Lecture 10

Computer Vision

Grouping and Segmentation
Outline

• Grouping problems in vision
  • Image segmentation: grouping of pixels

• Grouping cues in Human Visual System
  • Gestalt perceptual grouping laws

• Image Segmentation
  • 2-region (binary)
    • thresholding
    • graph cuts
      • used in MS office 2010 for background removal
      • based on the work of our faculty Yuri Boykov

• General Grouping (or unsupervised learning)
  • K-means clustering
Examples of Grouping in Vision

- Group pixels into regions
  - image segmentation

- Group video frames into shots

- Group image regions into objects
• For many applications, useful to segment image pixels into blobs that (hopefully) belong to the same object or surface

• How to do this without (necessarily) object recognition?
  • a bit subjective, but well-studied

• Inspiration from Gestalt psychology
  • humans perceive the world as a collection of objects with relationships between them, not as a set of pixels
Gestalt Psychology

- Whole is greater than the sum of its parts
  - eye sees an object in its entirety before perceiving its individual parts
- Identified factors that predispose a set of elements to be grouped by human visual system
  - perceptual grouping
• Most human observers report no particular grouping
Gestalt Principles of Grouping

- Common form, includes:

  - **shape**
  - **color**
  - **size**
Proximity
• Good continuation
Gestalt Principles of Grouping

• Connectivity
  • stronger than color
• Symmetry
Gestalt Principles of Grouping

- Familiarity
Gestalt Principles of Grouping

- Closure
Gestalt Principles of Grouping

• Closure
Gestalt Principles of Grouping

- Closure
Gestalt Principles of Grouping

- Common fate
Gestalt Principles of Grouping

- Higher level knowledge?
Gestalt Principles of Grouping

• Many other Gestalt grouping principles
  • parallelism, convexity, colinearity, common depth, etc.
• Gestalt principles are an inspiration to computer vision
  • they seem to rely on nature of objects in the world, most do not involve higher level knowledge (object recognition)
  • should help to segment objects without necessarily performing object recognition
• But most are difficult to implement in algorithms
  • used often
    • color, proximity
      • we will use these as well
  • used sometimes
    • convexity, good continuation, common motion, colinearity
• Many types of image segmentation

• We will focus on figure-ground (FG)
  • also called object/background segmentation
FG Segmentation: Thresholding

• Suppose the object is brighter than the background

• Threshold gray scale image $f$:
  
  if $f(x,y) < T$ then pixel $(x,y)$ is background
  
  if $f(x,y) \geq T$ then pixel $(x,y)$ is foreground

$T = 120, \quad T = 180, \quad T = 220$
FG Segmentation: Thresholding

- Tiny isolated foreground regions, isolated background regions
- Result looks wrong even if you did not know object is a swan

Can we clean this result up?
FG Segmentation: Motivation

- Know object is light, background is dark
- Do not know object shape
  - show background with red, foreground with blue

**bad result:** crazy object shape
**bad result:** object is dark, background light
**good result:** light object of good shape, dark background
• Formulate an **objective** or **energy** function $E$ to measure how good segmentation is
  • low value means good segmentation

• After energy function is designed, search over all possible segmentations for the best one
  • one with lowest energy
FG Segmentation: Energy Function

• Energy has two terms
  • **data term:**
    • makes it cheap to assign light pixels to foreground, expensive to the background
    • makes it cheap to assign dark pixels to the background, and expensive to the foreground
  
• **smoothness term:** ensures nice object shape

• both terms are needed for a good energy function
FG Segmentation: Data Term

- Should be cheap to assign light pixels to foreground, expensive to the background.
- For each pixel \((x, y)\), we will pay \(D_{(x, y)}\) (background) to assign it to background and \(D_{(x, y)}\) (foreground) to assign it to the foreground.
- Let the smallest image intensity be 5, and largest 20
  \[
  D_{(x, y)}\text{(background)} = f(x, y) - 5
  \]
  \[
  D_{(x, y)}\text{(foreground)} = 20 - f(x, y)
  \]

<table>
<thead>
<tr>
<th>11</th>
<th>17</th>
<th>19</th>
<th>11</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>19</td>
<td>19</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>20</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>7</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>12</th>
<th>14</th>
<th>6</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>14</td>
<td>14</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>15</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9</th>
<th>3</th>
<th>1</th>
<th>9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>13</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

- **Brown** pixel prefers foreground, **green** prefers background.
FG Segmentation: Data Term

