CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 15

Computer Vision

Grouping and Segmentation

Some slides are from S. Seitz, D. Jacobs, O. Camps, A. Torralba

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

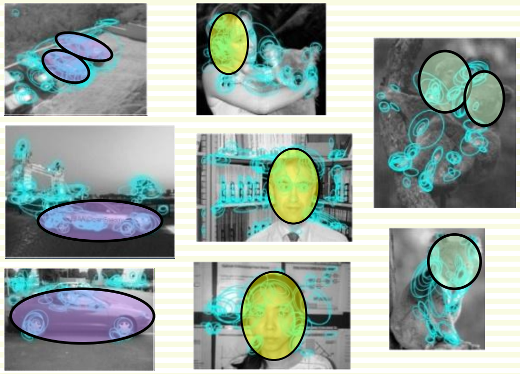
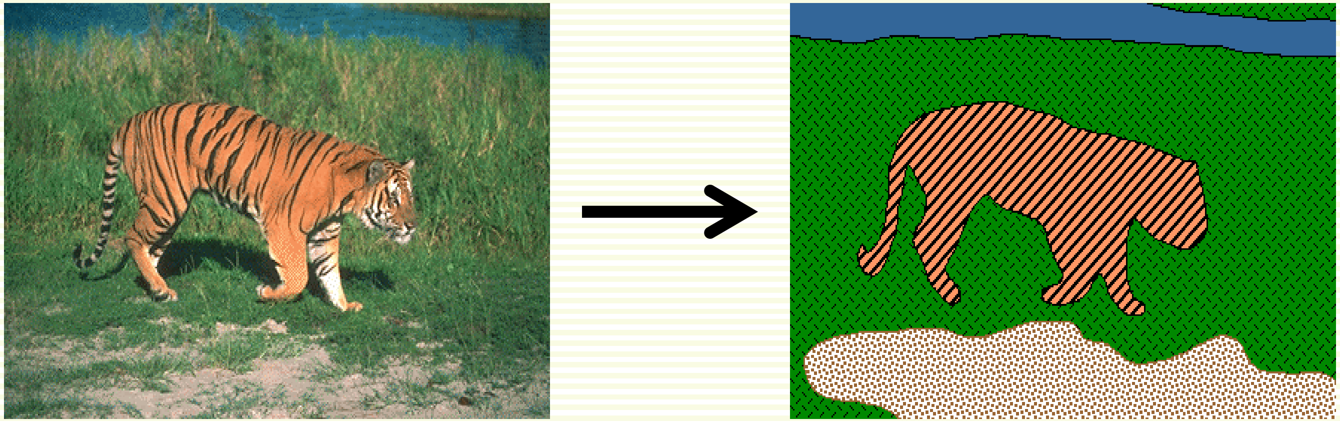


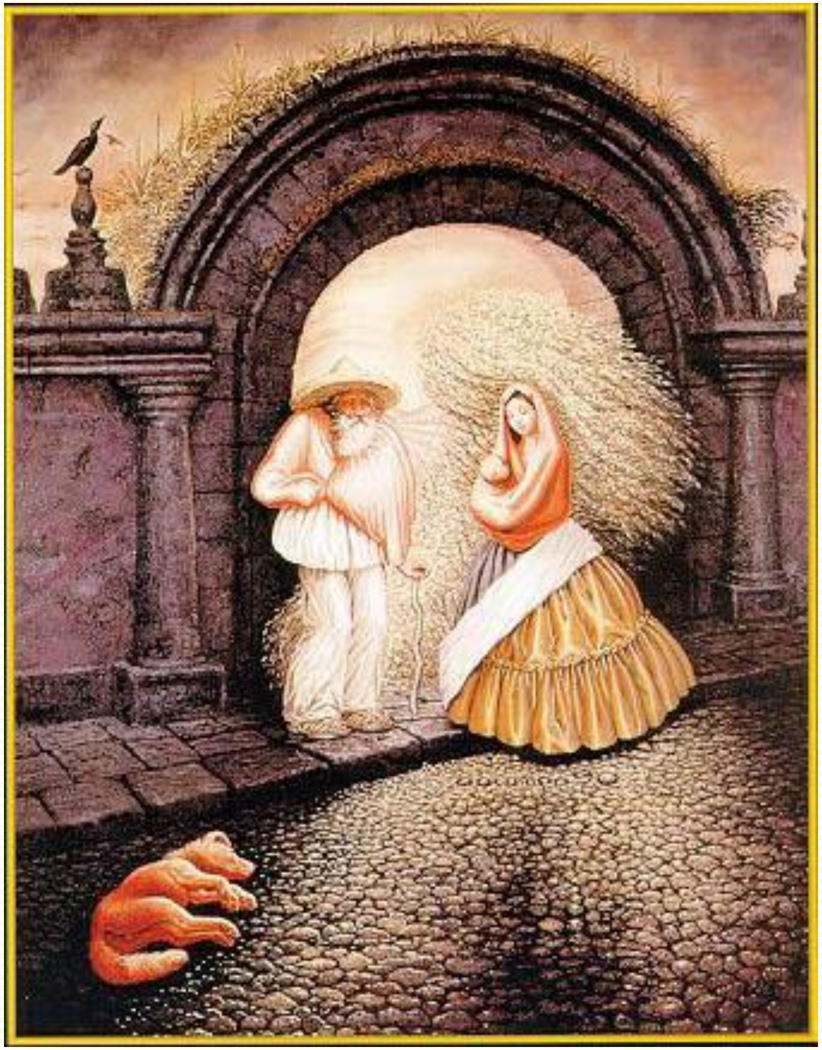
Image Segmentation



- 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

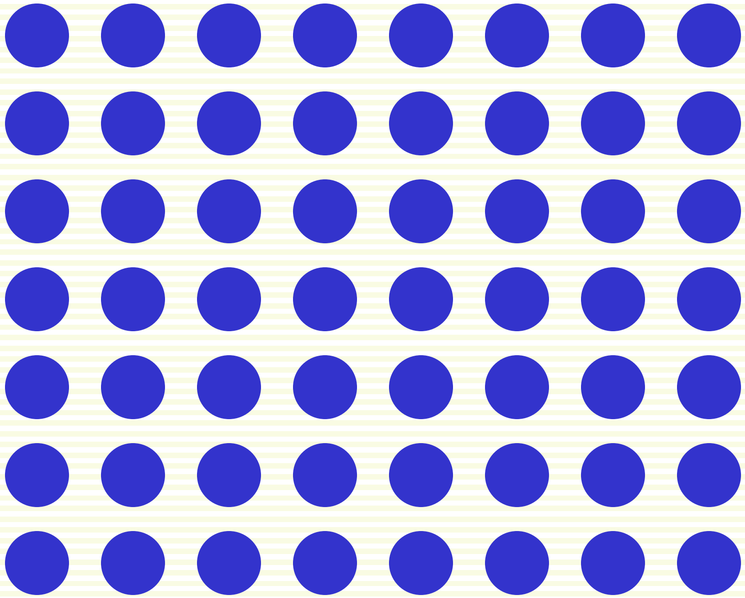
Gestalt Psychology

- 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**



Grouping

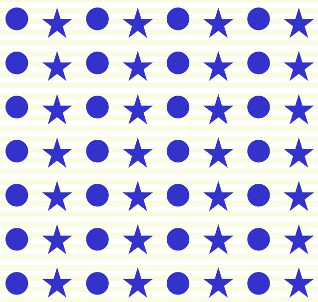
- Most human observers report no particular grouping



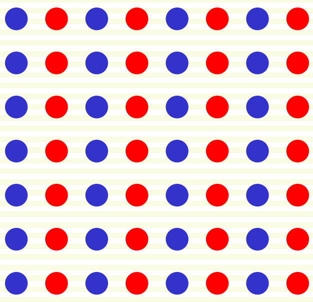
Gestalt Principles of Grouping

- Common form, includes:

