Lecture 15

Computer Vision

Grouping and Segmentation

Some slides are from S. Seitz, D. Jacobs, O. Camps, A. Torralba
• 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 \textit{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**
Gestalt Principles of Grouping

• Proximity
Gestalt Principles of Grouping

• Good continuation
Gestalt Principles of Grouping

- Connectivity
  - stronger than color
Gestalt Principles of Grouping

- 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

\[
\text{if } f(x,y) < T \text{ then pixel } (x,y) \text{ is background}
\]

\[
\text{if } f(x,y) \geq T \text{ then pixel } (x,y) \text{ is foreground}
\]

\[
T = 120 \quad T = 180 \quad T = 220
\]
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
- 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.

  \[
  D_{(x,y)}\text{(background)} = f(x,y) - 5 \\
  D_{(x,y)}\text{(foreground)} = 20 - f(x,y)
  \]

- Brown pixel prefers foreground, green prefers background.

![Input Image](input image)  
![Background Data Term](background data term)  
![Foreground Data Term](foreground data term)
FG Segmentation: Data Term

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

\[
E_{\text{data}} = 6 + 12 + 14 + 6 + 14 + 12 + 14 + 14 + 6 + 6 + 14 + 15 + 6 + 2 + 6 + 6 + 6 + 0 + 0 + 6 + 14 + 2 + 0 + 0
\]

\[
E_{\text{data}} = 73
\]
**FG Segmentation: Smoothness Term**

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

```
bad shape

17 discontinuities

E_{smooth} = 17
```

```
nice shape

8 discontinuities

E_{smooth} = 8
```

```
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

![Correct segmentation](image1)

- 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, \( \lambda = 10 \)

\[
\begin{align*}
E_{\text{data}} &= 73 \\
E_{\text{smooth}} &= 170 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 243
\end{align*}
\]
\[
\begin{align*}
E_{\text{data}} &= 83 \\
E_{\text{smooth}} &= 70 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 353
\end{align*}
\]
\[
\begin{align*}
E_{\text{data}} &= 97 \\
E_{\text{smooth}} &= 80 \\
E &= E_{\text{data}} + E_{\text{smooth}} = 177
\end{align*}
\]
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: $N$

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

- 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>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>u</td>
<td>w</td>
</tr>
<tr>
<td>y</td>
<td>h</td>
<td>z</td>
</tr>
</tbody>
</table>

pixel names

background $D$

| 9 | 3 | 1 |
| 6 | 12 | 14 |

foreground $D$

| 1 | 1 | 0 |
| 14 | 14 | 15 |

Segmentation $s$

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

$$= 9 + 12 + 1 + 1 + 1 + 0 + 1 + 0 + 1 + 0 + 0 + 1 + 1 = 57 + 6 = 63$$
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

- Good choice \( w_{pq} = \lambda \cdot e^{\frac{(f(p)-f(q))^2}{2\sigma^2}} \)

- 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

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

\[ E(s) = E(s)_{\text{data term as before}} + \]

\[ = 57 + 3 + 2 + 0 + 0 + 7 + 0 + 0 + 2 + 0 + 0 + 2 + 1 = 57 + 15 = 72 \]
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:
  ![Image 1](image1.png)  ![Image 2](image2.png)

- Add more seeds for correction:
  ![Image 3](image3.png)  ![Image 4](image4.png)
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

Supervised learning:
- Recall supervised learning

Unsupervised learning:
- Cluster similar samples into groups
  - think of clusters as ‘discovering’ labels
How does this Relate to Image Segmentation?

- Represent image pixels as feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$
  - 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

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

feature vectors for clustering based on color

[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]
```
How does this Relate to Image Segmentation?

input image

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

feature vectors for clustering based on color and image coordinates

\[
\begin{bmatrix}
9 & 4 & 2 & 0 & 0 \\
7 & 3 & 1 & 0 & 1 \\
8 & 6 & 8 & 0 & 2 \\
8 & 2 & 4 & 1 & 0 \\
5 & 8 & 5 & 1 & 1 \\
3 & 7 & 2 & 1 & 2 \\
9 & 4 & 5 & 2 & 0 \\
2 & 9 & 3 & 2 & 1 \\
1 & 4 & 4 & 2 & 2
\end{bmatrix}
\]
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

