Segmentation and graph-based techniques

16-385 Computer Vision
Spring 2020, Lecture 27

http://www.cs.cmu.edu/~16385/

Course announcements

- Programming assignment 6 is due tonight.
- Programming assignment 7 and take-home quiz 11 will be posted tonight.
- Anand will cover Friday’s OH, and I will cover Monday’s.

Course overview

1. Image processing. ← Lectures 1 – 7
   See also 18-793: Image and Video Processing

2. Geometry-based vision. ← Lectures 7 – 12
   See also 16-822: Geometry-based Methods in Vision

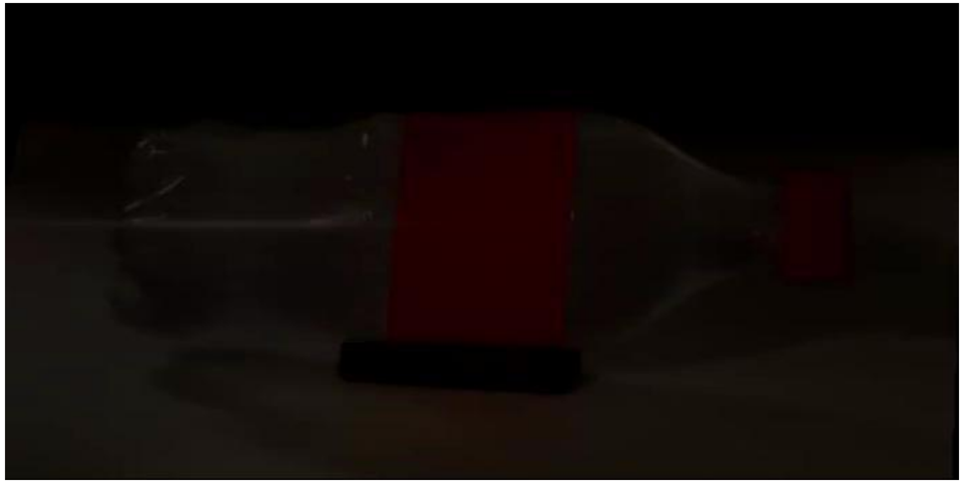
3. Physics-based vision. ← Lectures 13 – 16
   See also 16-823: Physics-based Methods in Vision
   See also 15-462: Computer Graphics
   See also 15-463: Computational Photography

4. Semantic vision. ← Lectures 17 – 20
   See also 16-824: Vision Learning and Recognition
   See also 10-703: Deep Reinforcement Learning

5. Dealing with motion. ← Lectures 21 – 24
   See also 16-831: Statistical Techniques in Robotics
   See also 16-833: Robot Localization and Mapping

15-463/15-663/15-862 Computational Photography

Learn about this and other unconventional cameras – and build some on your own!



cameras that take video at the speed of light

cameras that measure depth in real time

cameras that see around corners

cameras that capture entire focal stacks

http://graphics.cs.cmu.edu/courses/15-463/


ICCP 2020
Saint Louis
April 24-26, 2020
• Free registration!
• All talks and discussion sessions online
• Details at https://iccp2020.engr.wustl.edu/
The field of Computational Photography seeks to create new photographic functionalities and experiences that go beyond what is possible with traditional cameras and image processing tools. The IEEE International Conference on Computational Photography is organized with the vision of fostering the community of researchers, from many different disciplines, working on computational photography.

Overview of today's lecture

- Leftover from tracking.
- Segmentation.
- Image as a graph.
- Shortest graph paths and Intelligent scissors.
- Graph-cuts and GrabCut.
- Normalized cuts.
- Boundaries.
- Clustering for segmentation.

Slide credits

Most of these slides were adapted from:

- Kris Kitani (15-463, Fall 2016).

Some slides were inspired or taken from:

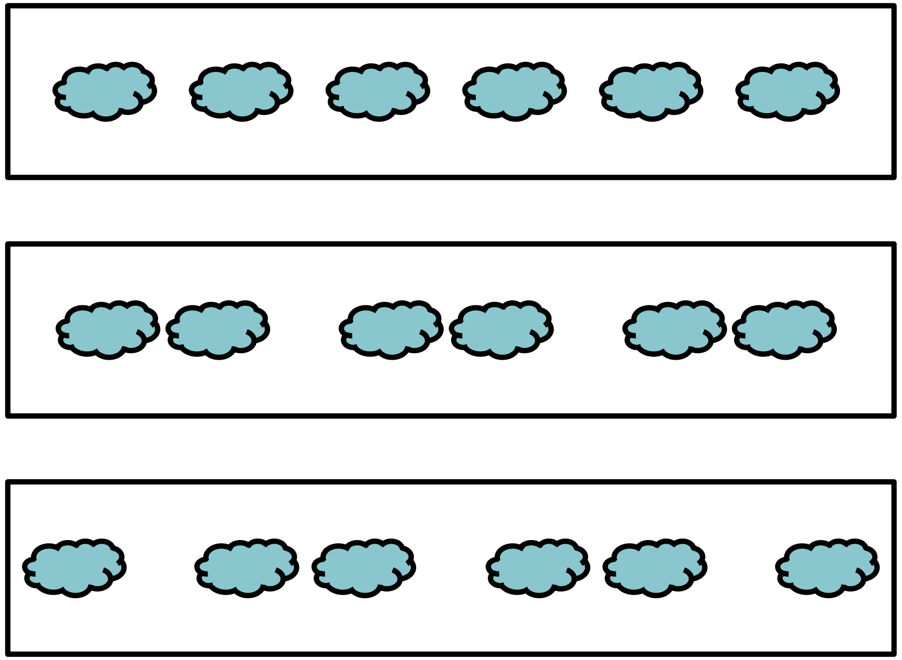
- Fredo Durand (MIT).
- James Hays (Georgia Tech).

Segmentation

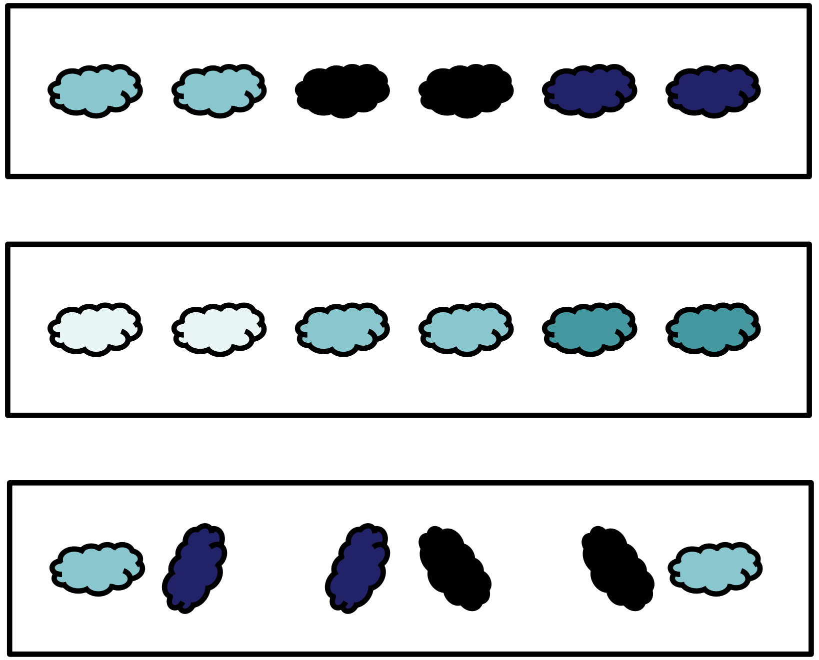
Gestalt Psychology

We perceive objects in their entirety before their individual parts.