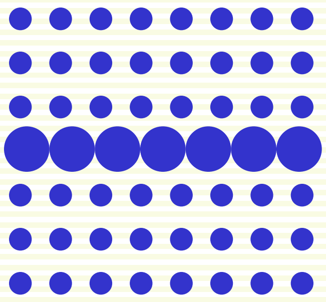
shape



color

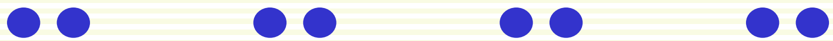


size



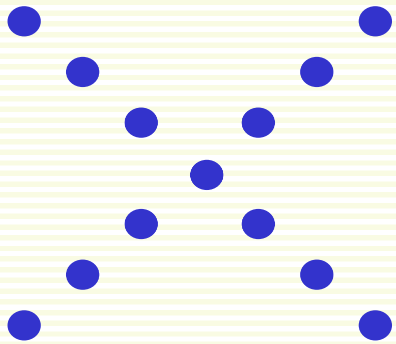
Gestalt Principles of Grouping

- Proximity



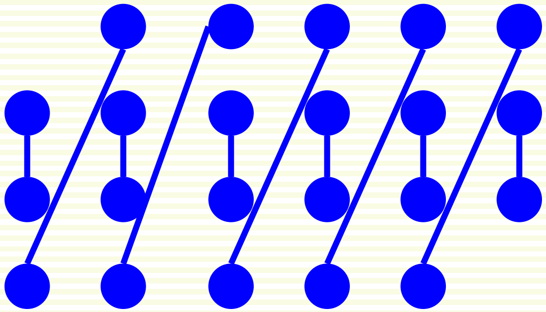
Gestalt Principles of Grouping

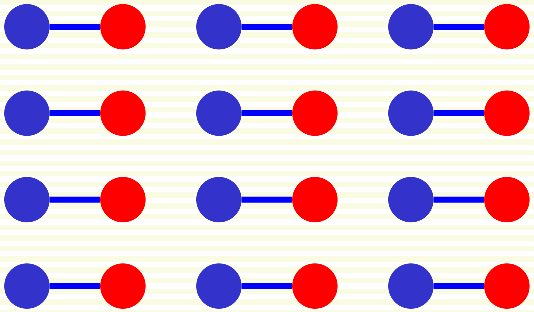
- Good continuation



Gestalt Principles of Grouping

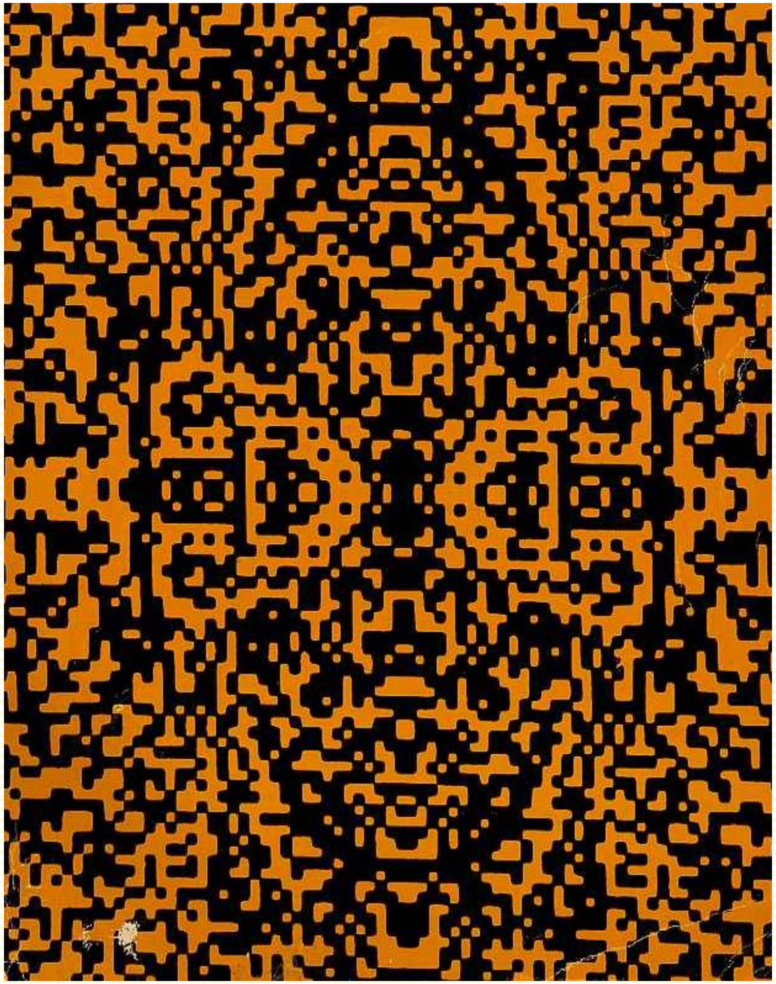
- Connectivity
  - stronger than color





Gestalt Principles of Grouping

- Symmetry

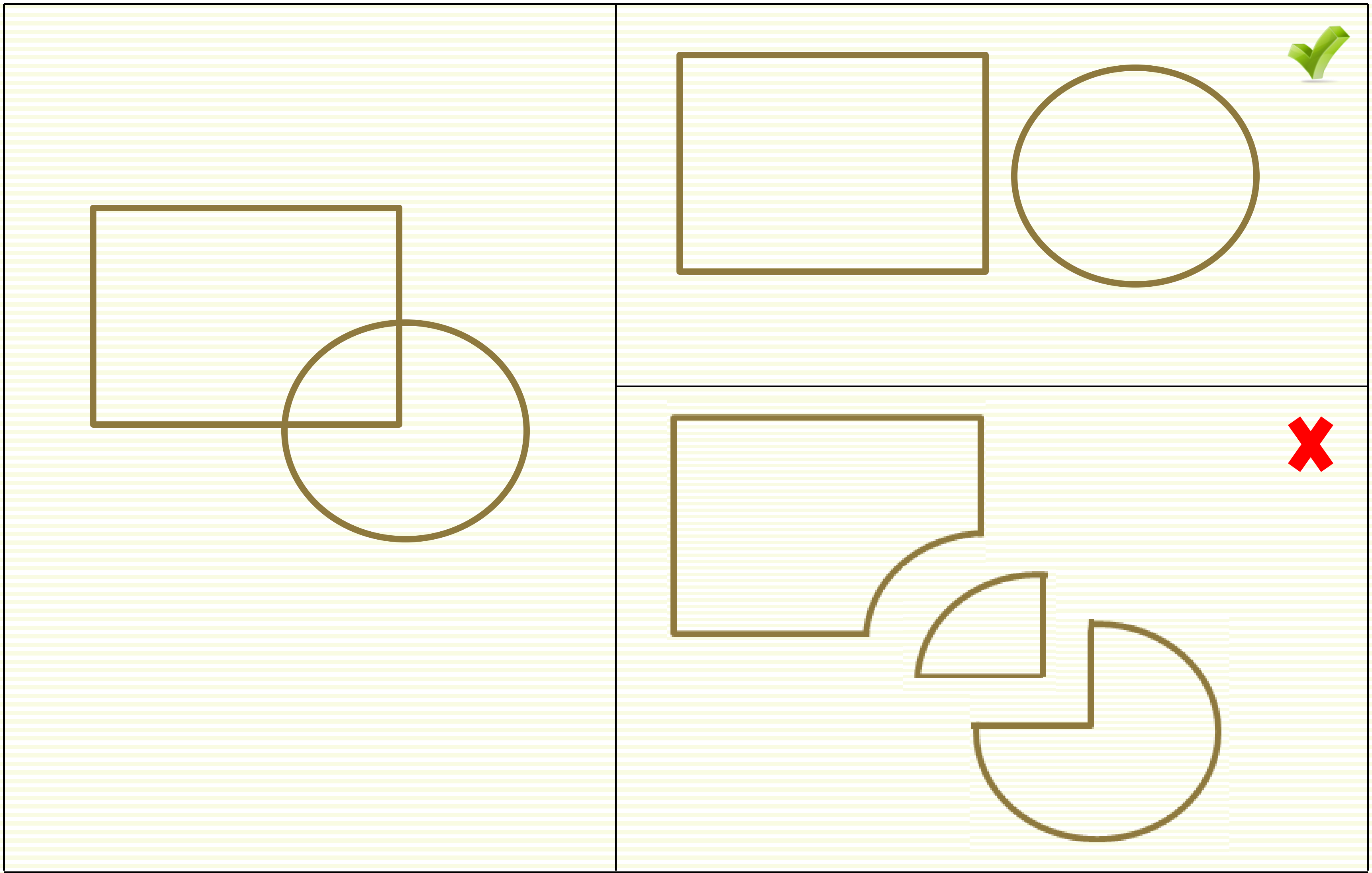


Gestalt Principles of Grouping

- Familiarity

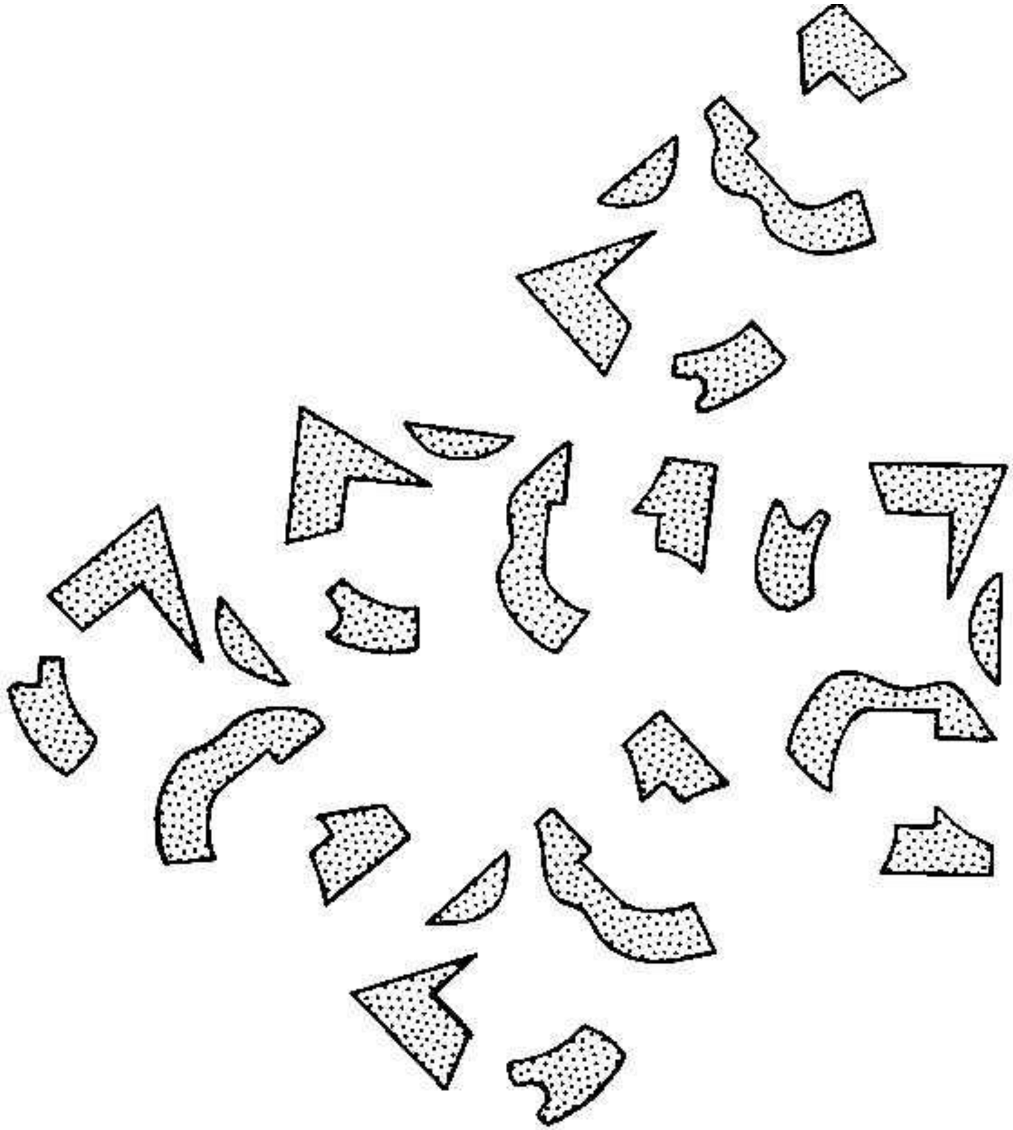
Gestalt Principles of Grouping

- Closure



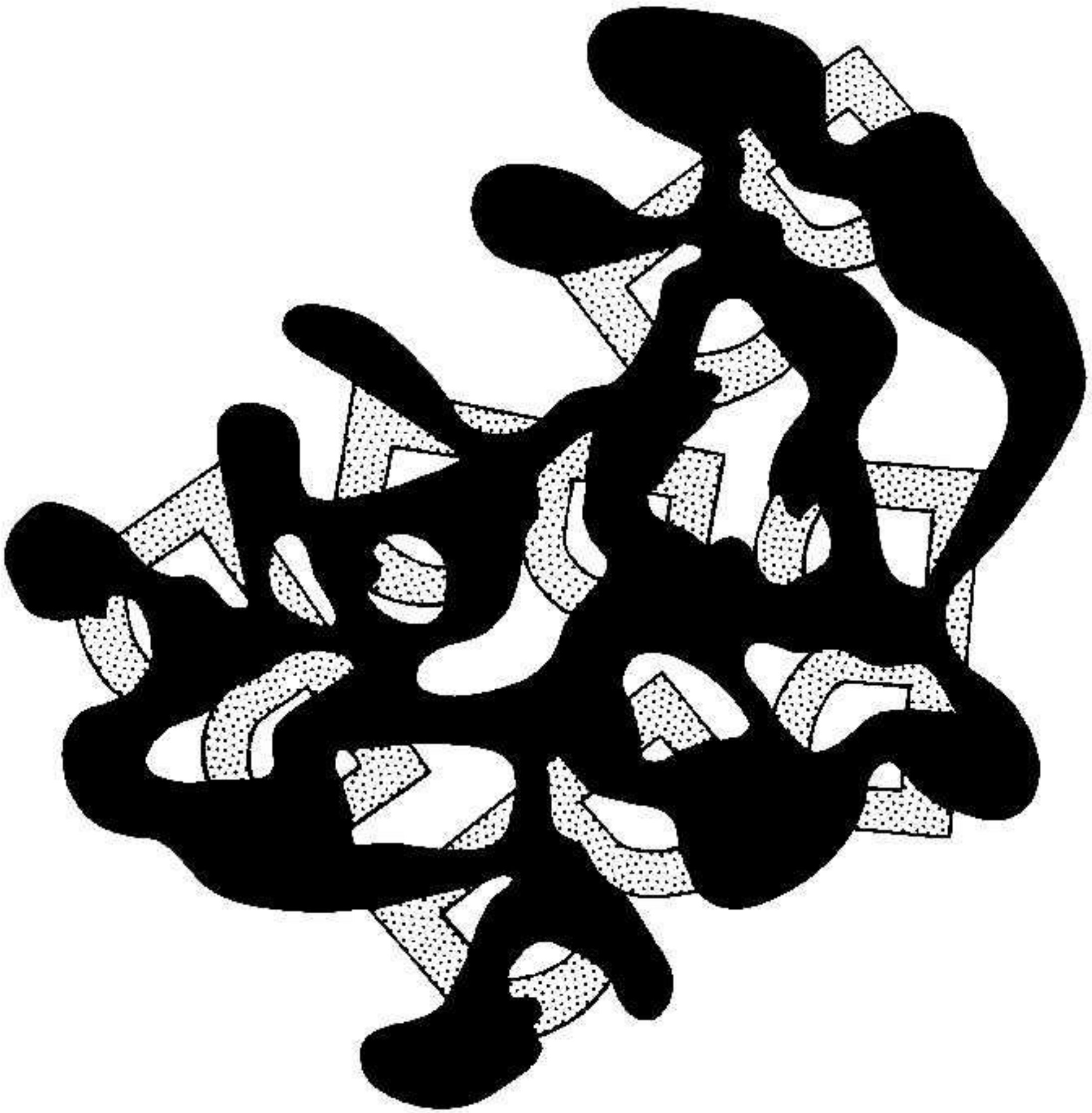
Gestalt Principles of Grouping

- Closure



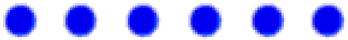
Gestalt Principles of Grouping

- Closure



Gestalt Principles of Grouping

- Common fate



Gestalt Principles of Grouping

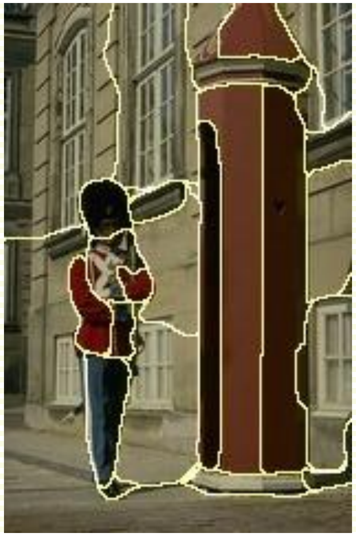
- Higher level knowledge?

