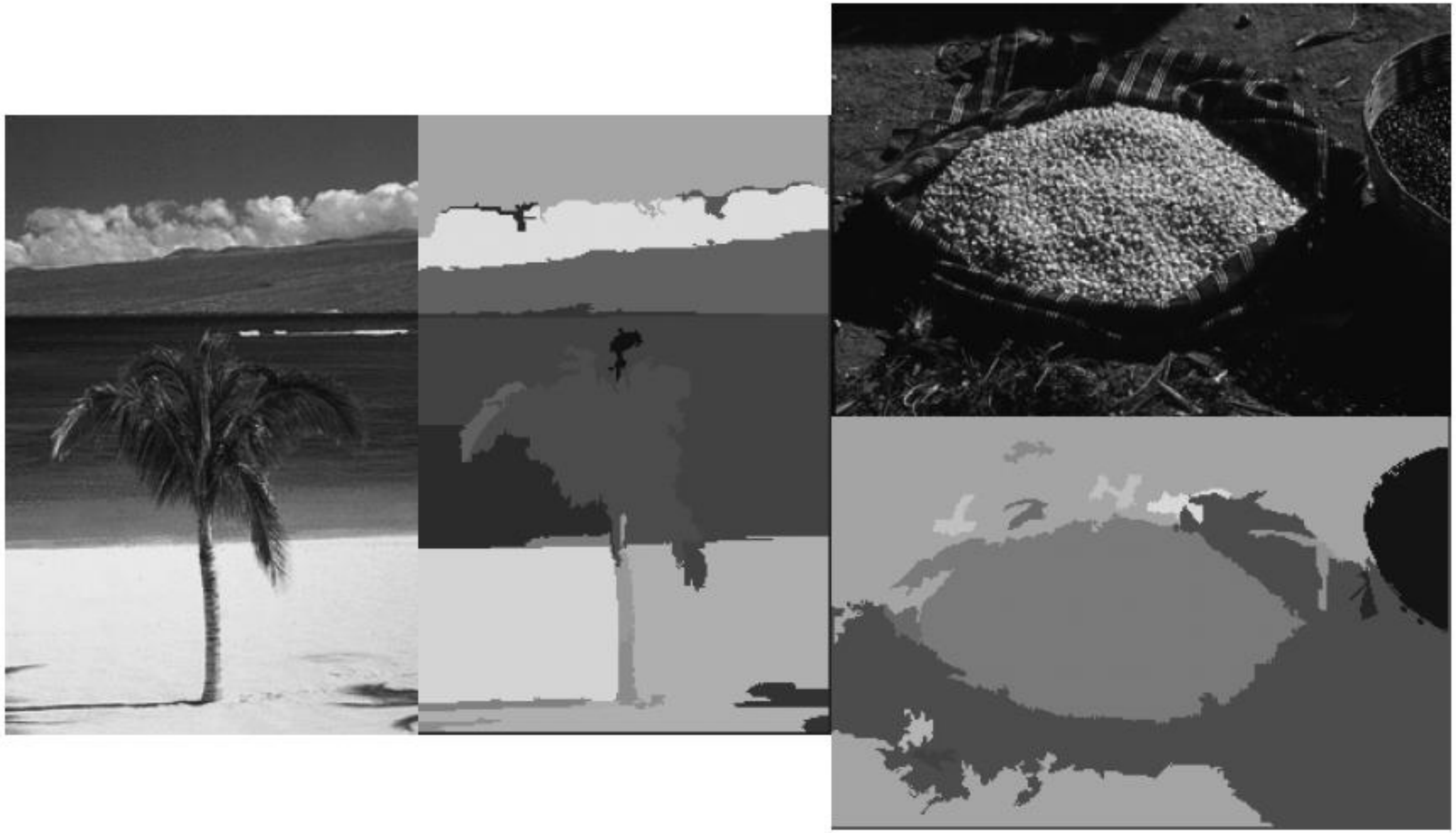
Agglomerative Clustering with a Graph

Huttenlocher (2004) define a function of two clusters, MInt, as

$$\mathrm{MInt}(\mathcal{C}_1, \mathcal{C}_2) = \min(\mathrm{int}(\mathcal{C}_1) + \tau(\mathcal{C}_1), \mathrm{int}(\mathcal{C}_2) + \tau(\mathcal{C}_2))$$

where $\tau(\mathcal{C})$ is a term that biases the internal difference upward for small clusters; Felzenszwalb and Huttenlocher (2004) use $\tau(\mathcal{C}) = k/ \mid \mathcal{C} \mid$, for k some constant parameter. This algorithm is notably fast and relatively accurate (Figure 9.22).

Agglomerative Clustering with a Graph



CAMBRIDGE
INSTITUTE OF TECHNOLOGY

Divisive Clustering with a Graph

- each pixel in the map carries one of three labels
- foreground, background or unknown (these maps are sometimes known as trimaps)
- pixel that looks like the foreground examples should get a foreground label
- Pixel should tend to have labels that are the same as their neighbors
- Boykov and Jolly (2001) phrase this problem as energy minimization.