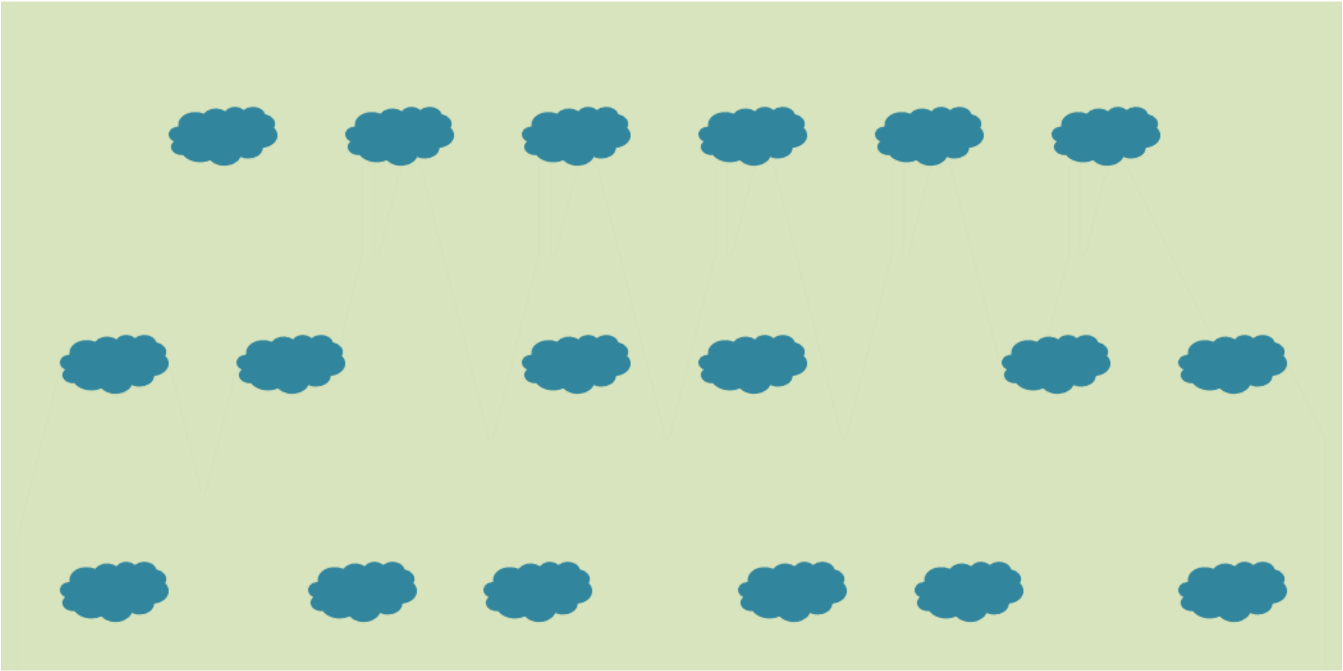
Closer objects are grouped together



Similar objects are grouped together

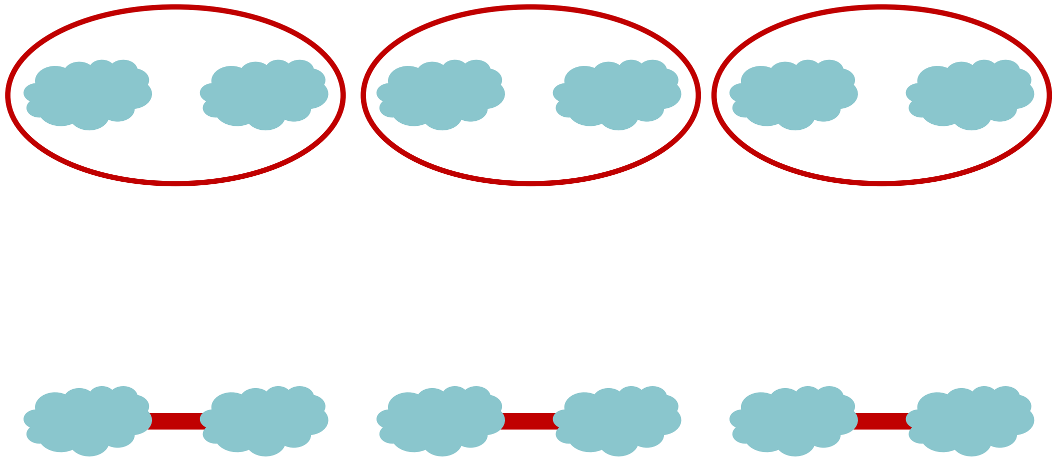


Common Fate



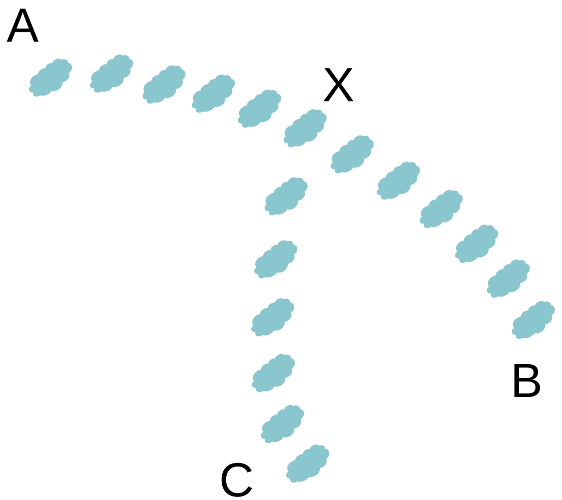
Objects with similar motion or change in appearance are grouped together