Gestalt Principles of Grouping

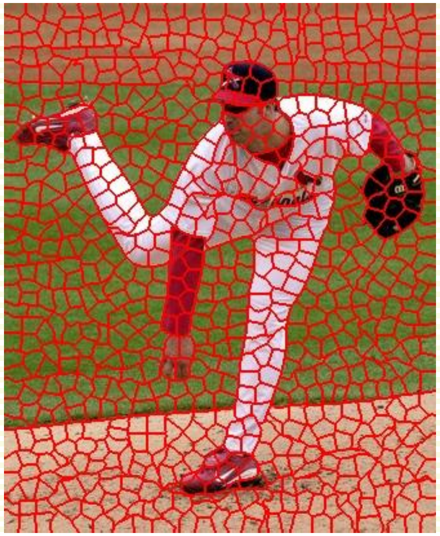
- Many other Gestalt grouping principles
  - parallelism, convexity, colinearity, common depth, etc.
- Gestalt principles are an inspiration to computer vision
  - they seem to rely on nature of objects in the world, most do not involve higher level knowledge (object recognition)
  - should help to segment 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

Image Segmentation

- Many types of image segmentation



regions



superpixels

figure-ground

- 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 **T = 180** **T = 220**

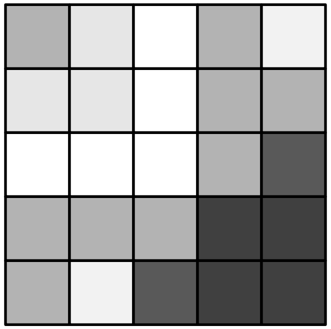
FG Segmentation: Thresholding

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

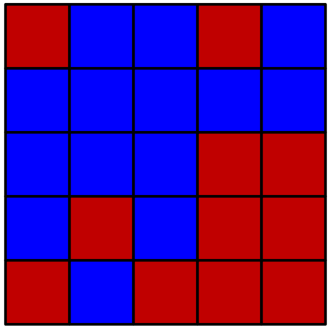
- Can we clean this result up?

FG Segmentation: Motivation

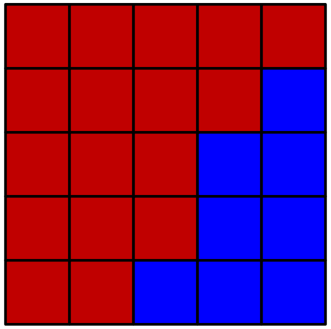
- Know object is light, background is dark
- Do not know object shape
  - show background with red, foreground with blue



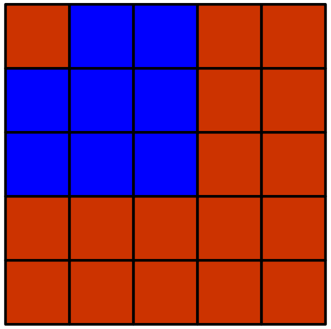
input image



bad result: crazy object shape



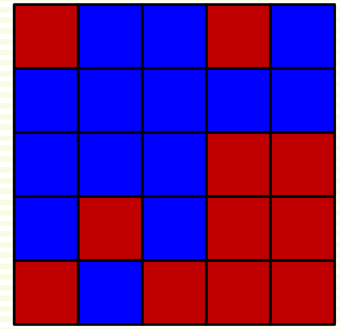
bad result: object is dark, background light

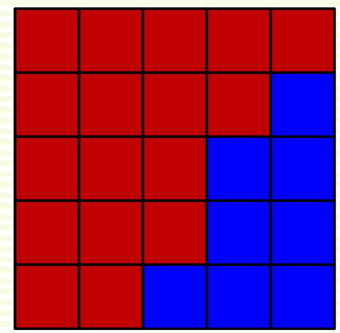


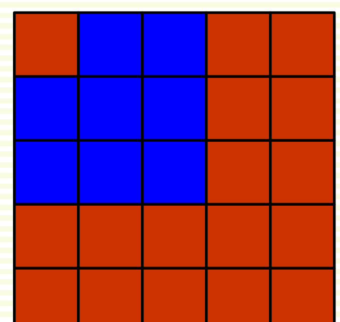
good result: light object of good shape, dark background

FG Segmentation: Energy Function

- 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

E()= high

E()=high

E()=low

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

10	17	19	10	19
17	17	19	10	10
19	19	19	10	7
10	10	10	5	5
10	19	7	5	5

input image f

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 $\boldsymbol{D}_{(\boldsymbol{x},\boldsymbol{y})}$(background) to assign it to background and $\boldsymbol{D}_{(\boldsymbol{x},\boldsymbol{y})}$(foreground) to assign it to the foreground
- Let the smallest image intensity be 5, and largest 20

$$\boldsymbol{D}_{(x,y)}(\text{background}) = f(\boldsymbol{x},\boldsymbol{y}) - 5$$

$$\boldsymbol{D}_{(x,y)}(\text{foreground}) = 20 - f(\boldsymbol{x},\boldsymbol{y})$$

11	17	19	11	19
17	19	19	11	11
19	19	20	11	7
11	11	11	5	5
11	19	7	5	5

input image *f*

6	12	14	6	14
12	14	14	6	6
14	14	15	6	2
6	6	6	0	0
6	14	2	0	0

background data term *D*

9	3	1	9	1
3	1	1	9	9
1	1	0	9	13
9	9	9	15	15
9	1	13	15	15

foreground data term *D*

- **Brown** pixel prefers foreground, **green** prefers background

FG Segmentation: Data Term

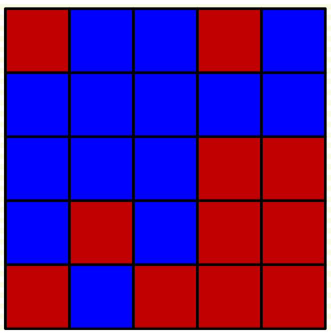
- E_{data} sums data $D_{(x,y)}$ term over all pixels (x,y)
- Foreground blue, background red

6	12	14	6	14
12	14	14	6	6
14	14	15	6	2
6	6	6	0	0
6	14	2	0	0

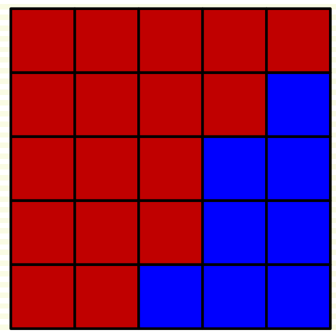
background *D*

9	3	1	9	1
3	1	1	9	9
1	1	0	9	13
9	9	9	15	15
9	1	13	15	15

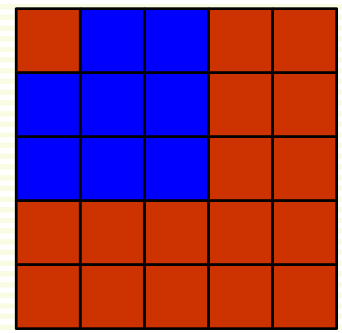
foreground *D*



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



$E_{data} = 283$

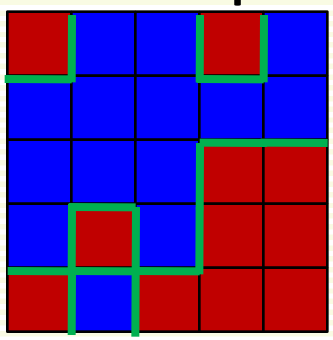


$E_{data} = 97$

FG Segmentation: Smoothness Term

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

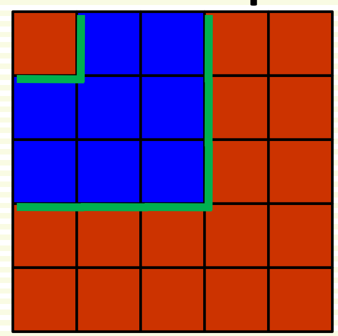
bad shape



17 discontinuities

$\mathbf{E}_{\mathbf{smooth}} = 17$

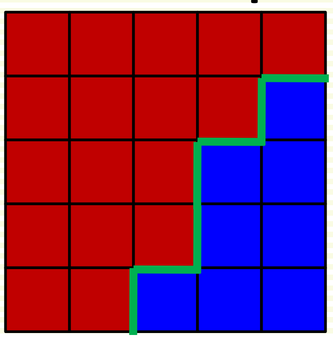
nice shape



8 discontinuities

$\mathbf{E}_{\mathbf{smooth}} = 8$

nice shape



7 discontinuities

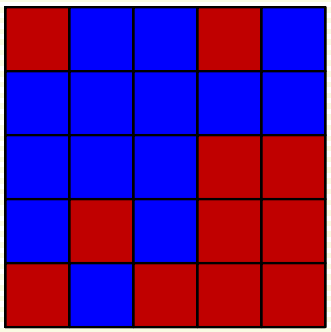
$\mathbf{E}_{\mathbf{smooth}} = 7$

- discontinuity: when two nearby pixels are in different segments
- smoothness term is the number of discontinuities

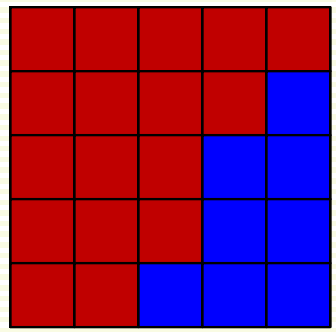
FG Segmentation: Total Energy

- Now combine both data and smoothness energy terms

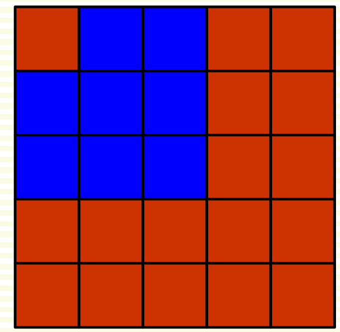
best



E_{data} = 73
E_{smooth} = 17
$E = E_{data} + E_{smooth}$ = 90



E_{data} = 283
E_{smooth} = 7
$E = E_{data} + E_{smooth}$ = 290



E_{data} = 97
E_{smooth} = 8
$E = E_{data} + E_{smooth}$ = 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

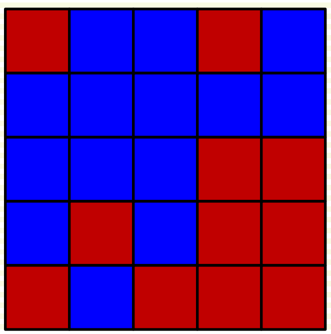
FG Segmentation: Total Energy

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

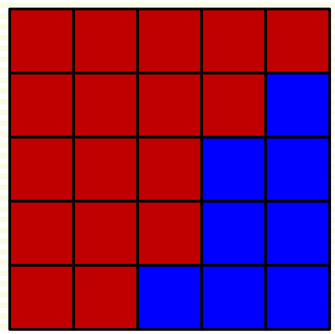
$$\mathbf{E = E_{data} + \lambda\, E_{smooth}}$$

- Take, for example, $\lambda = 10$

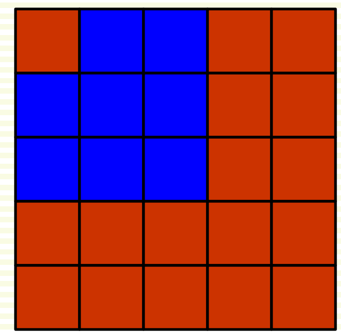
best



$\mathbf{E_{data}} = 73$
$\mathbf{E_{smooth}} = 170$
$\mathbf{E = E_{data} + E_{smooth}} = 243$



$\mathbf{E_{data}} = 83$
$\mathbf{E_{smooth}} = 70$
$\mathbf{E = E_{data} + E_{smooth}} = 353$



$\mathbf{E_{data}} = 97$
$\mathbf{E_{smooth}} = 80$
$\mathbf{E = E_{data} + E_{smooth}} = 177$

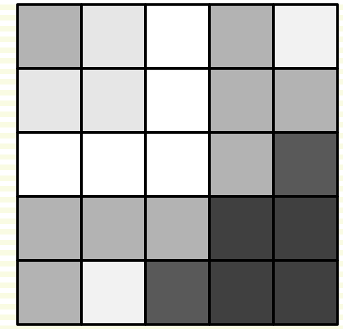
FG Segmentation: Energy Formula

- 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: $\boldsymbol{N}$



input image ***f***

0	1	1	0	1
1	1	1	1	1
1	1	1	0	0
1	0	1	0	0
0	1	0	0	0

segmentation ***s***

p	***q***	***r***
v	***u***	***w***
y	***h***	***z***

$\boldsymbol{N} = \{ (\boldsymbol{p},\boldsymbol{q}), (\boldsymbol{p},\boldsymbol{r}), (\boldsymbol{v},\boldsymbol{u}), (\boldsymbol{u},\boldsymbol{w}), (\boldsymbol{y},\boldsymbol{h}), (\boldsymbol{h},\boldsymbol{z}), (\boldsymbol{p},\boldsymbol{v}), (\boldsymbol{v},\boldsymbol{y}), (\boldsymbol{q},\boldsymbol{u}), (\boldsymbol{u},\boldsymbol{h}), (\boldsymbol{r},\boldsymbol{w}), (\boldsymbol{w},\boldsymbol{z}) \}$