Divisive Clustering with a Graph

- F/B/U for the set of pixels with foreground/background/unknown labels
- associate a binary variable δ_i with the ith unknown pixel
- $\delta_i = -1$ if the ith pixel is background, and $\delta_i = 1$ if the ith pixel is foreground
- Write p_i for a vector representing the ith pixel.
- could contain intensity; intensity and color; intensity, color and texture;

Divisive Clustering with a Graph

- Write df (p) for a function that compares the pixel vector p with the FG model;
- write db(p) for a function that compares the pixel vector with the background.
- pixel that looks like the foreground examples should get a foreground label
- Write N(i) for the neighbors of pixel i.
- Boykov and Jolly (2001) phrase this problem as energy minimization.

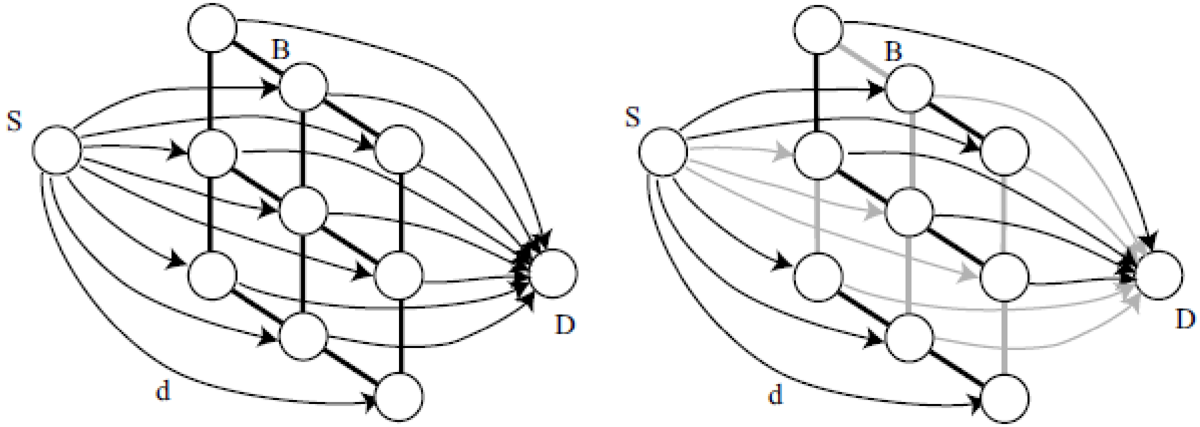
Divisive Clustering with a Graph

- B(pi, pj) for a non-negative, symmetric function that compares two pixels
- use as a cost for assigning neighboring pixels to different models
- B should be large for two pixels that are similar, and small for different pixels;
- $(1/2)(1 - \delta_i\delta_j)$ has the value 1 when δ_i and δ_j are different, and 0 otherwise.
- minimize subject to $\delta_k = 1$ for $k \in F$ and $\delta_k = 0$ for $k \in B$.

An energy function

$$
\begin{aligned}
E^*(\delta) \quad = \quad & \sum_{i \in \mathcal{I}} d_f(p_i)\frac{1}{2}(1+\delta_i) + d_b(p_i)\frac{1}{2}(1-\delta_i) + \\
& \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{N}(i)} B(p_i, p_j)(\frac{1}{2})(1-\delta_i\delta_j)
\end{aligned}
$$

Graph derived from an image to set up foreground / background segmentation as a graph cut problem



edge	weight	case
(i, j)	$B(\boldsymbol{p}_i, \boldsymbol{p}_j)$	i, j, neighbors
$(S \rightarrow i)$	K	$p \in \mathcal{F}$
	0	$p \in \mathcal{B}$
	$d_f(i)$	otherwise
$(i \rightarrow D)$	K	$p \in \mathcal{B}$
	0	$p \in \mathcal{F}$
	$d_b(i)$	otherwise

$K = 1 + \max_{\boldsymbol{p} \in \mathcal{I}} \sum_{\boldsymbol{q}:\{\boldsymbol{p},\boldsymbol{q}\} \in \mathcal{N}} B(\boldsymbol{p}, \boldsymbol{q})$.