- $E_{data}$ sums data $D_{(x,y)}$ term over all pixels $(x,y)$
- Foreground blue, background red

$E_{data} = 6 + 3 + 1 + 6 + 1 + 3 + 1 + 1 + 9 + 9 + 1 + 1 + 0 + 6 + 2 + 9 + 6 + 9 + 0 + 0 + 6 + 1 + 2 + 0 + 0 = 73$

$E_{data} = 283$

$E_{data} = 97$
FG Segmentation: Smoothness Term

- **Smoothness term**: ensures nice object shape
- Consider segmentations below

1. **bad shape**

   - 17 discontinuities
   - $E_{smooth} = 17$

2. **nice shape**

   - 8 discontinuities
   - $E_{smooth} = 8$

3. **nice shape**

   - 7 discontinuities
   - $E_{smooth} = 7$

- **discontinuity**: when two nearby pixels are in different segments
- **smoothness term** is the number of discontinuities
FG Segmentation: Total Energy

- Now combine both data and smoothness energy terms

\[ E_{\text{data}} = 73 \]
\[ E_{\text{smooth}} = 17 \]
\[ E = E_{\text{data}} + E_{\text{smooth}} = 90 \]

\[ E_{\text{data}} = 283 \]
\[ E_{\text{smooth}} = 7 \]
\[ E = E_{\text{data}} + E_{\text{smooth}} = 290 \]

\[ E_{\text{data}} = 97 \]
\[ E_{\text{smooth}} = 8 \]
\[ E = E_{\text{data}} + E_{\text{smooth}} = 105 \]

- What went wrong?
- Smoothness term weighs very little relative to the data term
  - it basically gets ignored in the combined energy
- Solution: increase the weight of the smoothness term
FG Segmentation: Total Energy

- **Solution:** *increase* the weight of the smoothness term

\[ E = E_{\text{data}} + \lambda E_{\text{smooth}} \]

- Take, for example, \( \lambda = 10 \)

\begin{align*}
E_{\text{data}} &= 73 \\
E_{\text{smooth}} &= 170 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 243
\end{align*}

\begin{align*}
E_{\text{data}} &= 83 \\
E_{\text{smooth}} &= 70 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 353
\end{align*}

\begin{align*}
E_{\text{data}} &= 97 \\
E_{\text{smooth}} &= 80 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 177
\end{align*}
FG Segmentation: Energy Formula

- Now we need to put everything into formulas
- $s(x,y)$ is the segmentation **label**
  
  $s(x,y) = 1$ means $(x,y)$ is foreground pixel
  
  $s(x,y) = 0$ means $(x,y)$ is background pixel
- Convenient to write pixel $(x,y)$ as $p$ (or $q$, $r$, ...)
- Denote all pairs of nearby pixels: $\mathbf{N}$

\[
\mathbf{N} = \{(p,q), (p,r), (v,u), (u,w), (y,h), (h,z), (p,v), (v,y), (q,u), (u,h), (r,w), (w,z)\}
\]

\[
E(s) = E_{\text{data}}(s) + \lambda \cdot E_{\text{smooth}}(s) = \sum_p D_p(s_p) + \lambda \sum_{(p,q) \in \mathbf{N}} [s_p \neq s_q]
\]

- where [true] = 1, [false] = 0
FG Segmentation: Formula Practice with $\lambda = 1$

$$E(s) = \sum_{p} D_p(s_p) + \lambda \sum_{(p,q) \in N} [s_p \neq s_q]$$

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>background $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>6 12 14</td>
</tr>
<tr>
<td>v</td>
<td>u</td>
<td>w</td>
<td>12 14 14</td>
</tr>
<tr>
<td>y</td>
<td>h</td>
<td>z</td>
<td>14 14 15</td>
</tr>
</tbody>
</table>

Pixel names:
- $D_0$: background
- $D_1$: foreground

Segmentation $s$:
- $D_p(0) + D_q(1) + D_r(0)$
- $D_v(0) + D_u(0) + D_w(0)$
- $D_y(0) + D_h(1) + D_z(1)$

$$E(s) = D_p(0) + D_q(1) + D_r(0) + D_v(0) + D_u(0) + D_w(0) + D_y(0) + D_h(1) + D_z(1)$$

$$= [s_p \neq s_q] + [s_q \neq s_r] + [s_v \neq s_u] + [s_u \neq s_w] + [s_v \neq s_y] + [s_u \neq s_h] + [s_r \neq s_w] + [s_v \neq s_y] + [s_u \neq s_h] + [s_w \neq s_z]$$

$$= 9 + 12 + 1 + 3 + 1 + 1 + 1 + 14 + 15 = 57 + 6 = 63$$
FG Segmentation: Contrast Sensitive Discontinuity

- Where is object boundary more likely?