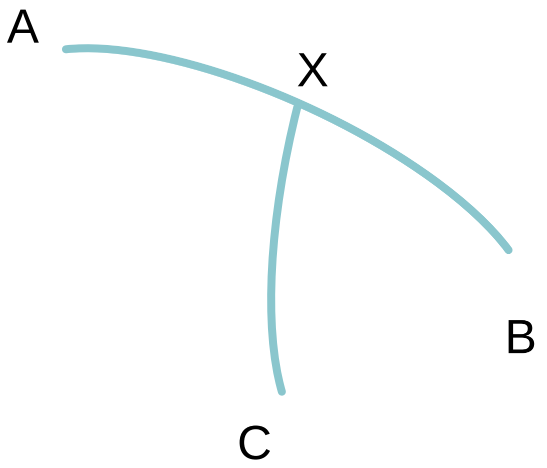
Common Region/Connectivity



Connected objects are grouped together

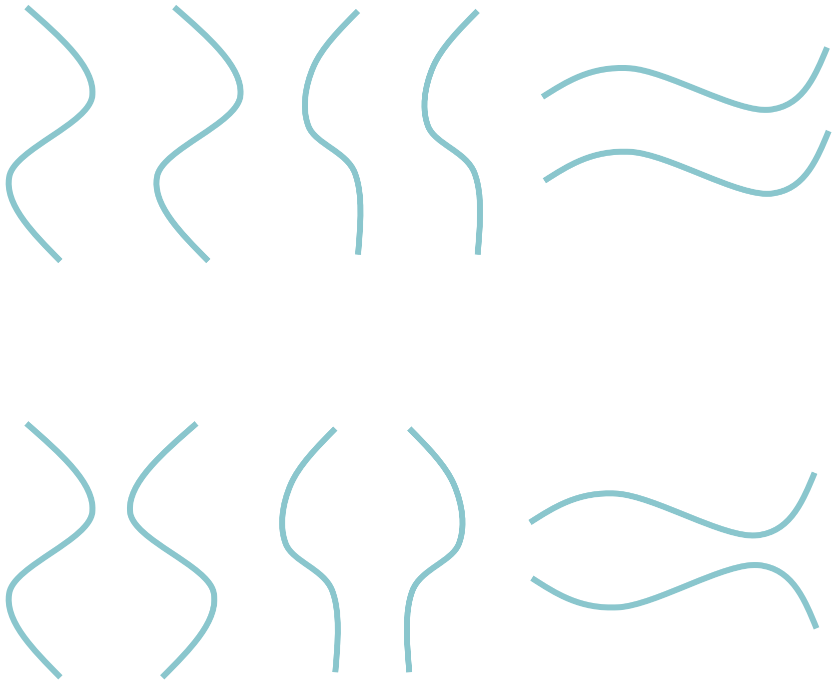
Continuity Principle



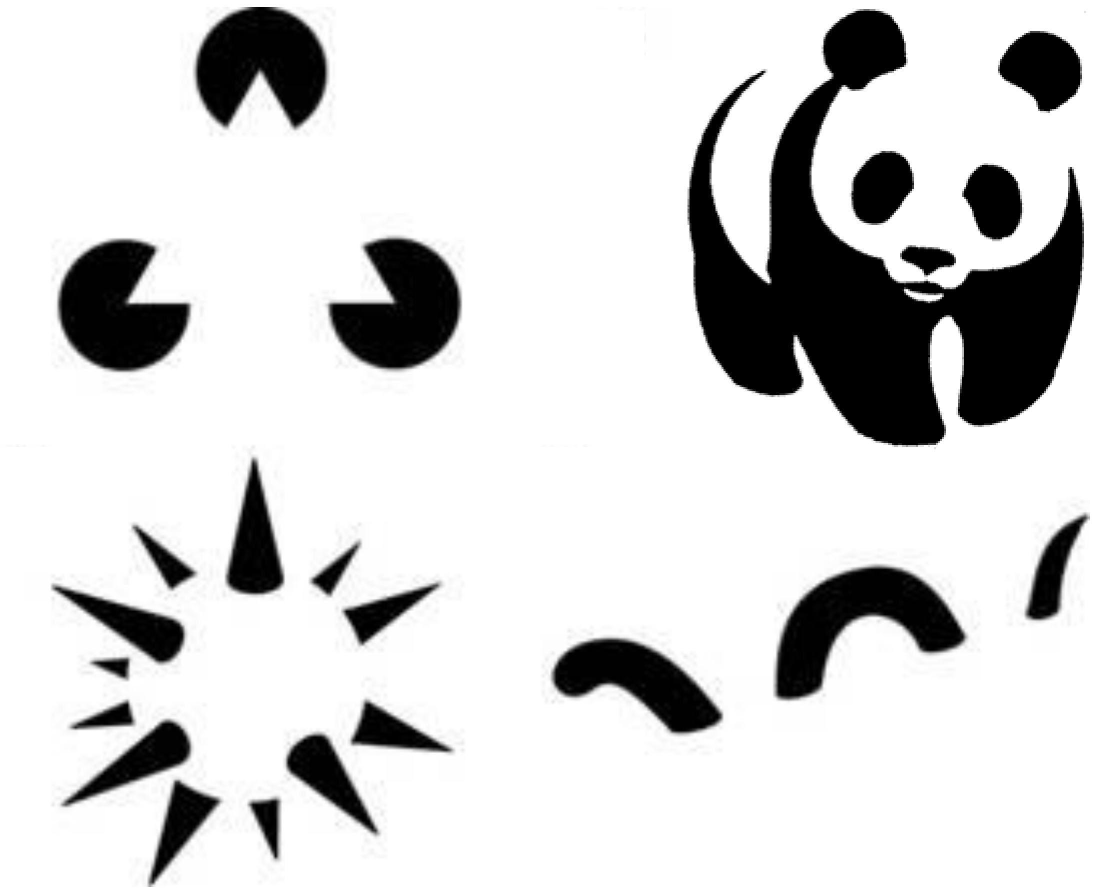


Features on a continuous curve are grouped together

Symmetry Principle



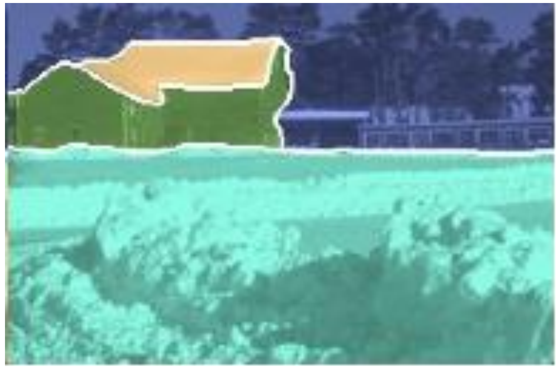
Completion



Illusory or subjective contours are perceived

Segmentation/Clustering

$k = 4$

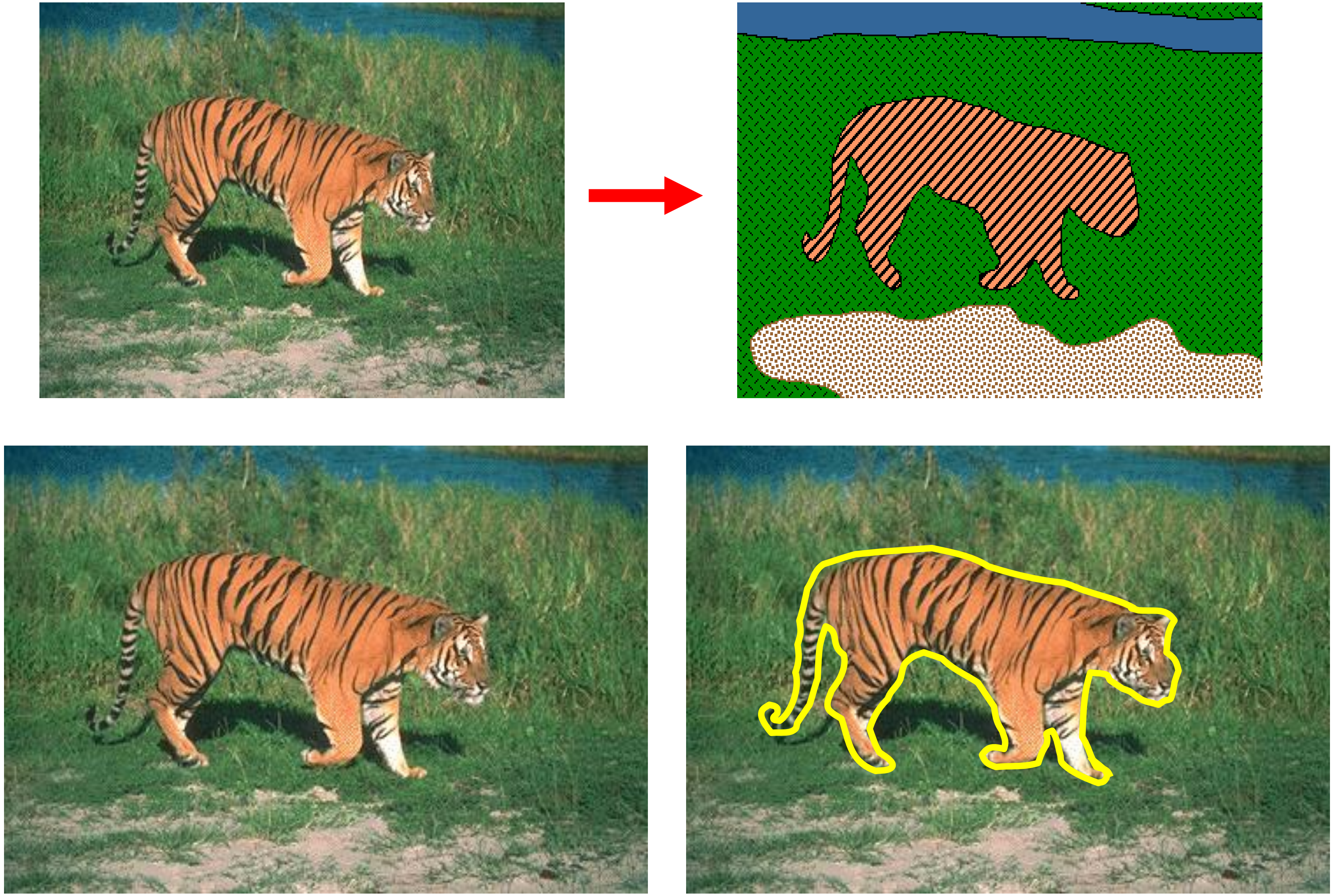


$nc = .0017$

$k = 5$

$nc = .0060$

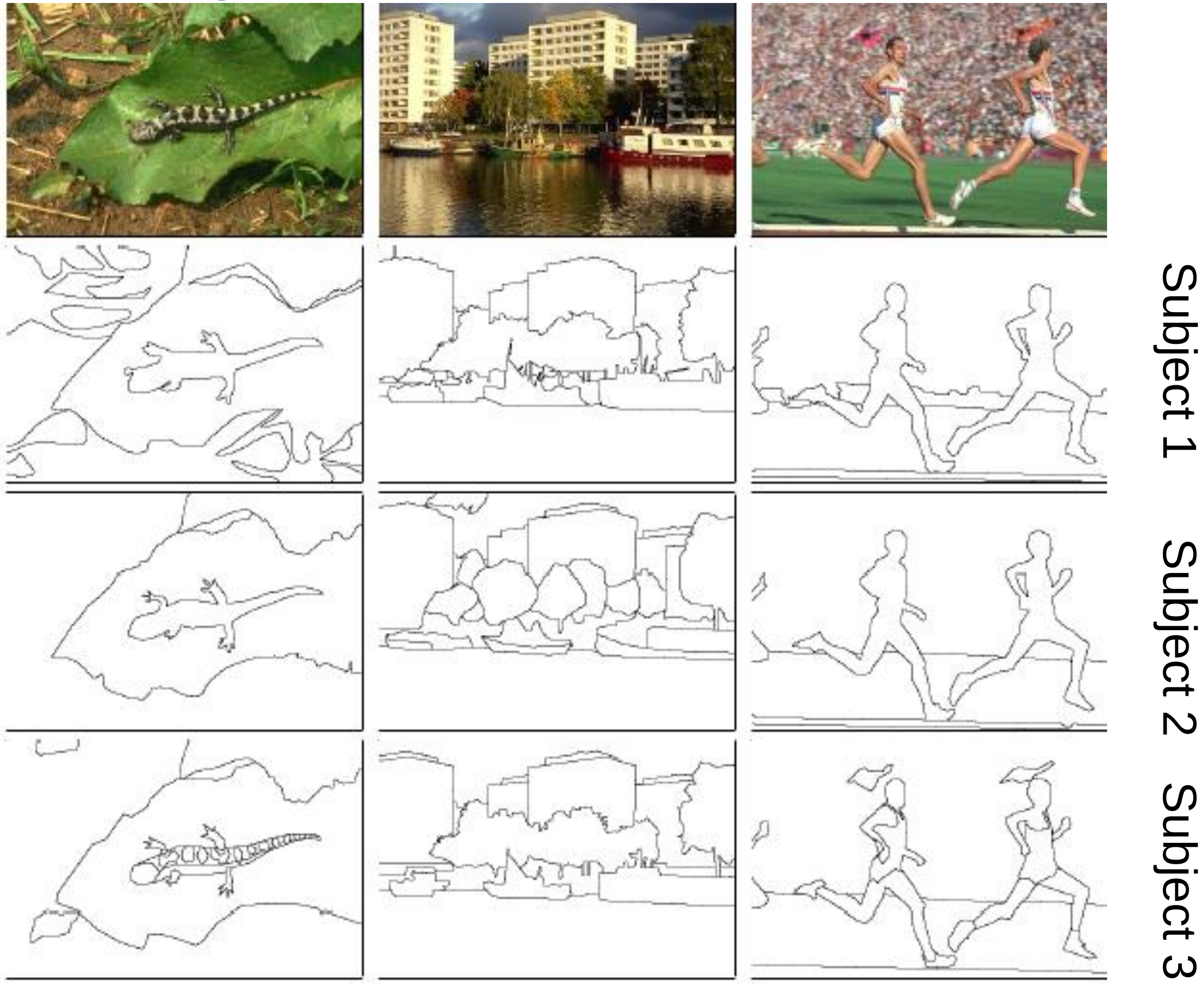
$k = 11$



What is a “good” segmentation??