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

$$J_{SSE} = \sum D_1 + \sum D_2 + \sum D_3$$

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 $k = 2$

```
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

#### 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

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

\[
dist( [9\ 4\ 2] - [9\ 4\ 2] ) = 0 \implies [9\ 4\ 2] \text{ goes to pink cluster}
\]
\[
dist( [7\ 3\ 1] - [9\ 4\ 2] ) = (7-9)^2 + (3-4)^2 + (1-2)^2 = 6
\]
\[
dist( [7\ 3\ 1] - [5\ 8\ 5] ) = (7-5)^2 + (3-8)^2 + (1-5)^2 = 45
\]
\[
\begin{array}{ccc}
9 & 4 & 2 \\
8 & 2 & 4 \\
9 & 4 & 5 \\
\end{array}
\begin{array}{ccc}
7 & 3 & 1 \\
5 & 8 & 5 \\
1 & 4 & 4 \\
\end{array}
\begin{array}{ccc}
8 & 6 & 8 \\
3 & 7 & 2 \\
\end{array}
\]
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

$$\begin{align*}
\text{dist}( [9 \ 4 \ 2] - [9 \ 4 \ 2] ) &= 0 \Rightarrow \text{[9 \ 4 \ 2] goes to pink cluster} \\
\text{dist}( [7 \ 3 \ 1] - [9 \ 4 \ 2] ) &= (7-9)^2 + (3-4)^2 + (1-2)^2 = 6 \Rightarrow \text{[7 \ 3 \ 1] goes to pink cluster} \\
\text{dist}( [7 \ 3 \ 1] - [5 \ 8 \ 5] ) &= (7-5)^2 + (3-8)^2 + (1-5)^2 = 45 \\
\text{dist}( [8 \ 6 \ 8] - [9 \ 4 \ 2] ) &= (8-9)^2 + (6-4)^2 + (8-2)^2 = 41 \Rightarrow \text{[8 \ 6 \ 8] goes to blue cluster} \\
\text{dist}( [8 \ 6 \ 8] - [5 \ 8 \ 5] ) &= (8-5)^2 + (6-8)^2 + (8-5)^2 = 22
\end{align*}$$
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
  • repeat for the rest of feature vectors

\[
\begin{bmatrix}
8 & 2 & 4 \\
5 & 8 & 5 \\
3 & 7 & 2 \\
9 & 4 & 5 \\
2 & 9 & 3 \\
1 & 4 & 4 \\
\end{bmatrix}
\]
K-means Clustering: Example

- Iterate
  - compute cluster means

\[
\mu_1 = \frac{[9 \ 4 \ 2] + [7 \ 3 \ 1] + [8 \ 2 \ 4] + [9 \ 4 \ 5]}{4} = [8.25 \ 3.25 \ 3]
\]

\[
\mu_2 = \frac{[8 \ 6 \ 8] + [5 \ 8 \ 5] + [3 \ 7 \ 2] + [2 \ 9 \ 3] + [1 \ 4 \ 4]}{5} = [3.8 \ 6.8 \ 4.4]
\]
K-means Clustering: Example

• Iterate
  • compute cluster means
    \( \mathbf{\mu}_1 = [8.25 \ 3.25 \ 3] \)
    \( \mathbf{\mu}_2 = [3.8 \ 6.8 \ 4.4] \)
  • reassign samples to the closest mean

\[
\begin{align*}
\text{dist}( [9 \ 4 \ 2] - [8.25 \ 3.25 \ 3] ) &= (8.25-9)^2 + (3.25-4)^2 + (3-2)^2 \approx 2 \\
\text{dist}( [9 \ 4 \ 2] - [3.8 \ 6.8 \ 4.4] ) &= (3.8-9)^2 + (6.8-4)^2 + (4.4-2)^2 \approx 41
\end{align*}
\]

[9 4 2] 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**?
FG Segmentation: Improving Data Term

Currently 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

![Color distribution diagram for foreground seeds](image1)

• Build color *distribution* from background seeds

![Color distribution diagram for background seeds](image2)
FG Segmentation: Color Distributions

- Build color \textit{distribution} from foreground seeds

- Build color \textit{distribution} from background seeds

- Normalized histogram for distribution

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

- 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 kmeans
  - 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

clusters visualized with random colors

pixels painted with average color of pixels in its cluster
FG Segmentation: Segmentation Result

user input

reduced colors

segmentation

foreground $D$

background $D$

blue pixels prefer foreground
red pixels prefer background