- Make discontinuity cost depend on image contrast
  - helps align object boundary with image edges

- Replace $[s_p \neq s_q]$ with $w_{pq} \cdot [s_p \neq s_q]$ where $w_{pq}$ is
  - large if intensities of pixels $p,q$ are similar
  - small if intensities of pixels $p,q$ are not similar
FG Segmentation: Contrast Sensitive Discontinuity

\[
\frac{(f(p) - f(q))^2}{2\sigma^2}
\]

• Good choice \( w_{pq} = \lambda \cdot e \)

• Here \( f(p) \) is intensity of pixel \( p \), \( f(q) \) intensity of pixel \( q \)
  • for color image, replace \( (f(p) - f(q))^2 \) with \( ||f(p) - f(q)||^2 \)
  • equivalent to processing each color channel individually

• Parameter \( \sigma^2 \) is either
  • set by hand (trail and error)
  • or computed as average of \( (f(p) - f(q))^2 \) over all neighbors in \( N \)

• Complete energy:
  • note that is now folded into \( w_{pq} \)

\[
E(s) = \sum_p D_p(s_p) + \sum_{(p,q)\in N} w_{pq}[s_p \neq s_q]
\]
FG Segmentation: Example

\[ E(s) = \sum_p D_p(s_p) + \sum_{(p,q) \in N} w_{pq} [s_p \neq s_q] \]

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>v</th>
<th>u</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>h</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

pixel names

contrast sensitive weights

\[ E(s) \text{ (segmentation } s) = \text{ data term as before} + 3 \cdot [s_p \neq s_q] + 2 \cdot [s_q \neq s_r] + 6 \cdot [s_v \neq s_u] + 2 \cdot [s_u \neq s_w] + 7 \cdot [s_y \neq s_h] + 1 \cdot [s_h \neq s_z] + 3 \cdot [s_p \neq s_v] + 2 \cdot [s_q \neq s_u] + 6 \cdot [s_r \neq s_w] + 4 \cdot [s_v \neq s_y] + 2 \cdot [s_u \neq s_h] + 1 \cdot [s_w \neq s_z] \]

\[ = 3 + 2 + 0 + 0 + 7 + 0 + 0 + 2 + 0 + 0 + 2 + 1 = 57 + 15 = 72 \]
FG Segmentation: Optimization

- We are all set to find the best segmentation $s^*$

$$s^* = \arg \min_s E(s)$$

- How to do this efficiently?
- Even for a 9 pixel image, there are $2^9$ possible segmentations!

- $O(2^n)$ for an $n$ pixel image
FG Segmentation: Optimization Graph

- Build weighted graph
  - one node per pixel
    - connect to neighbor pixel nodes with weight $w_{pq}$
  - two special nodes (terminals) source $s$, sink $t$
    - $s$ connects to each pixel node $p$ with weight $D_p(0)$
    - $t$ connects to each pixel node $p$ with weight $D_p(1)$
    - graph below omits most of these edges for clarity
**FG Segmentation: Optimization with Graph Cut**

- **Cut** is subset of edges $C$ s.t. removing $C$ from graph makes $s$ and $t$ disconnected
  - cost of cut $C$ is sum of its edge weights
- Minimum Graph Cut Problem
  - find a cut $C$ of minimum cost
- Minimum cut $C$ gives the smallest cost segmentation [Boykov&Veksler, 1998]
  - nodes that stay connected to source in the `cut’ graph become foreground
  - nodes that stay connected to sink in the `cut’ graph become background
  - In the example, $p$ gets background label, $v$ and $y$ get foreground label
- Efficient algorithms for min-cut/max-flow
FG Segmentation: Segmentation Result

- Data terms
  - blue means low weight, red high weight

- Contrast sensitive edge weights
  - dark means low weight, bright high weight

input

segmentation

foreground background horizontal vertical
**FG Segmentation: Interactive**

- What if we do not know object/background color?
- Can ask user for help
- Interactive Segmentation [Boykov&Jolly, 2001]

User scribbles foreground and background seeds
- these are **hard** constrained to be foreground and background, respectively
  - for any pixel \( p \) that user marks as a **foreground**, set \( D_{p}(1) = 0, D_{p}(0) = \infty \)
  - for any pixel \( p \) that user marks as a **background**, set \( D_{p}(1) = \infty, D_{p}(0) = 0 \)
  - for unmarked pixels, set \( D_{p}(1) = D_{p}(0) = 0 \)