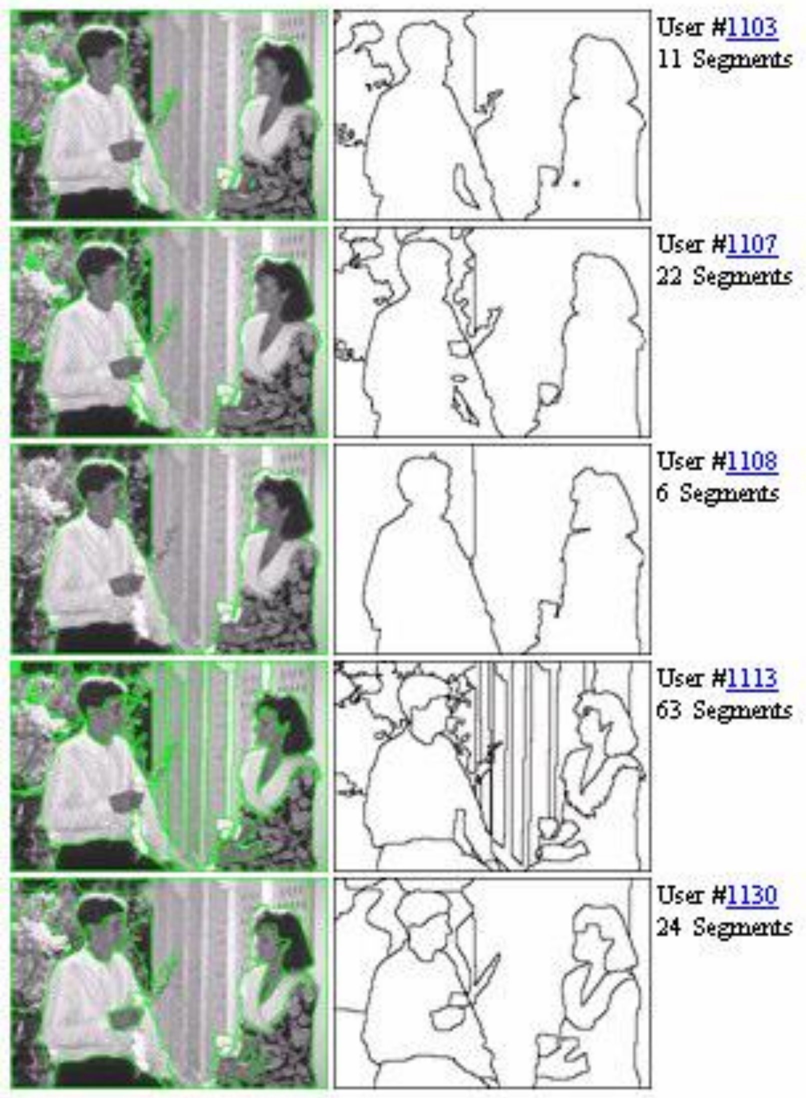
First idea: Compare to human segmentation or to "ground truth"

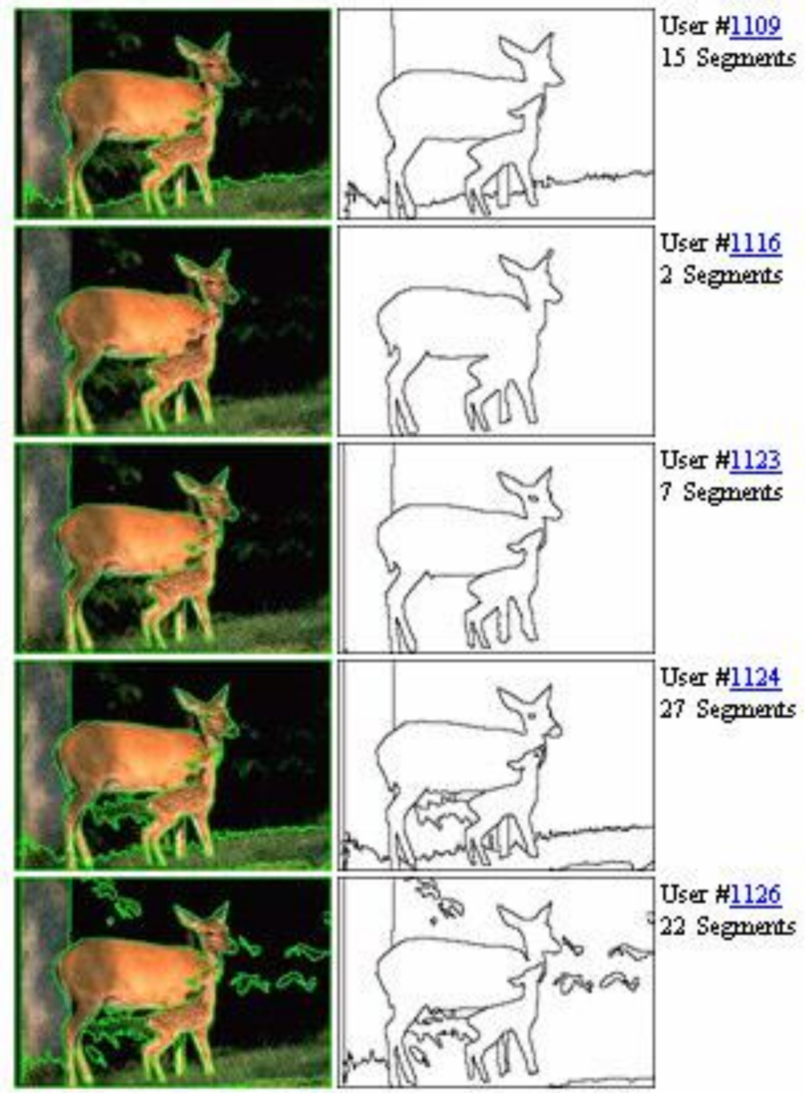
No objective definition of segmentation!



- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

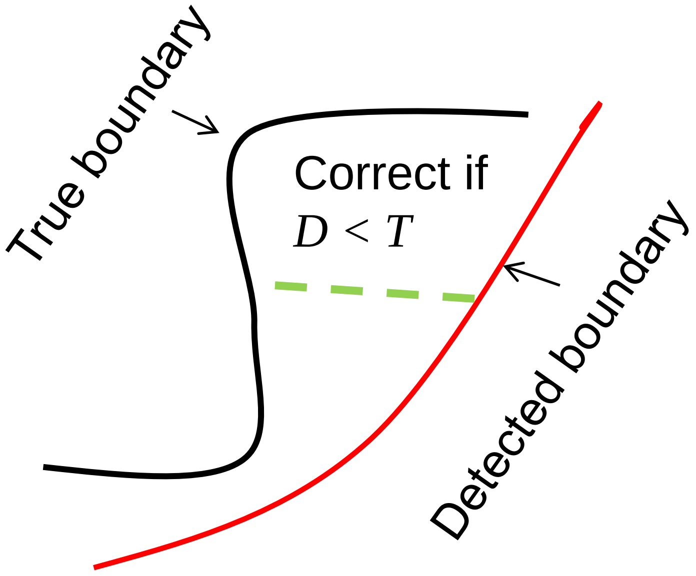
No objective definition of segmentation!