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

- where [true] = 1, [false] = 0

FG Segmentation: Formula Practice with $\lambda = 1$

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

pixel names

p	*q*	*r*
v	*u*	*w*
y	*h*	*z*

background *D*

9	3	1
3	1	1
1	1	0

foreground *D*

6	12	14
12	14	14
14	14	15

$$E\left(\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 1 & 1 \\ \hline\end{array}\right) = \begin{array}{l} D_p(0) + D_q(1) + D_r(0) \\ D_v(0) + D_u(0) + D_w(0) \\ D_y(0) + D_h(1) + D_z(1) \end{array} + \begin{array}{l} [s_p \neq s_q] + [s_q \neq s_r] + [s_v \neq s_u] \\ [s_u \neq s_w] + [s_y \neq s_h] + [s_h \neq s_z] \\ [s_p \neq s_v] + [s_q \neq s_u] + [s_r \neq s_w] \\ [s_v \neq s_y] + [s_u \neq s_h] + [s_w \neq s_z] \end{array}$$

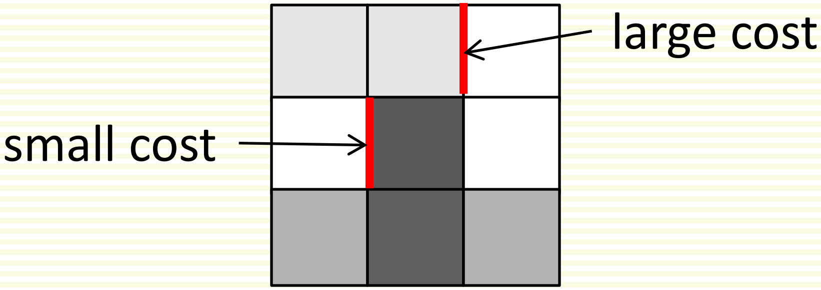
segmentation *s*

$$= \begin{array}{l} 9 + 12 + 1 \\ 3 + \ \ 1 + 1 \\ 1 + 14 + 15 \end{array} + \begin{array}{l} 1 + 1 + 0 \\ 0 + 1 + 0 \\ 0 + 1 + 0 \\ 0 + 1 + 1 \end{array} = 57 + 6 = 63$$

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 $[\boldsymbol{s}_p \neq \boldsymbol{s}_q]$ with $\boldsymbol{w}_{pq} \cdot [\boldsymbol{s}_p \neq \boldsymbol{s}_q]$ where $\boldsymbol{w}_{pq}$ 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 σ^2 is either
  - set by hand (trail and error)
  - or computed as average of $(f(p)-f(q))^2$ over all neighbors in $\boldsymbol{N}$
- Complete energy:
  - note that is now folded into $\boldsymbol{w}_{pq}$

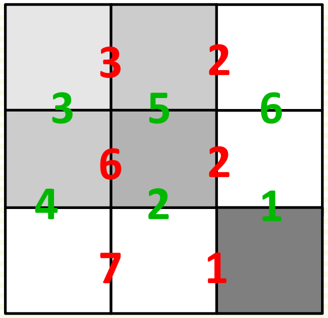
$$E(s) = \sum_p D_p(s_p) + \sum_{(p,q)\in N} w_{pq}[s_p \neq s_q]$$

FG Segmentation: Example

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

p	*q*	*r*
v	*u*	*w*
y	*h*	*z*

pixel names



contrast sensitive weights

$$\boldsymbol{E}\left(\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 1 & 1 \\ \hline \end{array}\right) = \text{data term as before} + \begin{array}{l} 3\cdot[\boldsymbol{s}_p \neq \boldsymbol{s}_q] + 2\cdot[\boldsymbol{s}_q \neq \boldsymbol{s}_r] + 6\cdot[\boldsymbol{s}_v \neq \boldsymbol{s}_u] \\ 2\cdot[\boldsymbol{s}_u \neq \boldsymbol{s}_w] + 7\cdot[\boldsymbol{s}_y \neq \boldsymbol{s}_h] + 1\cdot[\boldsymbol{s}_h \neq \boldsymbol{s}_z] \\ 3\cdot[\boldsymbol{s}_p \neq \boldsymbol{s}_v] + 2\cdot[\boldsymbol{s}_q \neq \boldsymbol{s}_u] + 6\cdot[\boldsymbol{s}_r \neq \boldsymbol{s}_w] \\ 4\cdot[\boldsymbol{s}_v \neq \boldsymbol{s}_y] + 2\cdot[\boldsymbol{s}_u \neq \boldsymbol{s}_h] + 1\cdot[\boldsymbol{s}_w \neq \boldsymbol{s}_z] \end{array}$$

segmentation ***s***

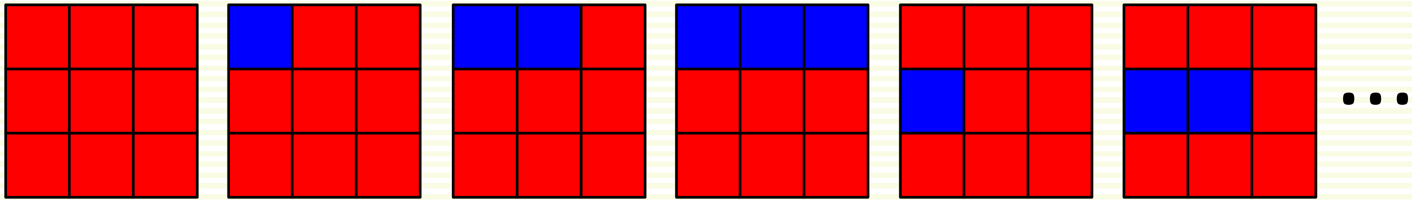
$$= \; 57 + \begin{array}{c} 3+2+0 \\ 0+7+0 \\ 0+2+0 \\ 0+2+1 \end{array} = \; 57 + 15 = 72$$

FG Segmentation: Optimization

- We are all set to find the best segmentation $\boldsymbol{s}^*$

$$\boldsymbol{s}^* = \arg\min_{\boldsymbol{s}} \boldsymbol{E}(\boldsymbol{s})$$

- 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

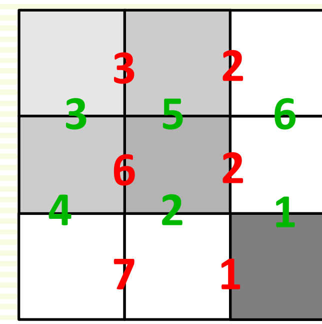
- Build weighted graph
  - one node per pixel
    - connect to neighbor pixel nodes with weight $\boldsymbol{w}_{pq}$
  - two special nodes (terminals) source ***s***, sink ***t***
    - ***s*** connects to each pixel node ***p*** with weight $\boldsymbol{D}_p(0)$
    - ***t*** connects to each pixel node ***p*** with weight $\boldsymbol{D}_p(1)$
    - graph below omits most of these edges for clarity

9	3	1
3	1	1
1	1	0

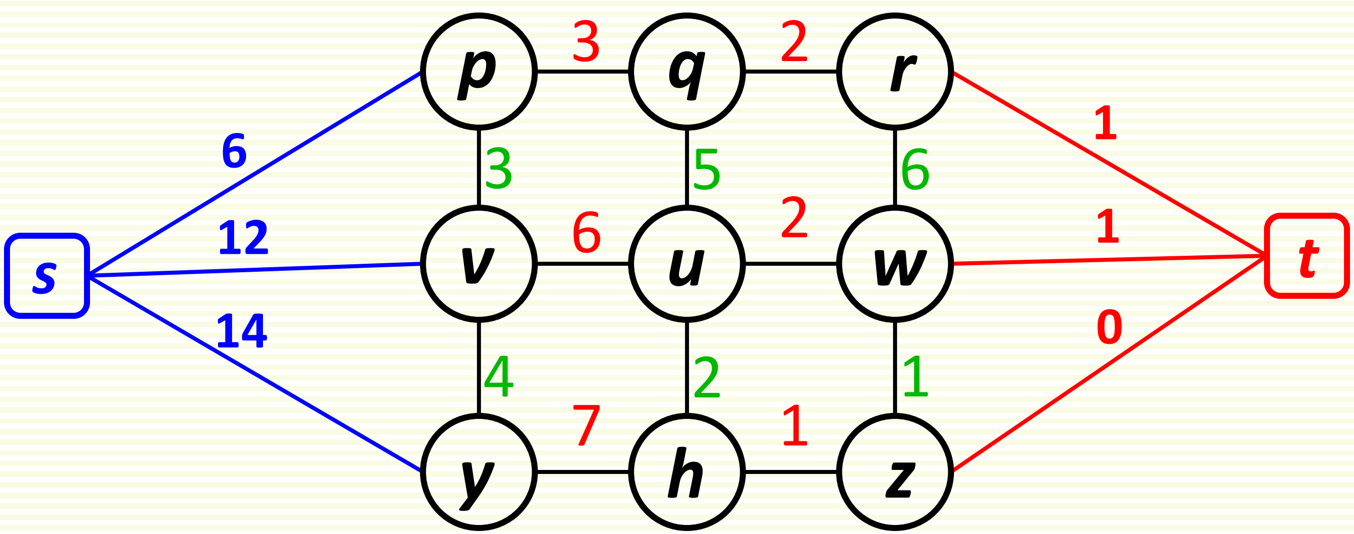
foreground *D*

6	12	14
12	14	14
14	14	15

background *D*



contrast sensitive weights

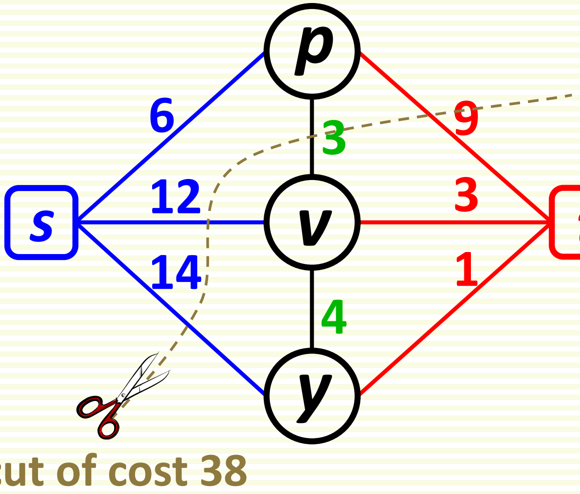


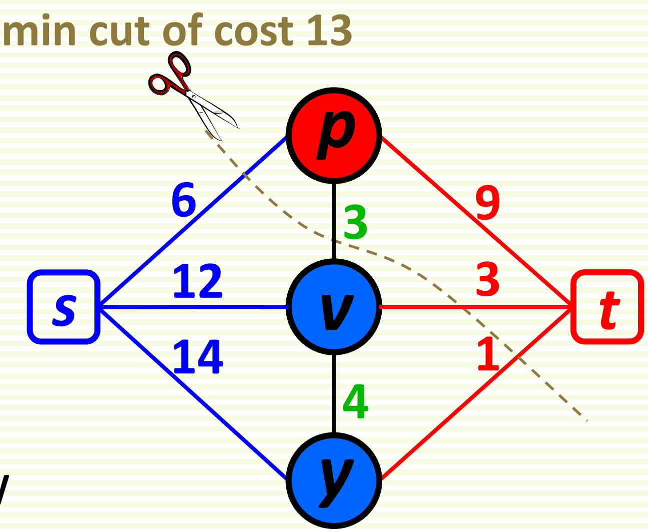
p	*q*	*r*
v	*u*	*w*
y	*h*	*z*

pixel names

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





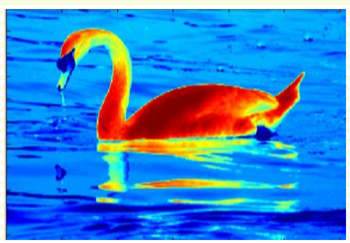
FG Segmentation: Segmentation Result

input

segmentation

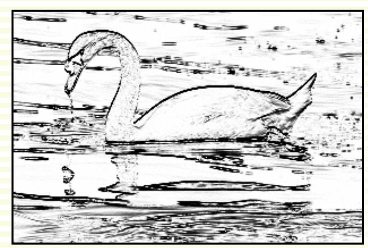
- Data terms
  - blue means low weight, red high weight