Normalized Cuts

- min-cut does not work well without good foreground and background models
- cut does not balance difference between segments with coherence within segments
- normalized cut
- cut the graph into two connected components such that
- Cost of the cut is a small fraction of the total affinity within each group

CAMBRIDGE
INSTITUTE OF TECHNOLOGY

Normalized Cuts

- a weight on each edge - affinity between the pixels
- weight of an arc connecting similar nodes should be large
- normalized cut must cut the graph into two connected components
- such that cost of cut is a small fraction of the total affinity within each group
- decomposing a weighted graph V into two components A and B

Different affinity functions comparing pixels for a graph based segmenter

Property	Affinity function	Notes
Distance	$\exp\left\{-\left((x-y)^t(x-y)/2\sigma_d^2\right)\right\}$	
Intensity	$\exp\left\{-\left((I(x)-I(y))^t(I(x)-I(y))/2\sigma_I^2\right)\right\}$	$I(x)$ is the intensity of the pixel at x.
Color	$\exp\left\{-\left(\mathrm{dist}(c(x),c(y))^2/2\sigma_c^2\right)\right\}$	$c(x)$ is the color of the pixel at x.
Texture	$\exp\left\{-\left((f(x)-f(y))^t(f(x)-f(y))/2\sigma_I^2\right)\right\}$	$f(x)$ is a vector of filter outputs describing the pixel at x computed as in Section 6.1.

Normalized Cuts

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

(where cut(A,B) is the sum of weights of all edges in V that have one end in A and the other in B,

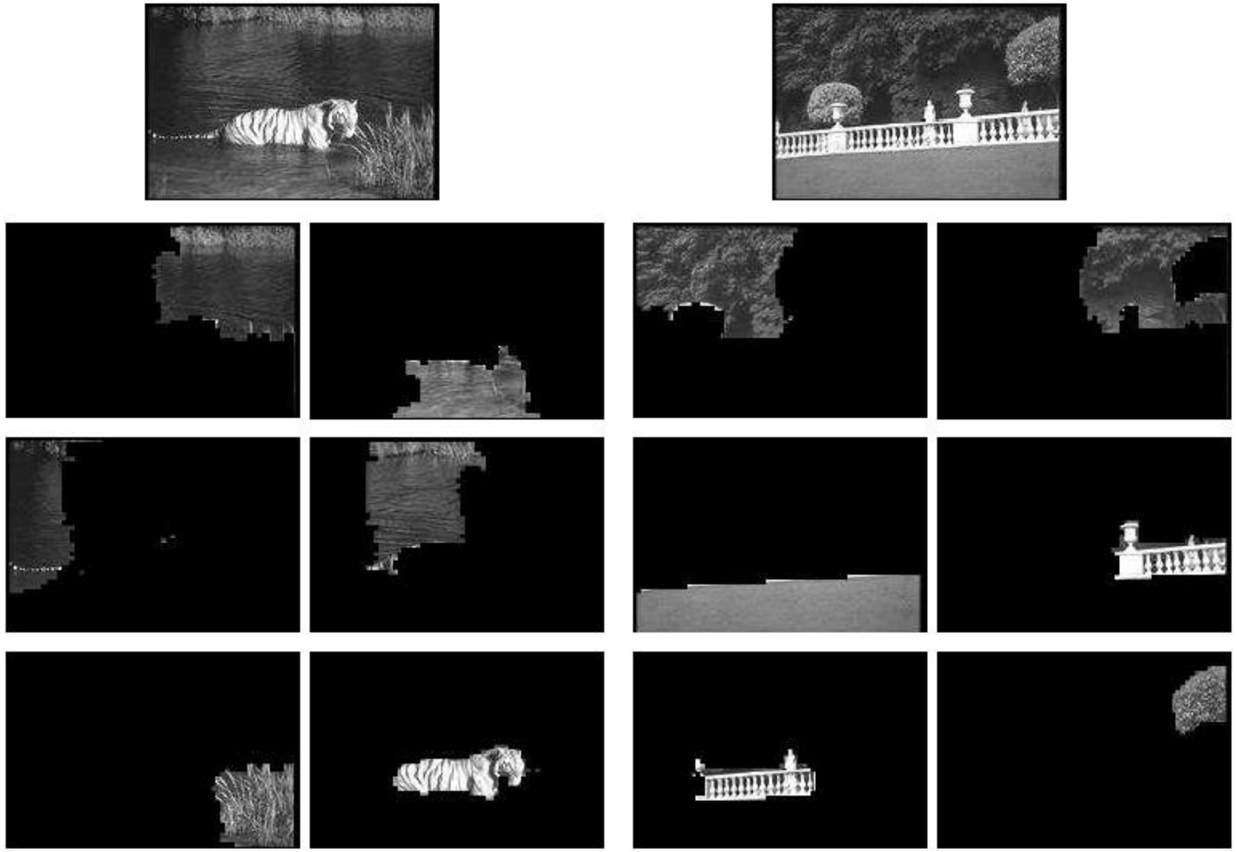
assoc(A, V) is the sum of weights of all edges that have one end in A).

score is small if the cut separates two components

find the cut with the minimum value of this criterion, called a normalized cut.

NP-complete problem

Images segmented using the normalized cuts framework



Conclusion

Plethora of applicatcions of computer vision

- code
- test
- use

Q&A

Contact

- **Prof. D. Antony Louis Piriyakumar**
  Dean (Research and development)
  Cambridge Institute of Technology
- K.R. Puram,
  560036 Bengaluru, India
- Mobile: +91 98459 25132
- E-mail:
  dean_rd@Cambridge.edu.in