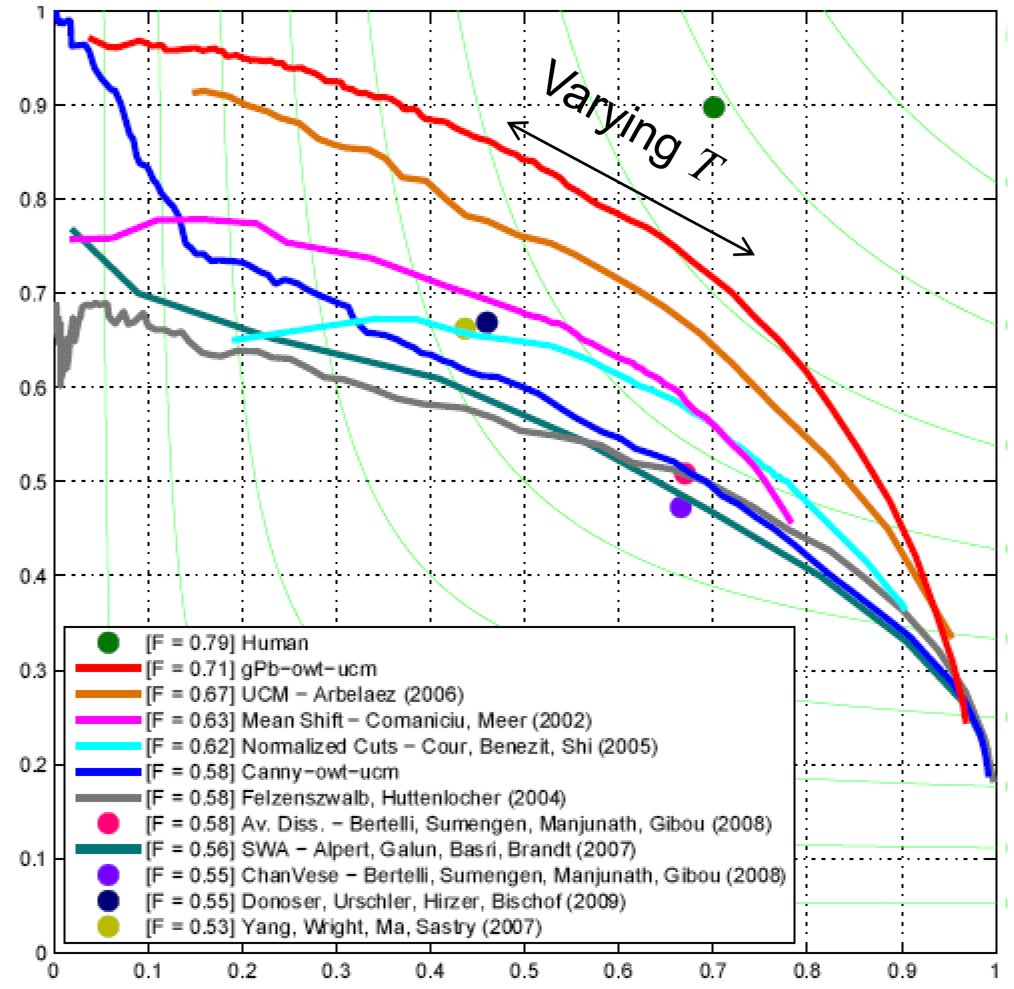




- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/color/317080.html

Evaluation: Boundary agreement

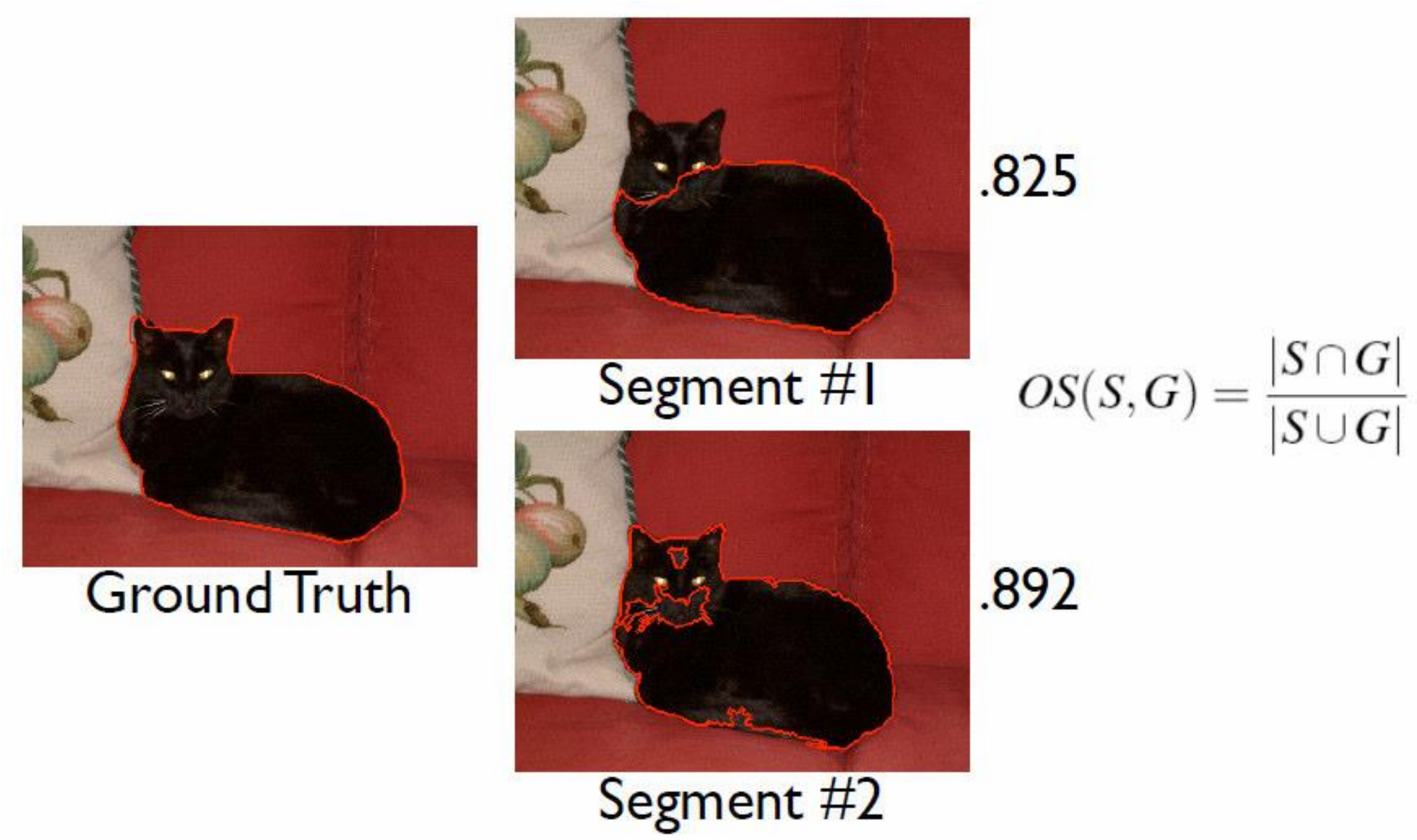




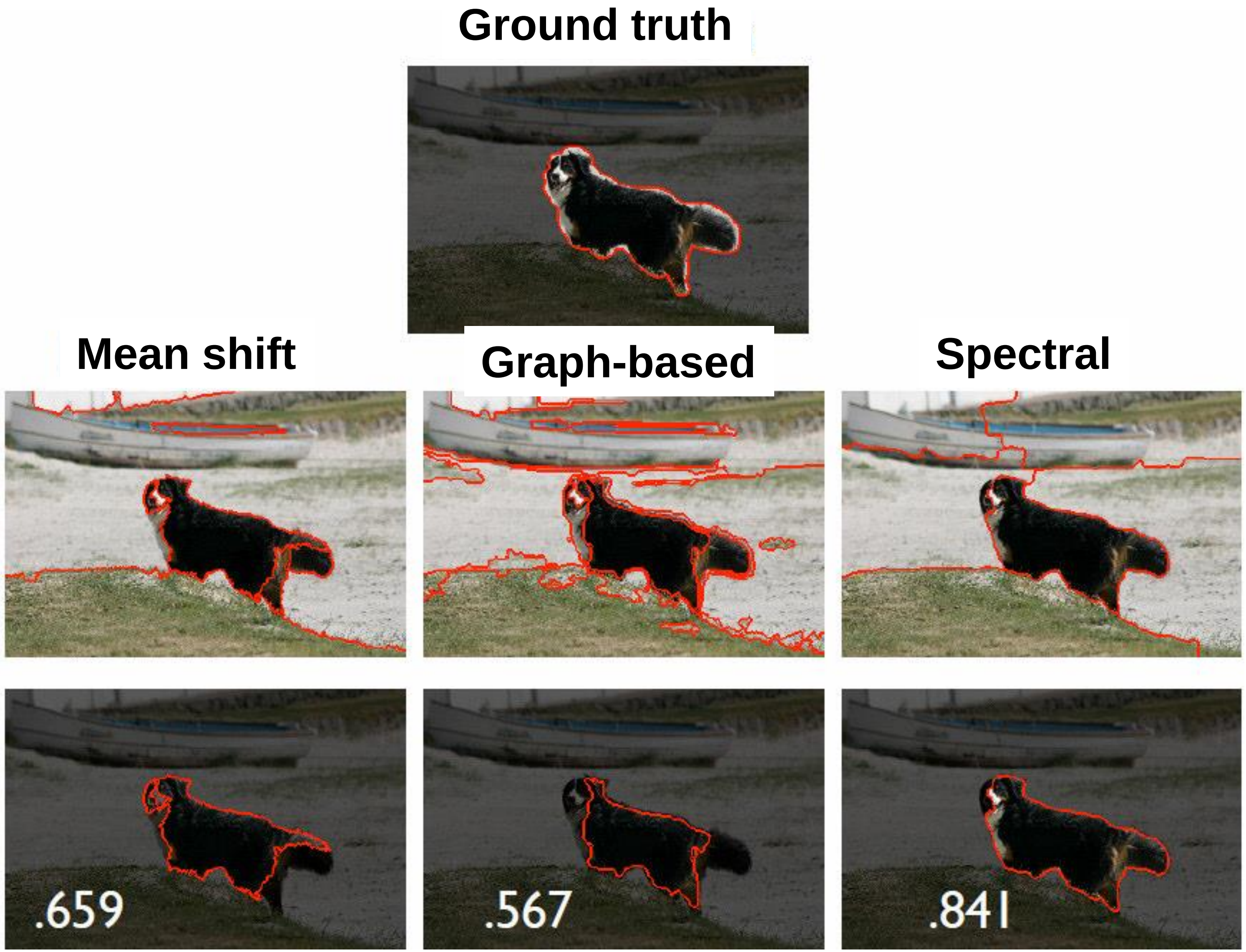
Precision = % of detected boundary pixels that are correct

Recall = % of boundary pixels that are detected

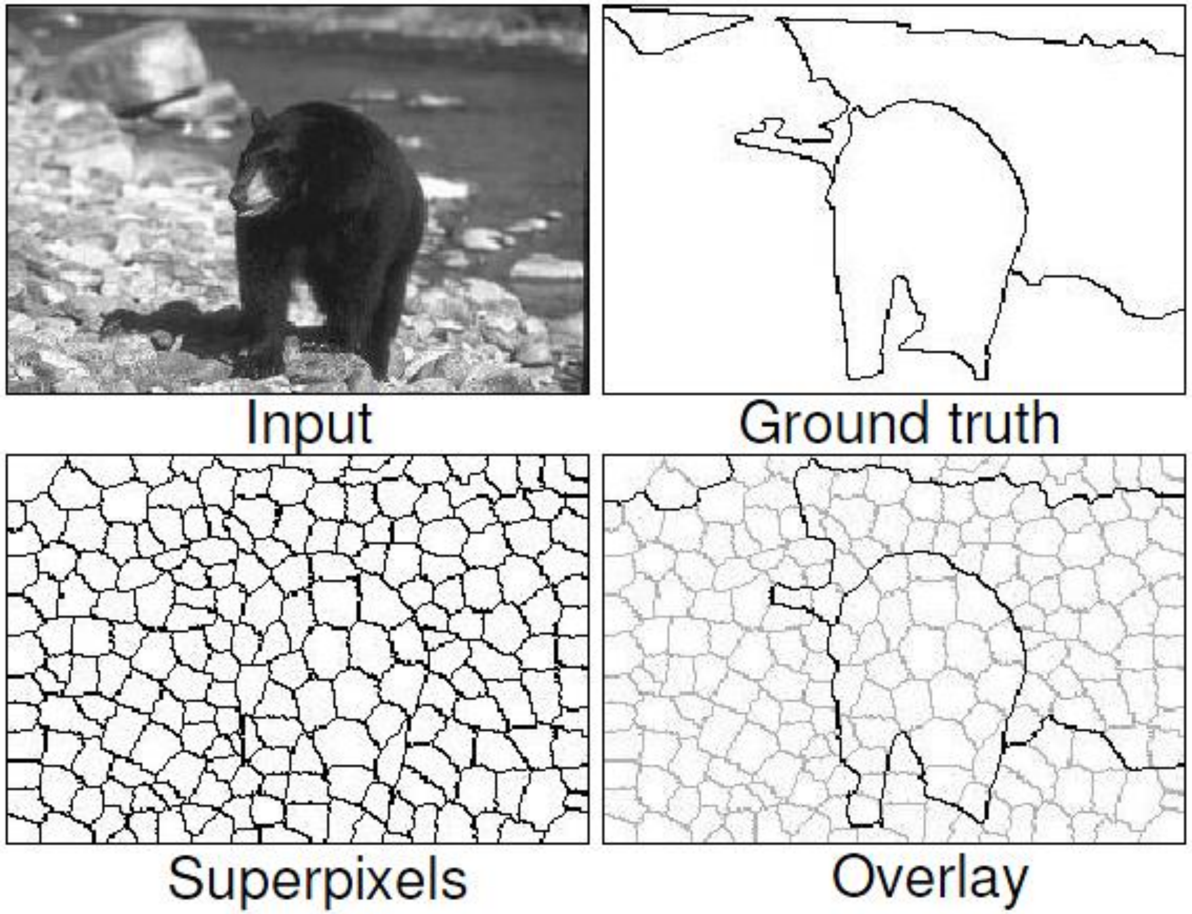
Evaluation: Region overlap with ground truth