foreground

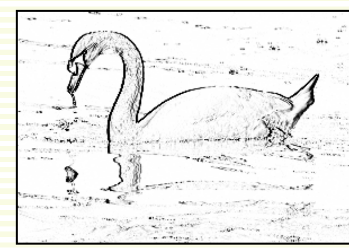


background

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



horizontal



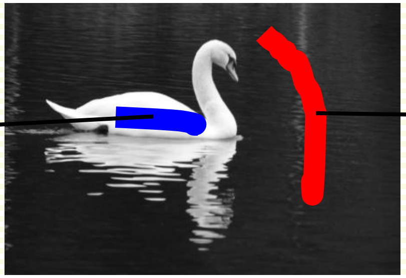
vertical

FG Segmentation: Interactive

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

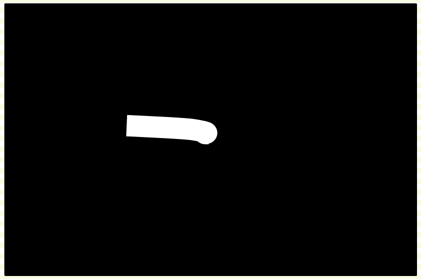
$D_p(1) = 0$

$D_p(0) = \infty$

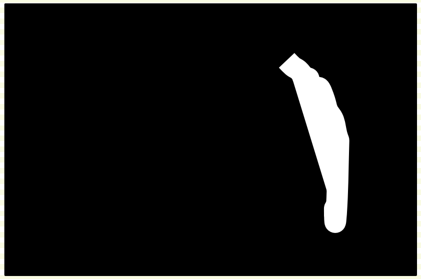


$D_p(1) = \infty$

$D_p(0) = 0$



background *D*

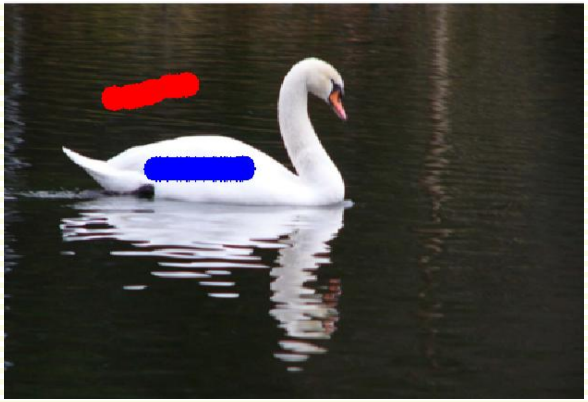


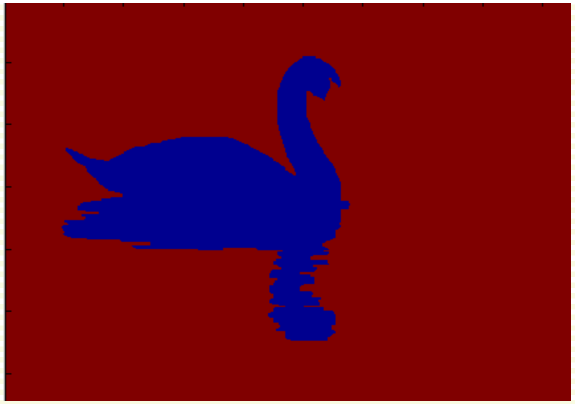
foreground *D*

- 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

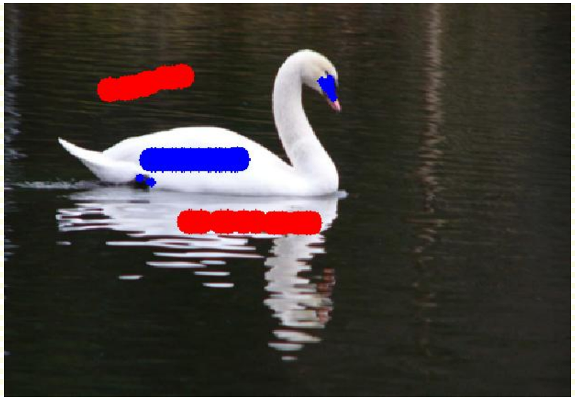
FG Segmentation: Interactive Results

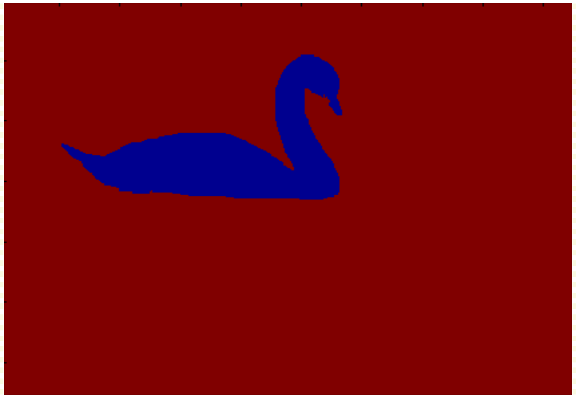
- Initial seeds:



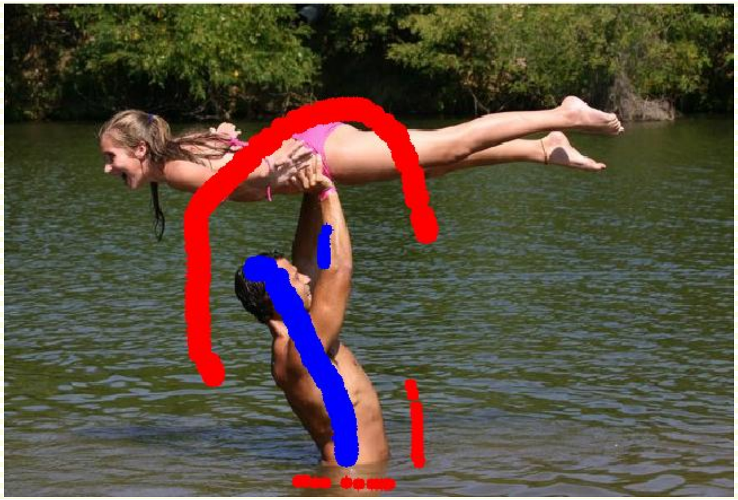


- Add more seeds for correction:



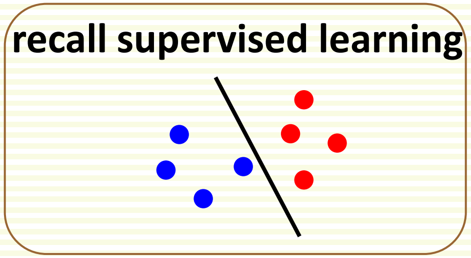


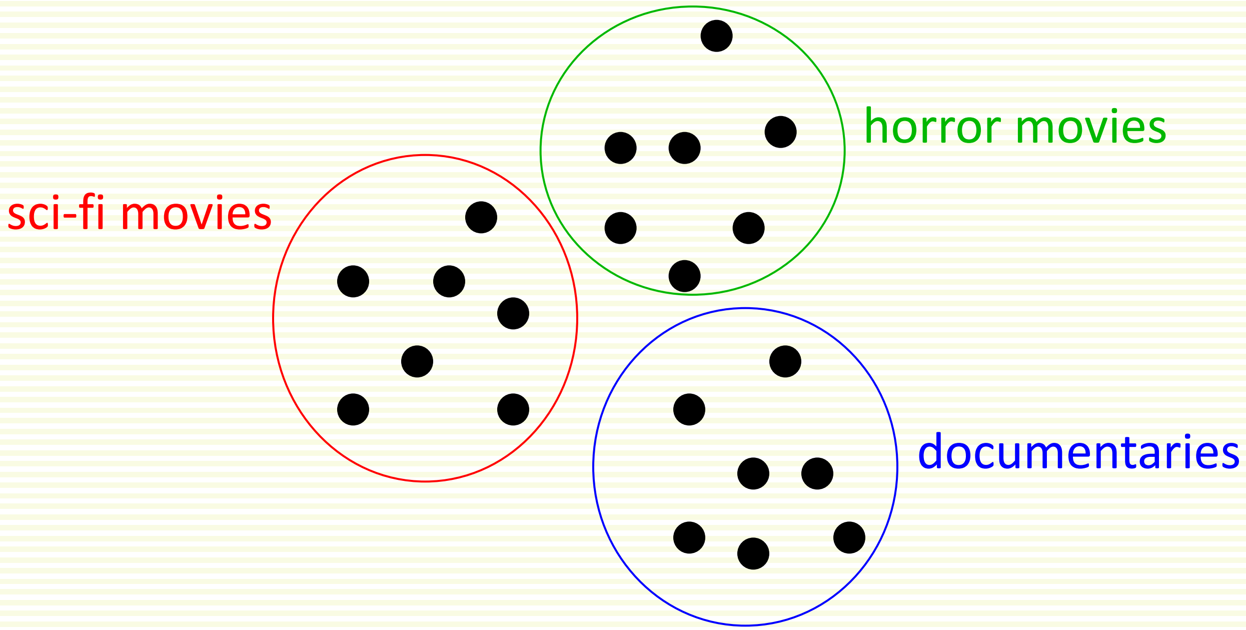
FG Segmentation: More Interactive Results



General Grouping or Clustering

- General Clustering (Grouping)
- Have samples (also called feature vectors, examples, etc.) $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$
- Cluster similar samples into groups
- This is also called unsupervised learning
  - samples have no labels
  - think of clusters as 'discovering' labels





How does this Relate to Image Segmentation?

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

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

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

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

feature vectors for clustering based on color and image coordinates

[9 4 2 0 0] [7 3 1 0 1] [8 6 8 0 2]

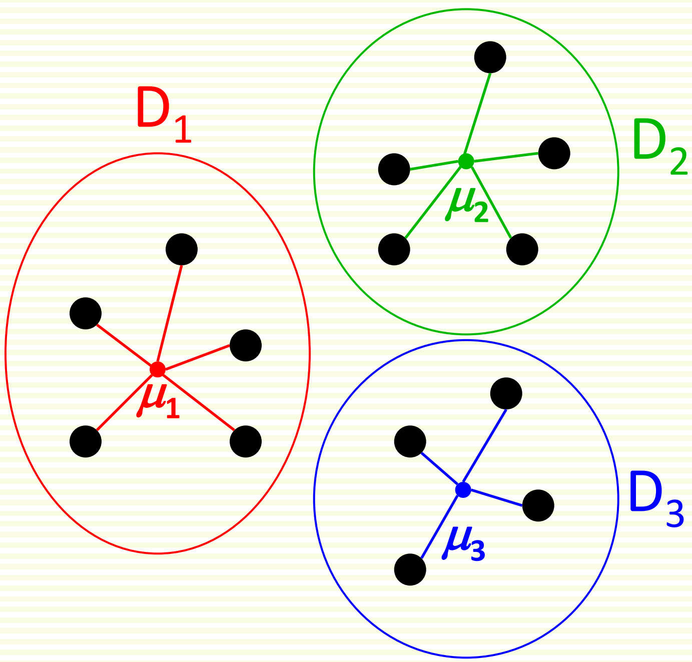
[8 2 4 1 0] [5 8 5 1 1] [3 7 2 1 2]

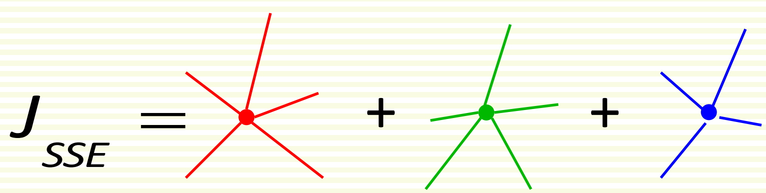
[9 4 5 2 0] [2 9 3 2 1] [1 4 4 2 2]

K-means Clustering: Objective Function

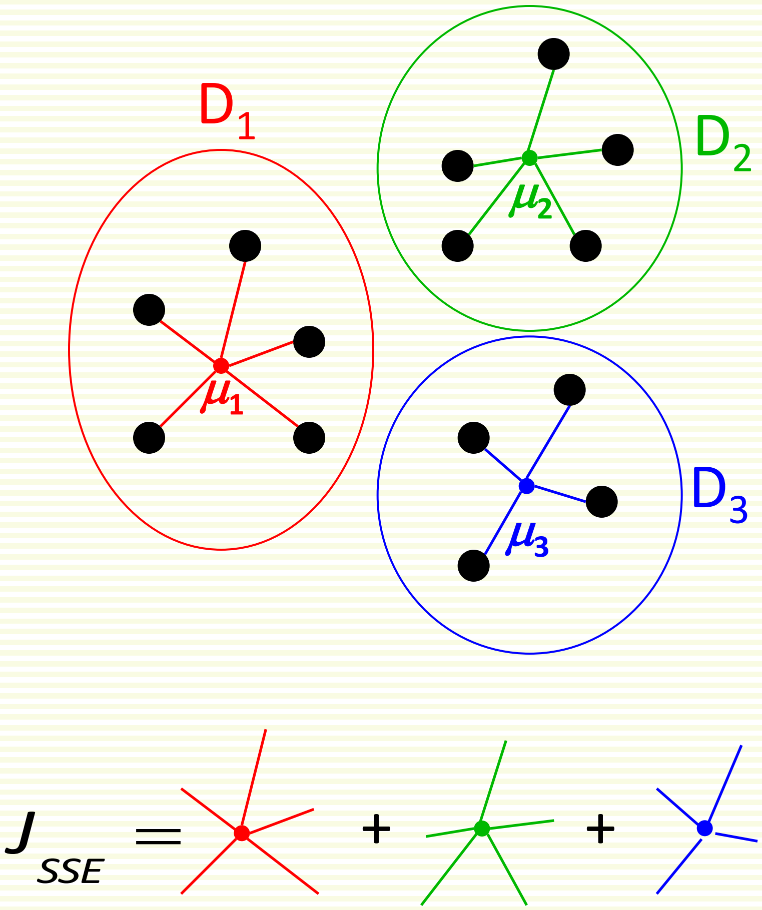
- Probably the most popular clustering algorithm
  - assumes know the number of clusters should be ***k***
- Optimizes (approximately) the following objective function