- Smoothness term is as before
  - Contrast sensitive works best for interactive segmentation
FG Segmentation: Interactive Results

- Initial seeds:

- Add more seeds for correction:
FG Segmentation: More Interactive Results
General Grouping or Clustering

- General Clustering (Grouping)
- Have samples (also called feature vectors, examples, etc.) \( x_1, \ldots, x_n \)
- Cluster similar samples into groups
- This is also called unsupervised learning
  - samples have no labels
  - think of clusters as ‘discovering’ labels

**Diagram:**
- Sci-fi movies
- Horror movies
- Documentaries

Recall supervised learning
How does this Relate to Image Segmentation?

- Represent image pixels as feature vectors $x_1, \ldots, x_n$
  - For example, each pixel can be represented as
    - intensity, gives one dimensional feature vectors
    - color, gives three-dimensional feature vectors
    - color + coordinates, gives five-dimensional feature vectors
- Cluster them into $k$ clusters, i.e. $k$ segments

```
input image | feature vectors for clustering based on color
-------------|------------------------------------------
9 4 2       | [9 4 2] [7 3 1] [8 6 8]
7 3 1       | [8 2 4] [5 8 5] [3 7 2]
8 6 8       | [9 4 5] [2 9 3] [1 4 4]
```


How does this Relate to Image Segmentation?

<table>
<thead>
<tr>
<th>input image</th>
<th>feature vectors for clustering based on color and image coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 4 2</td>
<td>[9 4 2 0 0] [7 3 1 0 1] [8 6 8 0 2]</td>
</tr>
<tr>
<td>7 3 1</td>
<td>[8 2 4 1 0] [5 8 5 1 1] [3 7 2 1 2]</td>
</tr>
<tr>
<td>8 6 8</td>
<td>[9 4 5 2 0] [2 9 3 2 1] [1 4 4 2 2]</td>
</tr>
</tbody>
</table>

The table shows an input image with feature vectors for clustering based on color and image coordinates.
K-means Clustering: Objective Function

- Probably the most popular clustering algorithm
  - assumes know the number of clusters should be $k$
- Optimizes (approximately) the following objective function

$$
J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} \left\| x - \mu_i \right\|^2
$$
K-means Clustering: Objective Function

\[ J_{SSE} = D_1 + D_2 + D_3 \]

Good (tight) clustering
smaller value of \( J_{SSE} \)

Bad (loose) clustering
larger value of \( J_{SSE} \)
K-means Clustering: Algorithm

- Initialization step
  1. pick \( k \) cluster centers randomly
K-means Clustering: Algorithm

• Initialization step
  1. pick \( k \) cluster centers randomly
K-means Clustering: Algorithm

• Initialization step
  1. pick $k$ cluster centers randomly
  2. assign each sample to closest center
K-means Clustering: Algorithm

• Initialization step
  1. pick \( k \) cluster centers randomly
  2. assign each sample to closest center

• Iteration step
  1. compute means in each cluster
K-means Clustering: Algorithm

- **Initialization step**
  1. pick \( k \) cluster centers randomly
  2. assign each sample to closest center

- **Iteration step**
  1. compute means in each cluster
  2. re-assign each sample to the closest mean
K-means Clustering: Algorithm

- **Initialization step**
  1. pick $k$ cluster centers randomly
  2. assign each sample to closest center

- **Iteration step**
  1. compute means in each cluster
  2. re-assign each sample to the closest mean

- Iterate until clusters stop changing
K-means Clustering: Algorithm

- **Initialization step**
  1. pick $k$ cluster centers randomly
  2. assign each sample to closest center

- **Iteration step**
  1. compute means in each cluster
  2. re-assign each sample to the closest mean

- Iterate until clusters stop changing

- Can prove that this procedure decreases the value of the objective function $J_{\text{SEE}}$
K-means: Approximate Optimization

- K-means is fast and works quite well in practice
- But can get stuck in a local minimum of objective $J_{\text{SEE}}$
  - not surprising, since the problem is NP-hard

**Initialization**

**Converged to local min**

**Global minimum**
K-means Clustering: Example

- with $k = 2$

<table>
<thead>
<tr>
<th>Feature Vectors</th>
<th>[9 4 2]</th>
<th>[7 3 1]</th>
<th>[8 6 8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[8 2 4]</td>
<td>[5 8 5]</td>
<td>[3 7 2]</td>
<td></td>
</tr>
<tr>
<td>[9 4 5]</td>
<td>[2 9 3]</td>
<td>[1 4 4]</td>
<td></td>
</tr>
</tbody>
</table>
K-means Clustering: Example