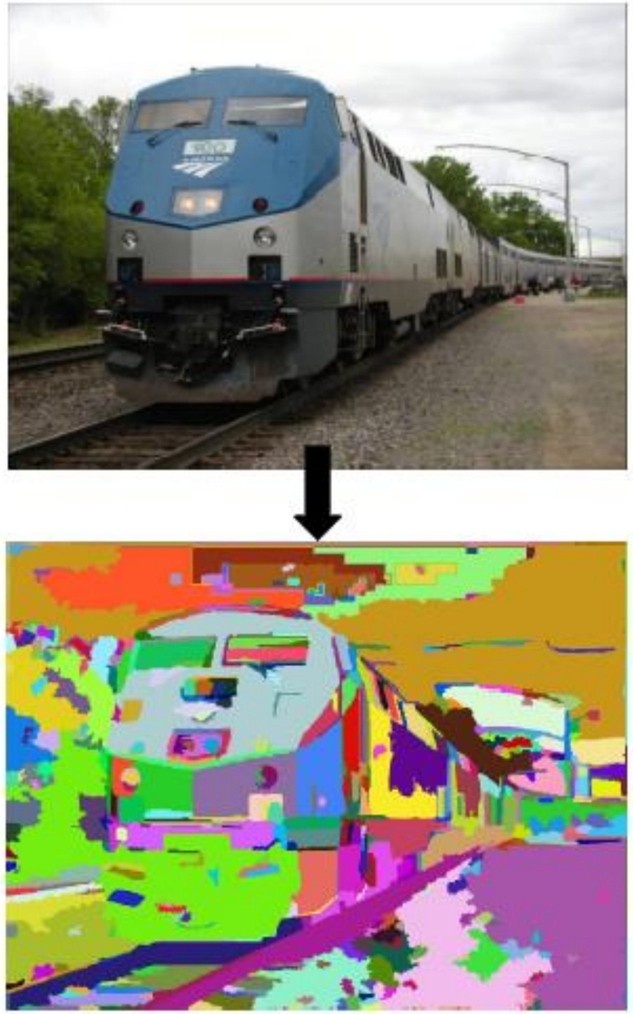


Evaluation: Region overlap with ground truth



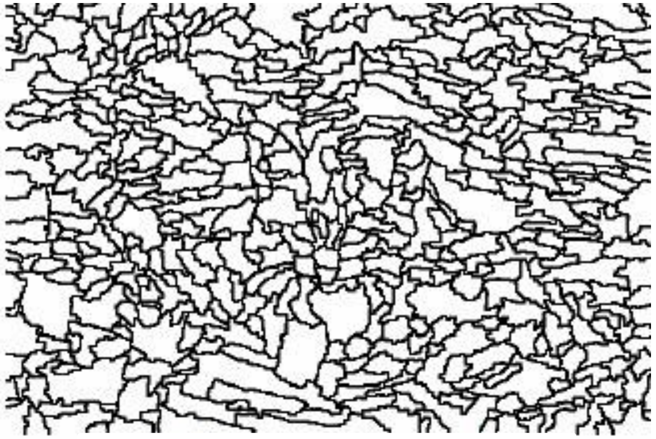
Second idea: Superpixels



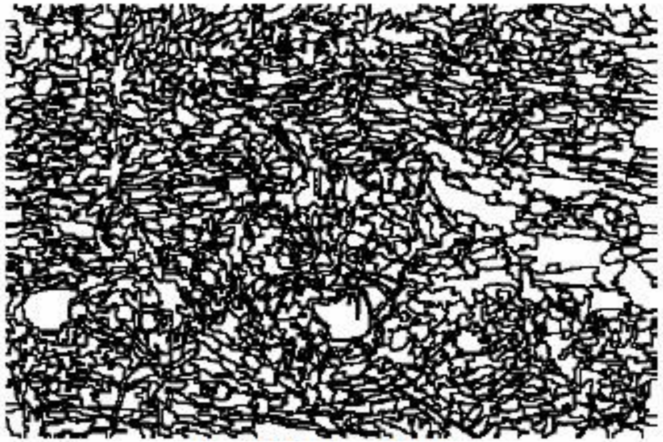


- Let’s not even try to compute a “correct” segmentation
- Let’s be content with an *oversegmentation* in which each region is very likely (formal guarantees are hard) to be uniform

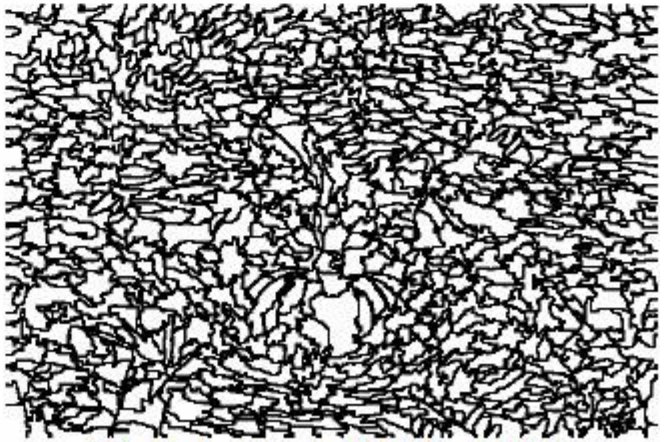
Second idea: Superpixels



Watershed



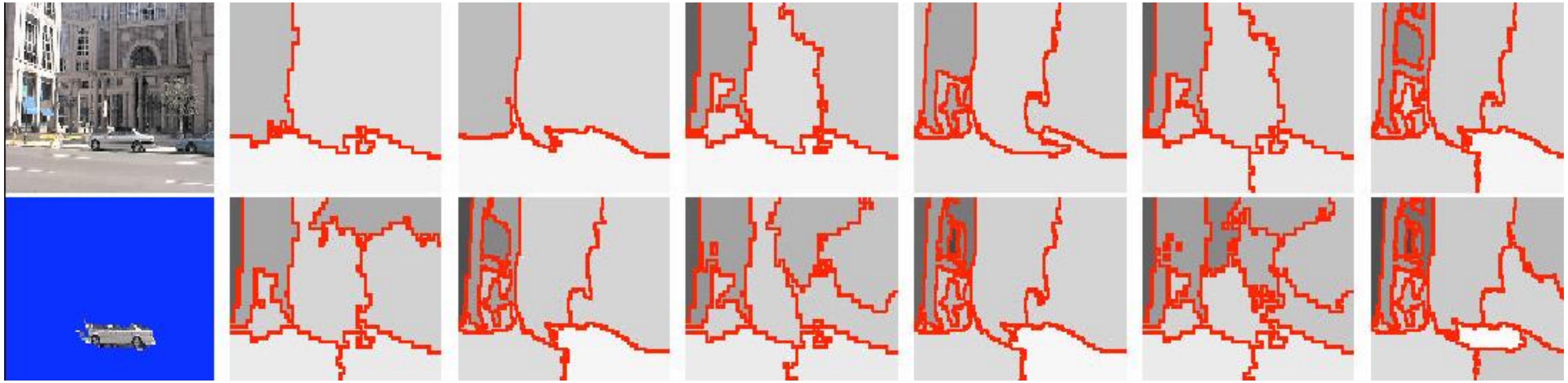
Mean shift



Graph-based

- Example from: How Do Superpixels Affect Image Segmentation?
- Progress in Pattern Recognition, Image Analysis and Applications. Springer LNCS. Volume 5197/2008.