$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} \left\| x - \mu_i \right\|^2$$

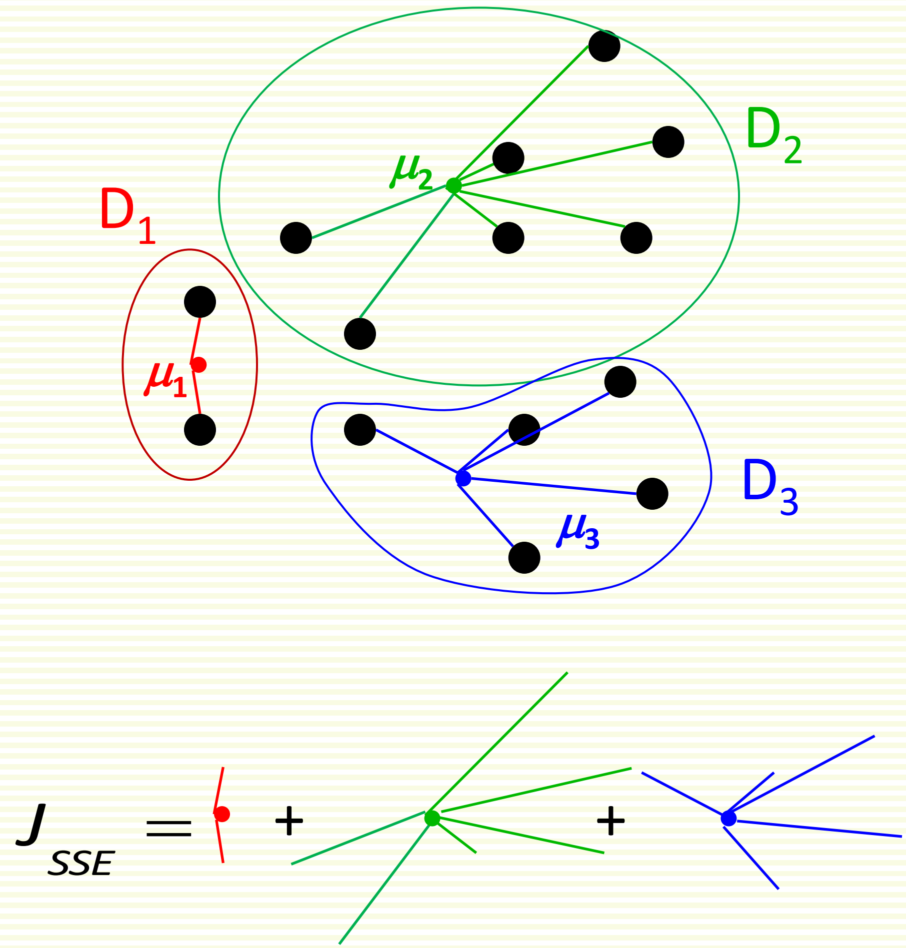




K-means Clustering: Objective Function



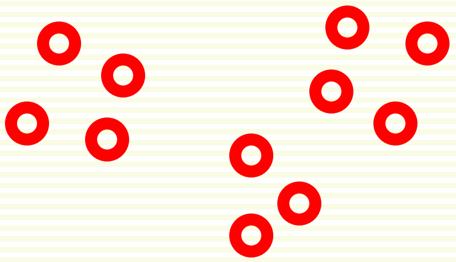
Good (tight) clustering
smaller value of J_{SSE}



Bad (loose) clustering
larger value of J_{SSE}

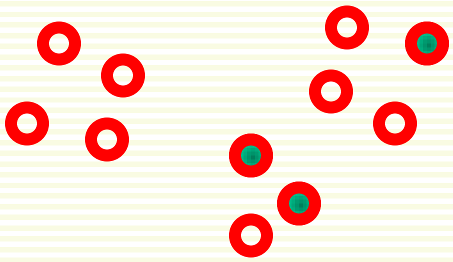
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly

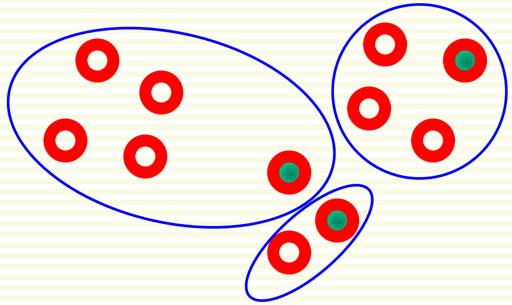


K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly



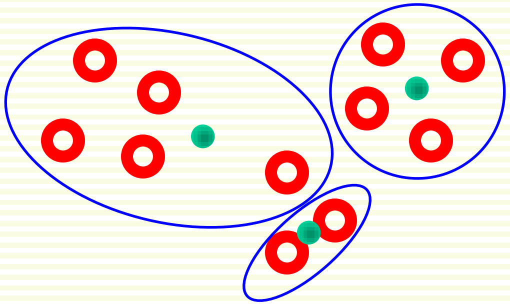
K-means Clustering: Algorithm



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

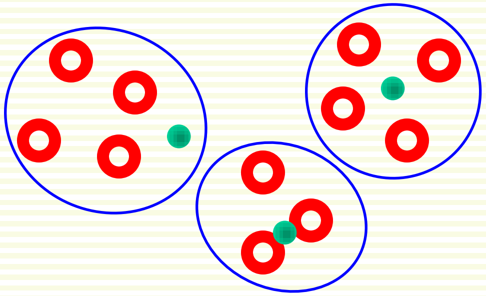
K-means Clustering: Algorithm

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



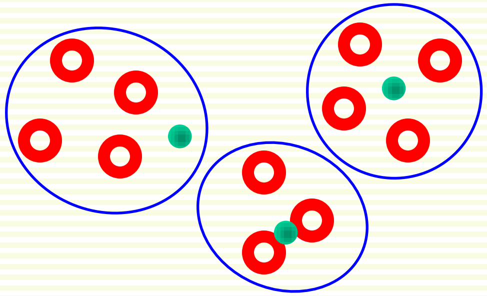
- Iteration step
  1. compute means in each cluster

K-means Clustering: Algorithm



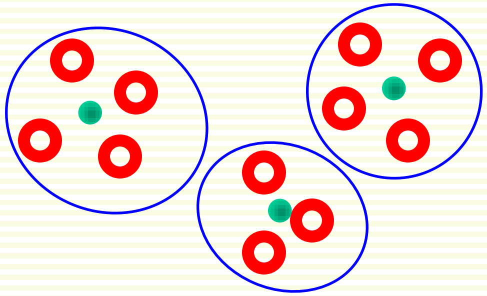
- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration step
  1. compute means in each cluster
  2. re-assign each sample to the closest mean

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



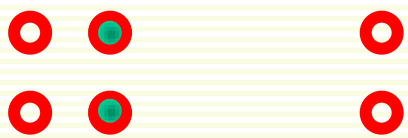
- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration step
  1. compute means in each cluster
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing

- Can prove that this procedure decreases the value of the objective function J_{SEE}

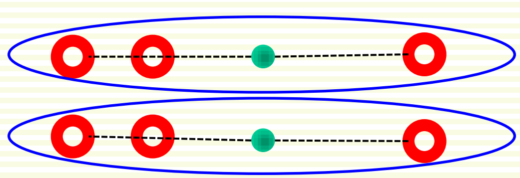
K-means: Approximate Optimization

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

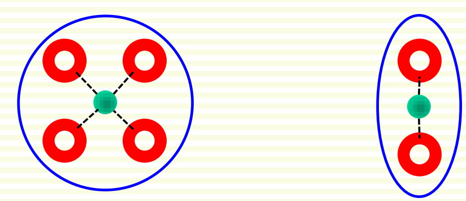
initialization



converged to local min



global minimum



K-means Clustering: Example

- with ***k*** = 2

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

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

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

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

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

dist([9 4 2] - [9 4 2]) = 0 $\Rightarrow$ [9 4 2] 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

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

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

dist([7 3 1] - [9 4 2]) = $(7\text{-}9)^2 + (3\text{-}4)^2 + (1\text{-}2)^2 = 6$

dist([7 3 1] – [5 8 5]) = $(7\text{-}5)^2 + (3\text{-}8)^2 + (1\text{-}5)^2 = 45$

[7 3 1] 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

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

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

dist([7 3 1] - [9 4 2]) = $(7\text{-}9)^2 + (3\text{-}4)^2 + (1\text{-}2)^2$ = 6

dist([7 3 1] – [5 8 5]) = $(7\text{-}5)^2 + (3\text{-}8)^2 + (1\text{-}5)^2$ = 45

[7 3 1] goes to pink cluster

dist([8 6 8] - [9 4 2]) = $(8\text{-}9)^2 + (6\text{-}4)^2 + (8\text{-}2)^2$ = 41

dist([8 6 8] – [5 8 5]) = $(8\text{-}5)^2 + (6\text{-}8)^2 + (8\text{-}5)^2$ = 22

[8 6 8] goes to blue 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
  - repeat for the rest of feature vectors

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

    [9 4 5] [2 9 3] [1 4 4]

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

initial clustering

K-means Clustering: Example

- Iterate
  - compute cluster means

initial clustering

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

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

$$\mu_2 = \frac{[8\ \ 6\ \ 8] + [5\ \ 8\ \ 5] + [3\ \ 7\ \ 2] + [2\ \ 9\ \ 3] + [1\ \ 4\ \ 4]}{5} = [3.8\ \ 6.8\ \ 4.4]$$

K-means Clustering: Example

- Iterate
  - compute cluster means

    μ_1 = [8.25 3.25 3]

    μ_2 = [3.8 6.8 4.4]

  - reassign samples to the closest mean

initial clustering

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

dist([9 4 2] - [8.25 3.25 3]) = $(8.25-9)^2 + (3.25-4)^2 + (3-2)^2 \approx 2$

dist([9 4 2] – [3.8 6.8 4.4]) = $(3.8-9)^2 + (6.8-4)^2 + (4.4-2)^2 \approx 41$

[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

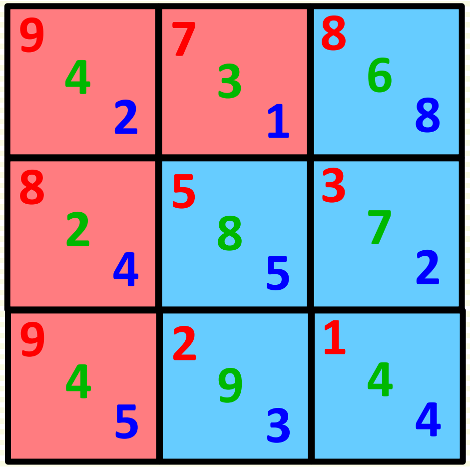
      [7 3 1] [8 6 8]

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

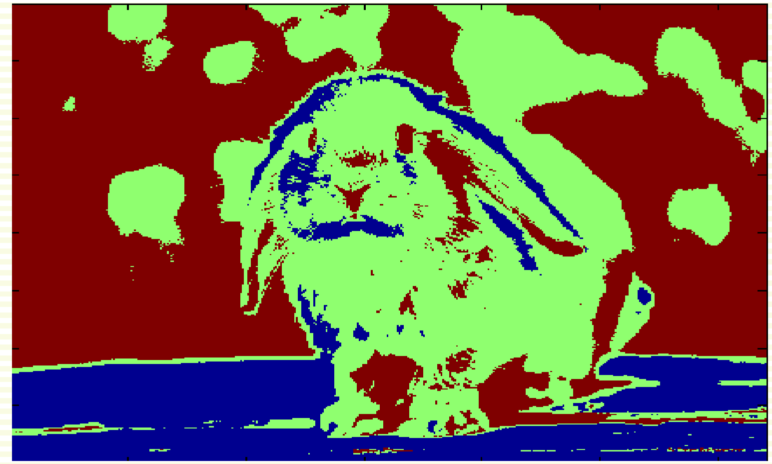
      [9 4 5] [2 9 3] [1 4 4]

- Converged!