- with $k = 2$
- Initialize
  - pick $[9 \ 4 \ 2]$ $[5 \ 8 \ 5]$ as cluster centers

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

feature vectors

$[9 \ 4 \ 2]$ $[7 \ 3 \ 1]$ $[8 \ 6 \ 8]$

$[8 \ 2 \ 4]$ $[5 \ 8 \ 5]$ $[3 \ 7 \ 2]$

$[9 \ 4 \ 5]$ $[2 \ 9 \ 3]$ $[1 \ 4 \ 4]$
K-means Clustering: Example

- with $k = 2$
- Initialize
  - pick $[9 \ 4 \ 2] \ [5 \ 8 \ 5]$ as cluster centers
  - assign each feature vector to closest center

$\text{dist}( [9 \ 4 \ 2] - [9 \ 4 \ 2] ) = 0 \Rightarrow [9 \ 4 \ 2]$ goes to pink cluster
• with $k = 2$

• Initialize
  • pick $[9 \ 4 \ 2]$ $[5 \ 8 \ 5]$ as cluster centers
  • assign each feature vector to closest center

\[
\begin{align*}
\text{dist}( [9 \ 4 \ 2] - [9 \ 4 \ 2] ) &= 0 \implies [9 \ 4 \ 2] \text{ goes to pink cluster} \\
\text{dist}( [7 \ 3 \ 1] - [9 \ 4 \ 2] ) &= (7-9)^2 + (3-4)^2 + (1-2)^2 = 6 \\
\text{dist}( [7 \ 3 \ 1] - [5 \ 8 \ 5] ) &= (7-5)^2 + (3-8)^2 + (1-5)^2 = 45
\end{align*}
\]
K-means Clustering: Example

- with $k = 2$
- Initialize
  - pick $[9 \ 4 \ 2]$ $[5 \ 8 \ 5]$ as cluster centers
  - assign each feature vector to closest center

\[
\begin{align*}
\text{dist}( [9 \ 4 \ 2] - [9 \ 4 \ 2] ) &= 0 \quad \Rightarrow [9 \ 4 \ 2] \text{ goes to pink cluster} \\
\text{dist}( [7 \ 3 \ 1] - [9 \ 4 \ 2] ) &= (7-9)^2 + (3-4)^2 + (1-2)^2 = 6 \\
\text{dist}( [7 \ 3 \ 1] - [5 \ 8 \ 5] ) &= (7-5)^2 + (3-8)^2 + (1-5)^2 = 45 \\
\text{dist}( [8 \ 6 \ 8] - [9 \ 4 \ 2] ) &= (8-9)^2 + (6-4)^2 + (8-2)^2 = 41 \\
\text{dist}( [8 \ 6 \ 8] - [5 \ 8 \ 5] ) &= (8-5)^2 + (6-8)^2 + (8-5)^2 = 22
\end{align*}
\]

\[\begin{array}{ccc}
9 & 4 & 2 \\
7 & 3 & 1 \\
8 & 6 & 8 \\
\end{array}\]
K-means Clustering: Example

- with $k = 2$
- Initialize
  - pick [9 4 2] [5 8 5] as cluster centers
  - assign each feature vector to closest center
  - repeat for the rest of feature vectors

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>initial clustering</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial clustering:

- [8 2 4] [5 8 5] [3 7 2]
- [9 4 5] [2 9 3] [1 4 4]
K-means Clustering: Example

- Iterate
  - compute cluster means

\[
\begin{align*}
\mu_1 &= \frac{[9 \ 4 \ 2] + [7 \ 3 \ 1] + [8 \ 2 \ 4] + [9 \ 4 \ 5]}{4} = [8.25 \ 3.25 \ 3] \\
\mu_2 &= \frac{[8 \ 6 \ 8] + [5 \ 8 \ 5] + [3 \ 7 \ 2] + [2 \ 9 \ 3] + [1 \ 4 \ 4]}{5} = [3.8 \ 6.8 \ 4.4]
\end{align*}
\]
K-means Clustering: Example

- Iterate
  - compute cluster means
    \[ \mu_1 = [8.25 \ 3.25 \ 3] \]
    \[ \mu_2 = [3.8 \ 6.8 \ 4.4] \]
  - reassign samples to the closest mean