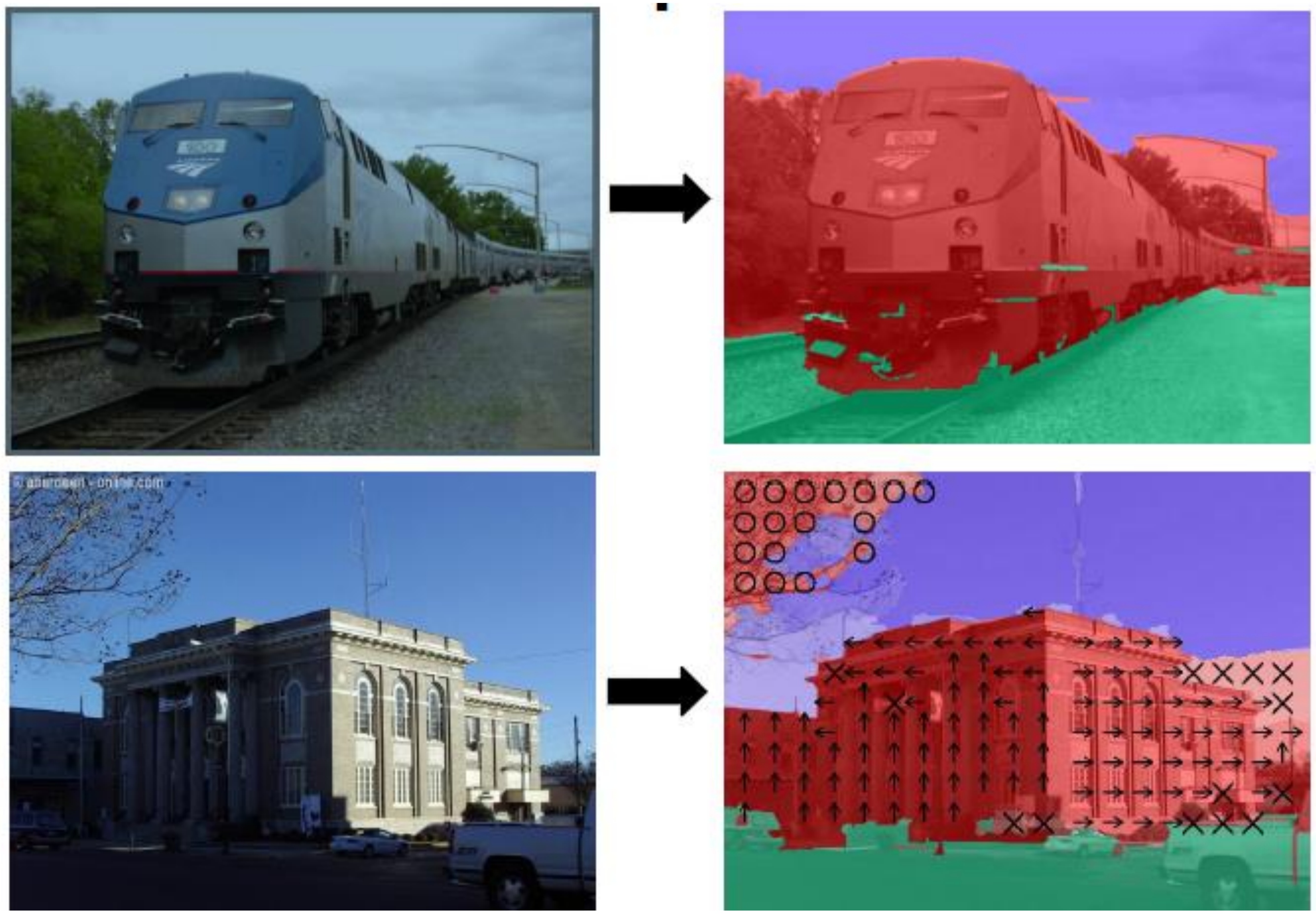
Third idea:
Multiple segmentations



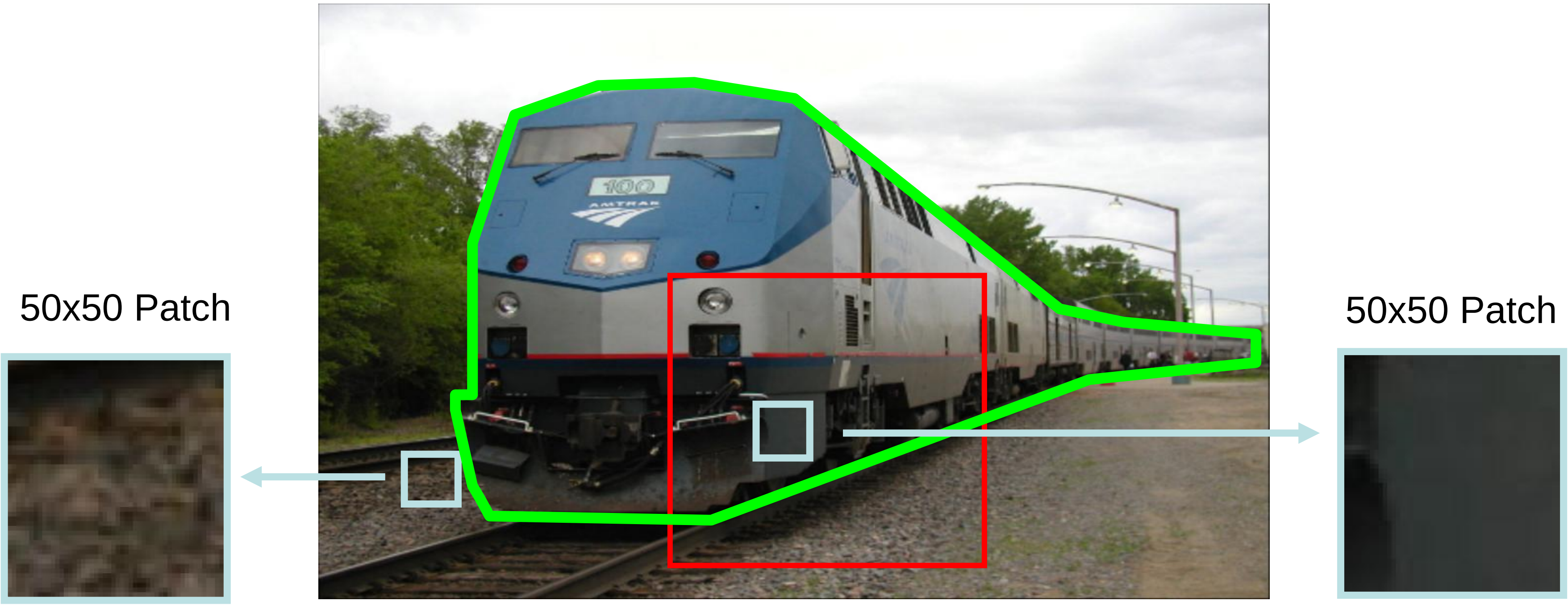
- Generate many segmentations of the same image
- Even though many regions are “wrong”, some consensus should emerge

Example: Improving Spatial Support for Objects via Multiple Segmentations
Tomasz Malisiewicz and Alexei A. Efros. British Machine Vision Conference (BMVC), September, 2007.

Multiple segmentations: Example



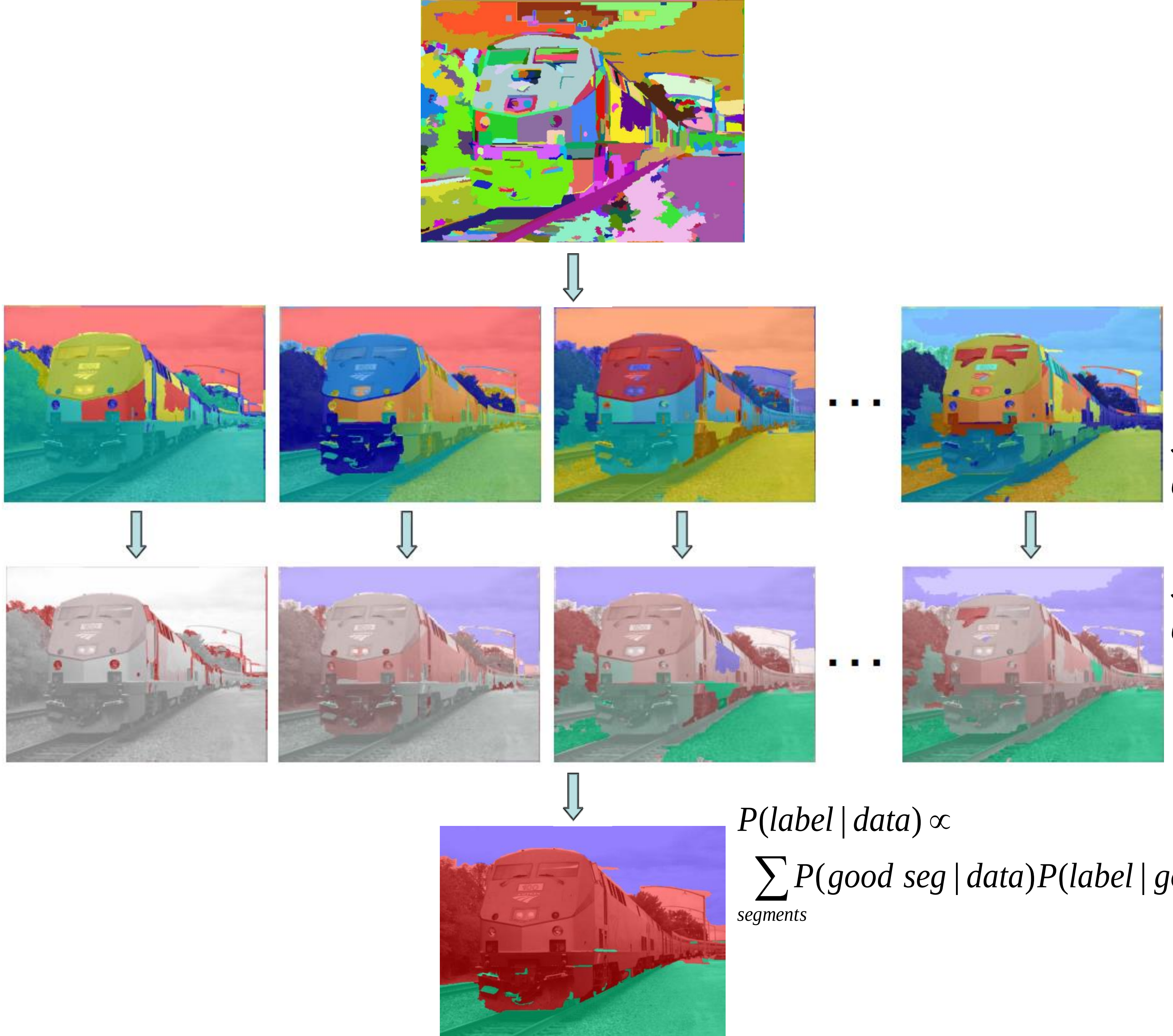
- Task: Regions → Features → Labels (horizontal, vertical, sky, etc.)