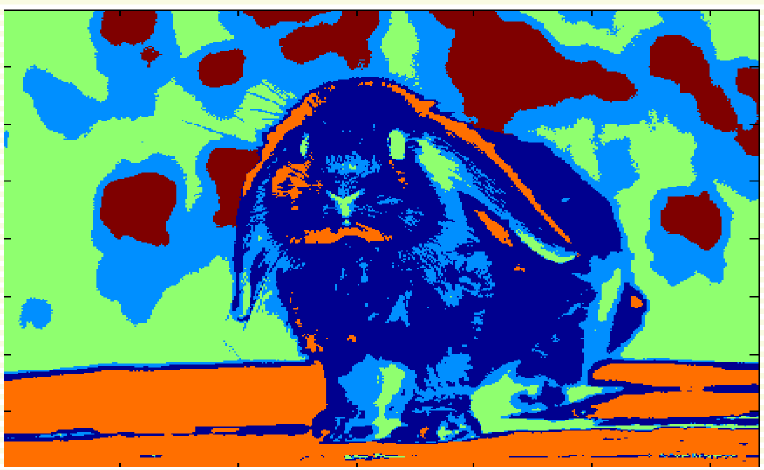
initial clustering



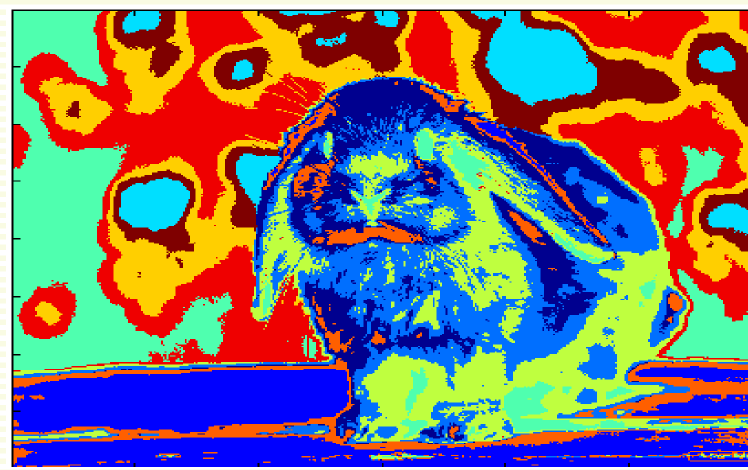
K-means Clustering: Examples



k = 3



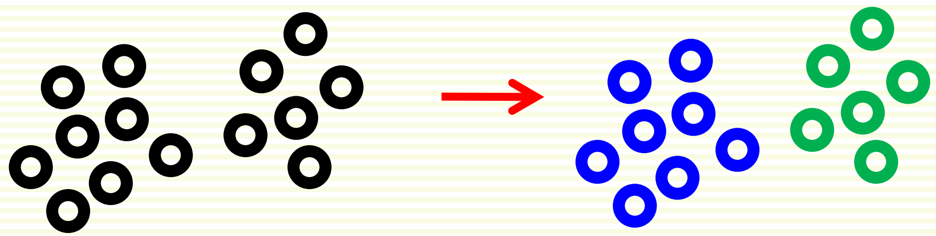
k = 5



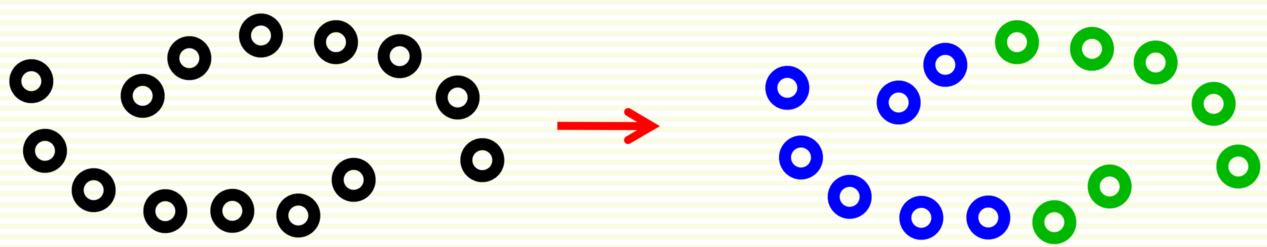
k = 10

K-means Properties

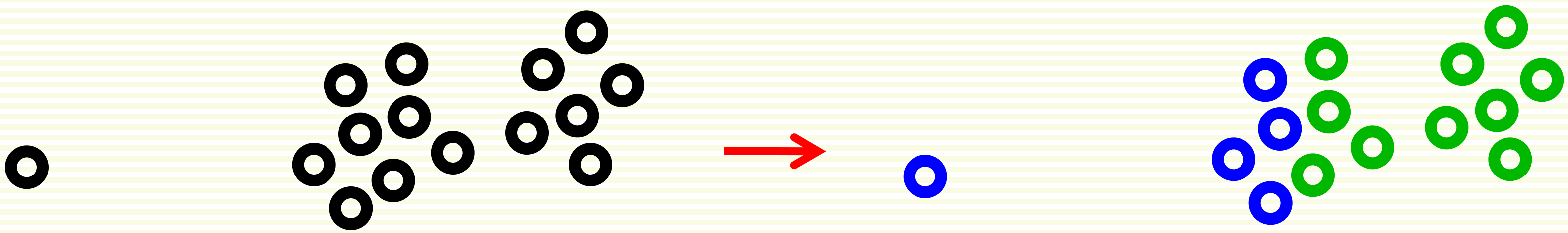
- Works best when clusters are spherical (blob like)



- Fails for elongated clusters
  - J_{SEE} is not an appropriate objective function in this case



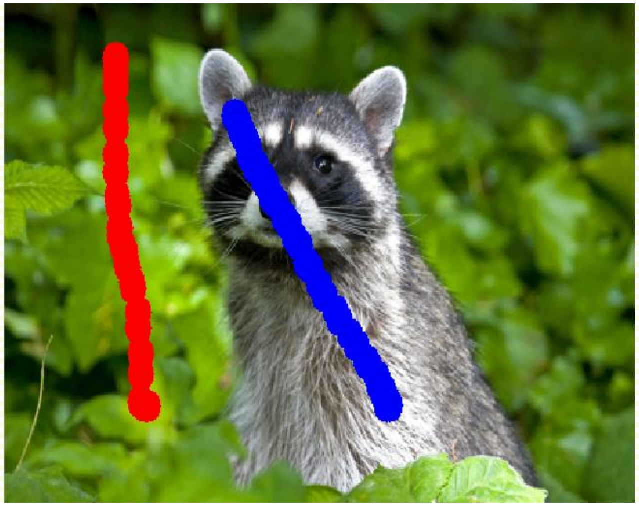
- Sensitive to outliers



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

Back to FG Segmentation: Improving Data Term

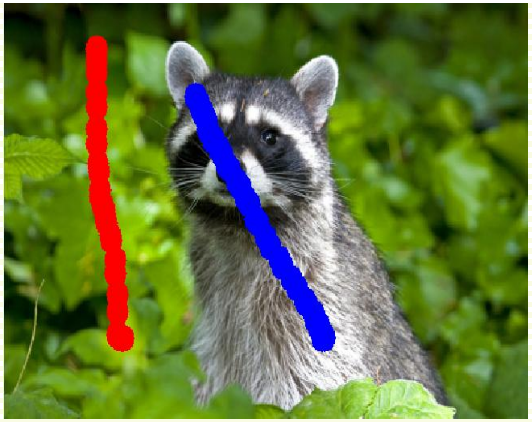


user strokes

initial result

- 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

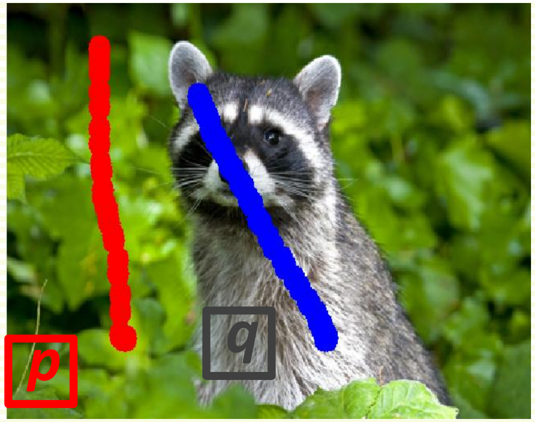


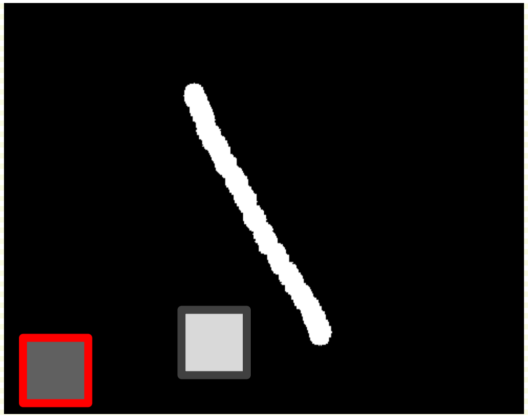
foreground D

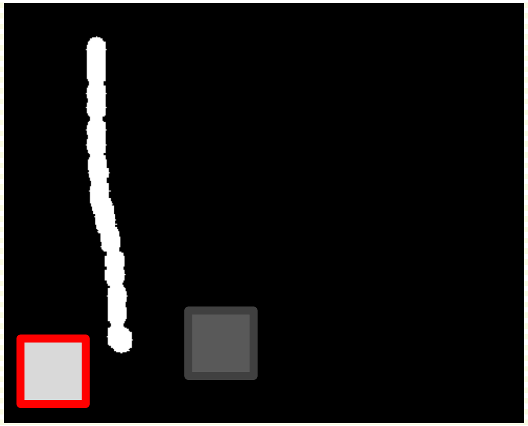
background D

- 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:

$\boldsymbol{D}_p(0) = 0$

$\boldsymbol{D}_p(1) = 0$

$\boldsymbol{D}_q(0) = 0$

$\boldsymbol{D}_q(1) = 0$

Want to have:

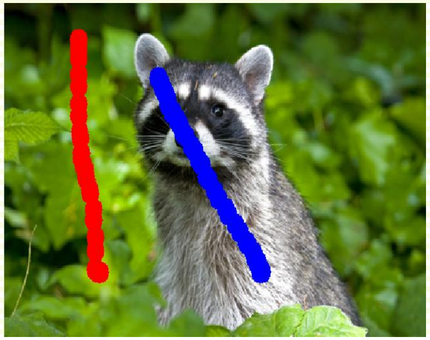
$\boldsymbol{D}_p(0) = \text{small}$

$\boldsymbol{D}_p(1) = \text{large}$

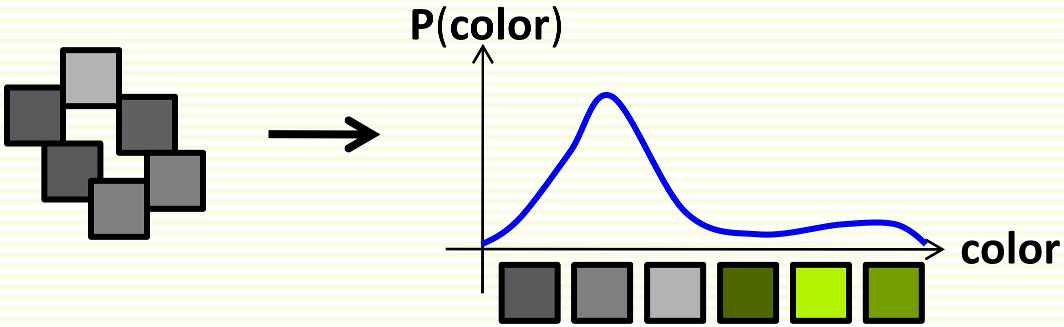
$\boldsymbol{D}_q(0) = \text{large}$

$\boldsymbol{D}_q(1) = \text{small}$

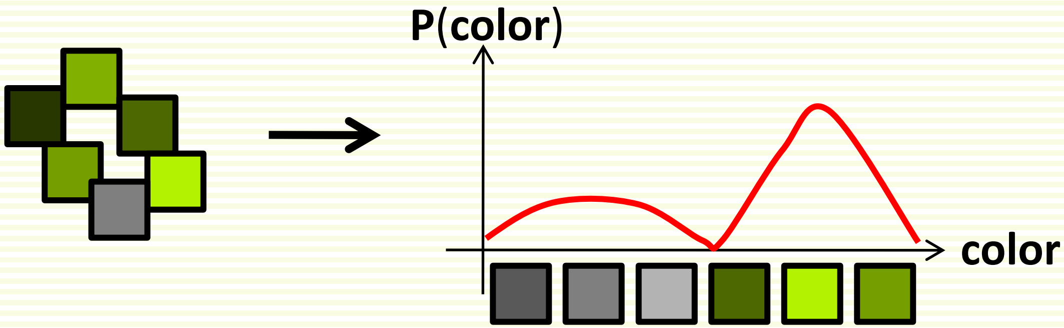
FG Segmentation: Color Distributions



- Build color ***distribution*** from foreground seeds

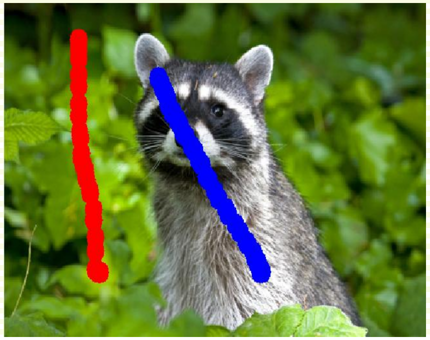


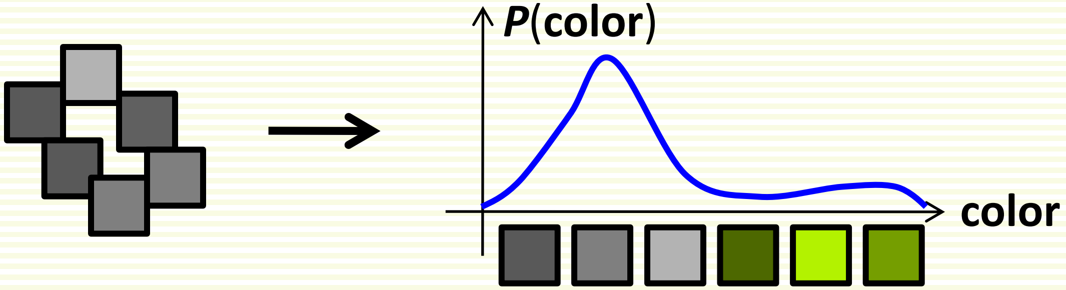
- Build color ***distribution*** from background seeds



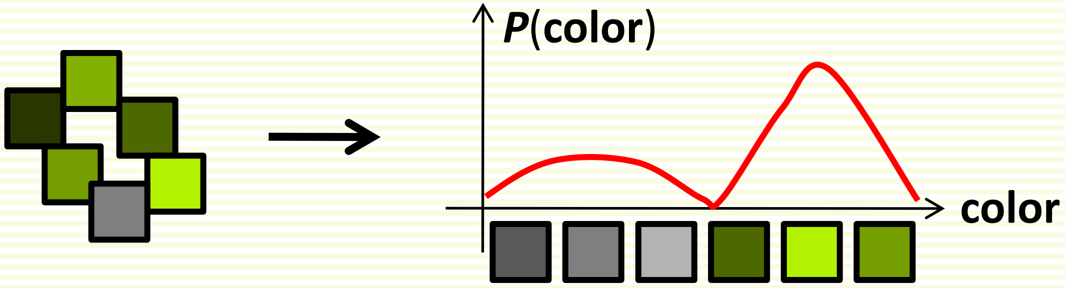
FG Segmentation: Color Distributions

- Build color ***distribution*** from foreground seeds





- Build color ***distribution*** from background seeds

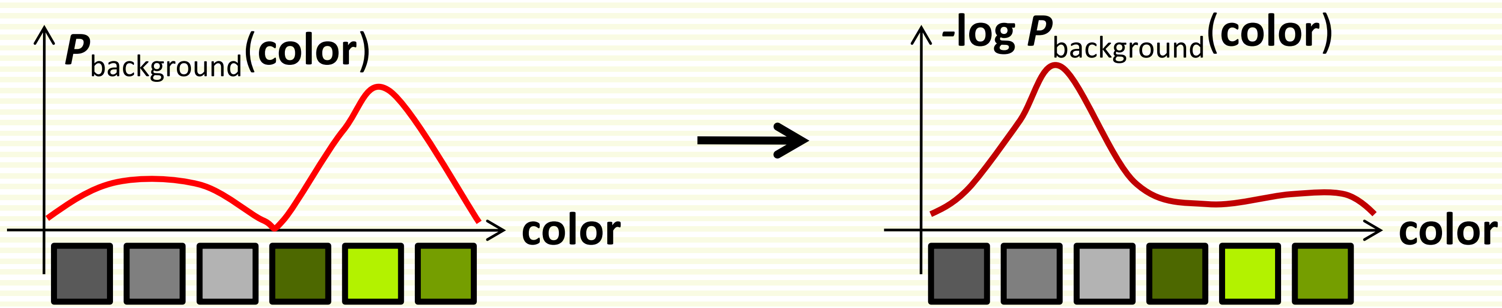


- Normalized histogram for distribution

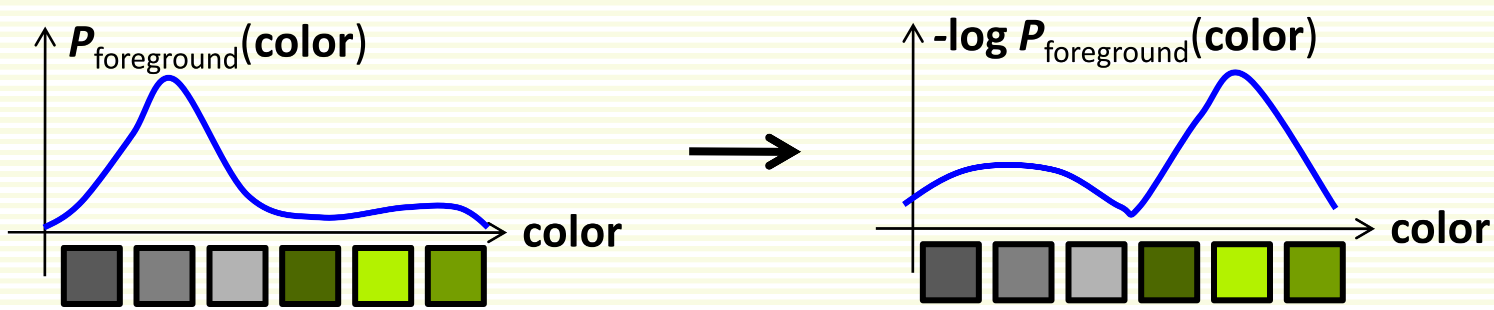
$$\boldsymbol{P}_{\text{foreground}}(\textbf{color}) = \frac{\text{number of foreground seeds of color}}{\text{total number of foreground seeds}}$$

FG Segmentation: Color Distributions

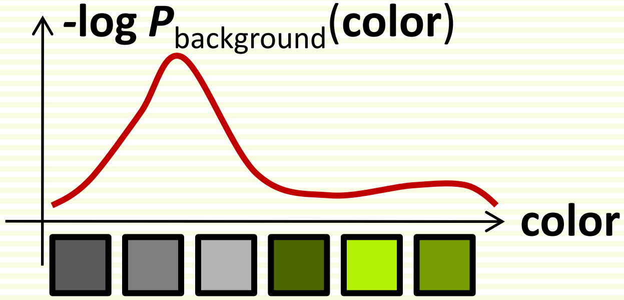
- For green pixels p, $P_{background}(p)$ is high, $P_{background}(p)$ low
- We want just the opposite for the data term
- Convert to "opposite" using **–log()**

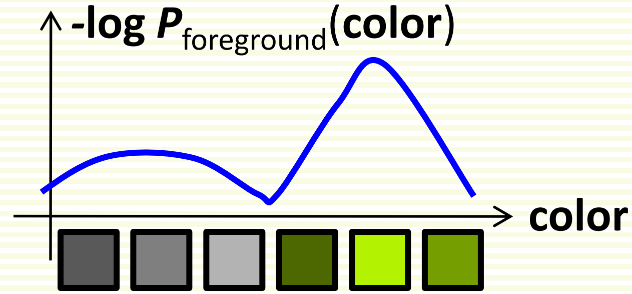


- Do the same for the foreground



FG Segmentation: Color Distributions

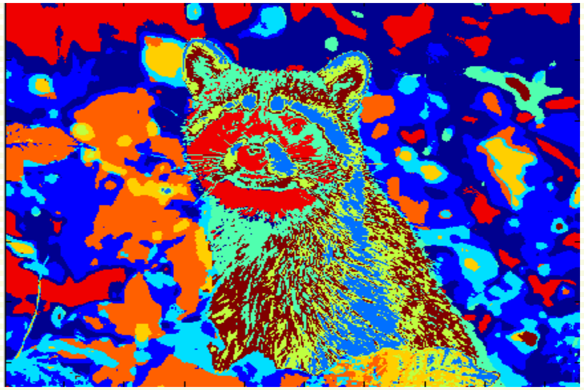




- D_p(foreground) = - log $P_{foreground}$(color of ***p***)
- D_p(background) = - log $P_{background}$(color of ***p***)
- Problem:
- The number of colors is too high: 256^3
  - too large to build a normalized histogram
- Cluster colors using kmeans clustering, and treat each cluster as the “new” color

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

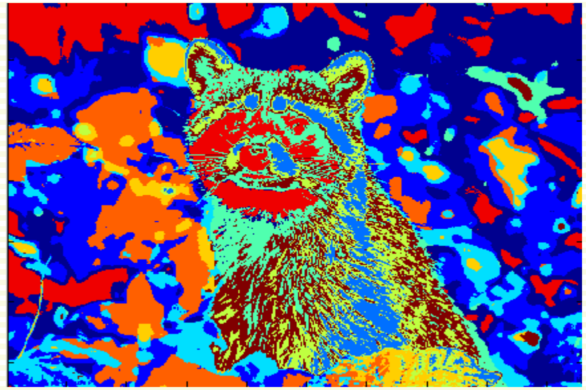


clusters visualized with random colors

pixels painted with average color of pixels in its cluster

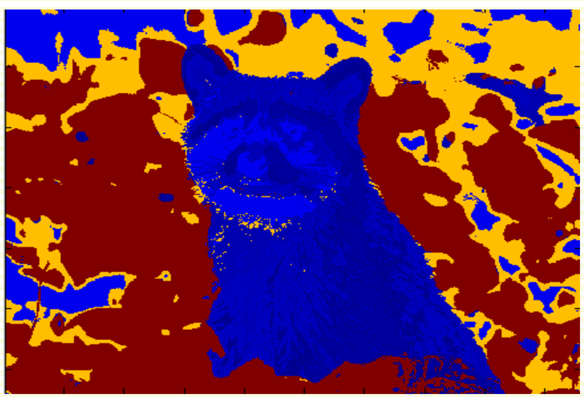
FG Segmentation: Segmentation Result

user input

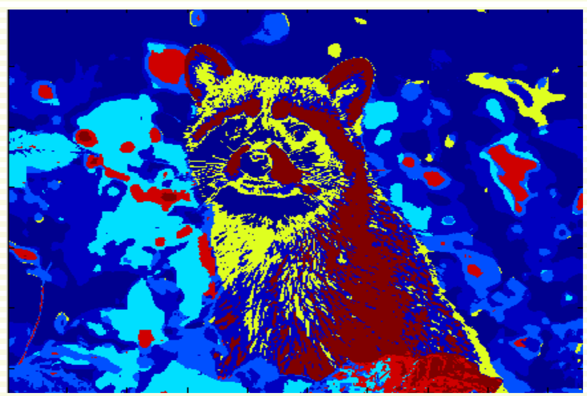


reduced colors

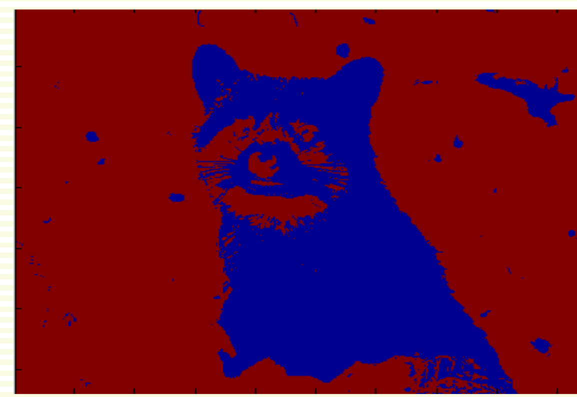
segmentation



foreground *D*



background *D*



blue pixels prefer foreground
red pixels prefer background