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>4</th>
<th>2</th>
<th>7</th>
<th>3</th>
<th>1</th>
<th>8</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{dist}([\mathbf{9} \ 4 \ 2] - [8.25 \ 3.25 \ 3]) = (8.25-9)^2 + (3.25-4)^2 + (3-2)^2 \approx 2
\]
\[
\text{dist}([\mathbf{9} \ 4 \ 2] - [3.8 \ 6.8 \ 4.4]) = (3.8-9)^2 + (6.8-4)^2 + (4.4-2)^2 \approx 41
\]
\[
[\mathbf{9} \ 4 \ 2] \text{ goes to pink cluster}
\]
K-means Clustering: Example

- **Iterate**
  - compute cluster means
    - $\mu_1 = [8.25\ 3.25\ 3]$  
    - $\mu_2 = [3.8\ 6.8\ 4.4]$  
  - reassign samples to the closest mean
  - repeat for
    - $[7\ 3\ 1]\ [8\ 6\ 8]$  
    - $[8\ 2\ 4]\ [5\ 8\ 5]\ [3\ 7\ 2]$  
    - $[9\ 4\ 5]\ [2\ 9\ 3]\ [1\ 4\ 4]$  

- **Converged!**
K-means Clustering: Examples

\[ k = 3 \]
\[ k = 5 \]
\[ k = 10 \]
K-means Properties

- Works best when clusters are spherical (blob like)
- Fails for elongated clusters
  - $J_{\text{SEE}}$ is not an appropriate objective function in this case
- Sensitive to outliers
K-means Summary

• Advantages
  • Principled (objective function) approach to clustering
  • Simple to implement
  • Fast

• Disadvantages
  • Only a local minimum is found
  • May fail for non-blob like clusters
  • Sensitive to initialization
  • Sensitive to choice of \( k \)
  • Sensitive to outliers
Can improve segmentation with more user strokes
But can we get a better initial result?
We are not using color information in the image effectively
FG Segmentation: Improving Data Term

- Data terms are 0 for most pixels
  - no preference to either foreground or background
- However
  - background strokes are mostly green
  - foreground strokes are mostly grey
- Can we push green non-seed pixels to prefer background?
- Can we push grey non-seed pixels to prefer foreground?
Current have:

\[ D_p(0) = 0 \]
\[ D_p(1) = 0 \]
\[ D_q(0) = 0 \]
\[ D_q(1) = 0 \]

Want to have:

\[ D_p(0) = \text{small} \]
\[ D_p(1) = \text{large} \]
\[ D_q(0) = \text{large} \]
\[ D_q(1) = \text{small} \]
FG Segmentation: Color Distributions

- Build color *distribution* from foreground seeds

- Build color *distribution* from background seeds
FG Segmentation: Color Distributions

• Build color **distribution** from foreground seeds

\[
P_{\text{foreground}}(\text{color}) = \frac{\text{number of foreground seeds of color}}{\text{total number of foreground seeds}}
\]

• Build color **distribution** from background seeds

• Normalized histogram for distribution
FG Segmentation: Color Distributions

- For green pixels $p$, $P_{\text{background}}(p)$ is high, $P_{\text{background}}(p)$ low
- We want just the opposite for the data term
- Convert to “opposite” using $-\log()$

\[
P_{\text{background}}(\text{color}) 
\quad \rightarrow \quad -\log P_{\text{background}}(\text{color})
\]

\[
P_{\text{foreground}}(\text{color}) 
\quad \rightarrow \quad -\log P_{\text{foreground}}(\text{color})
\]

- Do the same for the foreground
FG Segmentation: Color Distributions

- $D_p(\text{foreground}) = - \log P_{\text{foreground}}(\text{color of } p)$
- $D_p(\text{background}) = - \log P_{\text{background}}(\text{color of } p)$

Problem:

- The number of colors is too high: $256^3$
  - too large to build a normalized histogram
- Cluster colors using kmeans clustering, and treat each cluster as the “new” color
FG Segmentation: Cluster Colors

• Need to reduce number of colors
• Group similar colors together and treat the group as the same color
• 10 color clusters with k-means
  • cluster 1 = color 1
  • cluster 2 = color 2
  • ...
  • cluster 10 = color 10
• Now we only have 10 colors
• Build foreground/background color models over 10 “new” colors
FG Segmentation: Segmentation Result

user input

reduced colors

segmentation

foreground $D$

background $D$

blue pixels prefer foreground
red pixels prefer background