- Chicken and egg problem:
  - If we knew the regions, we could compute the features and label the right regions
  - But to know the right regions we need to know the labels!
- Solution:
  - Generate lots of segmentations
  - Combine the classifications to get consensus

Example from D. Hoiem

Recovering Surface Layout from an Image. D. Hoiem, A.A. Efros, and M. Hebert. IJCV, Vol. 75, No. 1, October 2007.



P(*good segment* | *data*) x

P(*label* | *good segment*, *data*)

$$P(label \mid data) \propto \sum_{segments} P(good\ seg \mid data) P(label \mid good\ seg, data)$$

Generalities: Summary

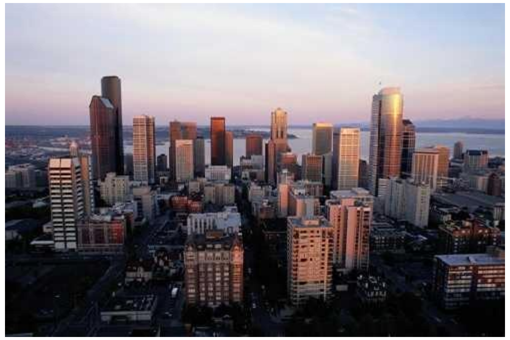
- Match ground truth (no objective definition)
- Superpixels = oversegmentation
- Using multiple segmentations

Main approaches

- Spectral techniques
- Segmentation as boundary detection
- Graph-based techniques
- Clustering (K-means and probabilistic)
- Mean shift

Cut and paste procedure

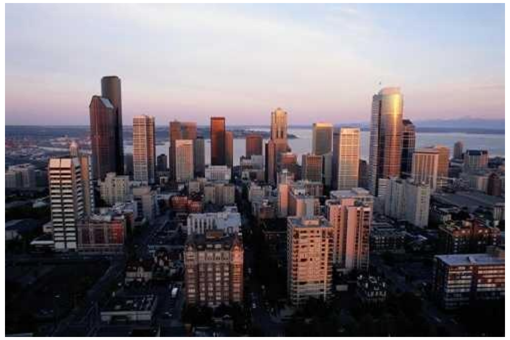
1. Extract Sprites



2. Blend them into the composite

Cut and paste procedure

1. Extract Sprites

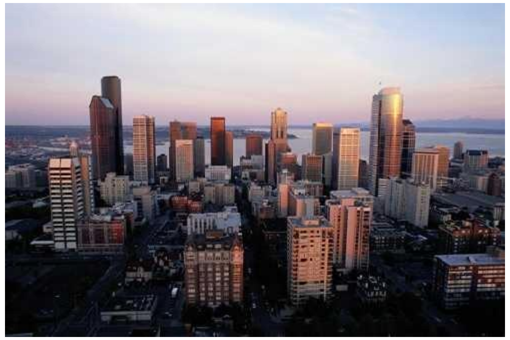


2. Blend them into the composite

How do we do this?

Cut and paste procedure

1. Extract Sprites



How do we do this?

Two different ways to think about the same thing:

- Finding seams (i.e., finding the pixels where to cut an image)

- Segmentation (i.e., splitting the image into “foreground” and “background”)

I will be using the two terms interchangeable

Applications

Finding seams is also useful for:

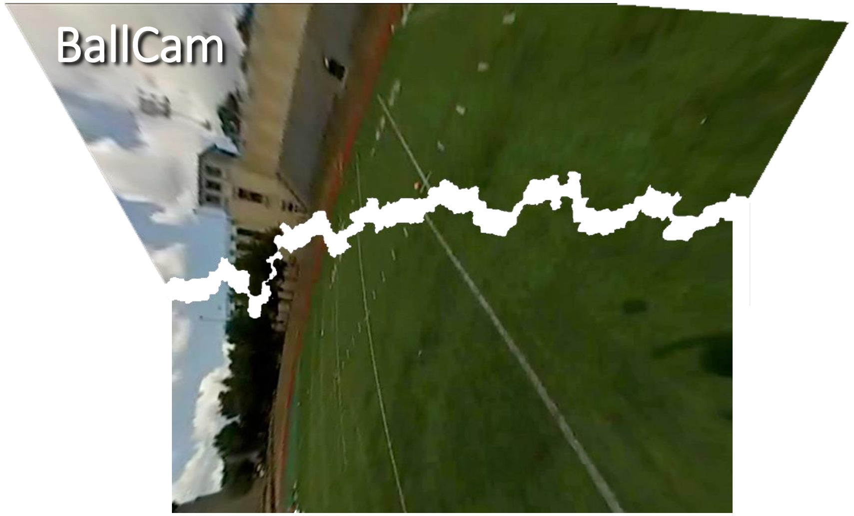


image stitching

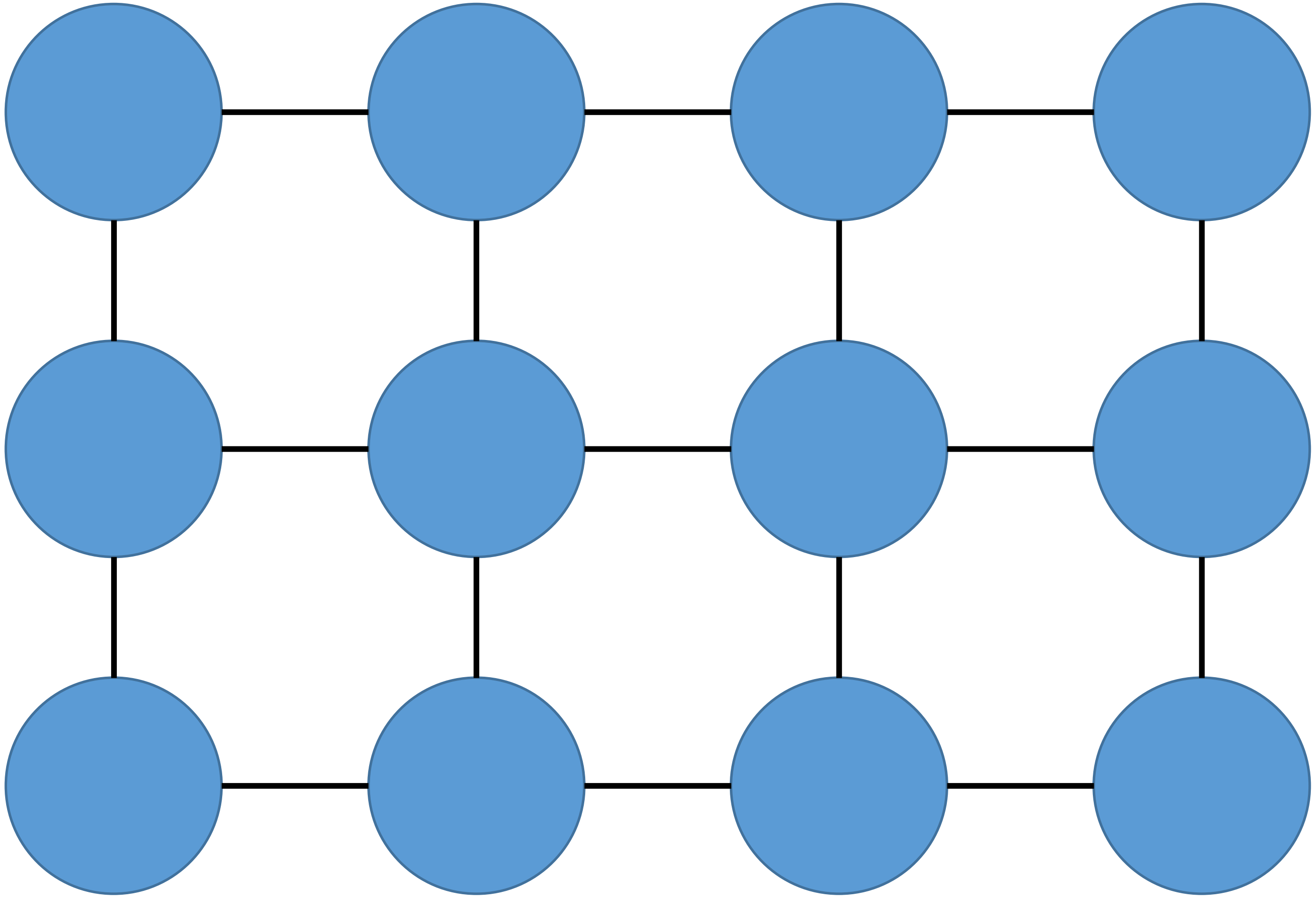
retargeting

segmentation

Image as a graph

Fundamental theme of today's lecture

Images can be viewed as graphs

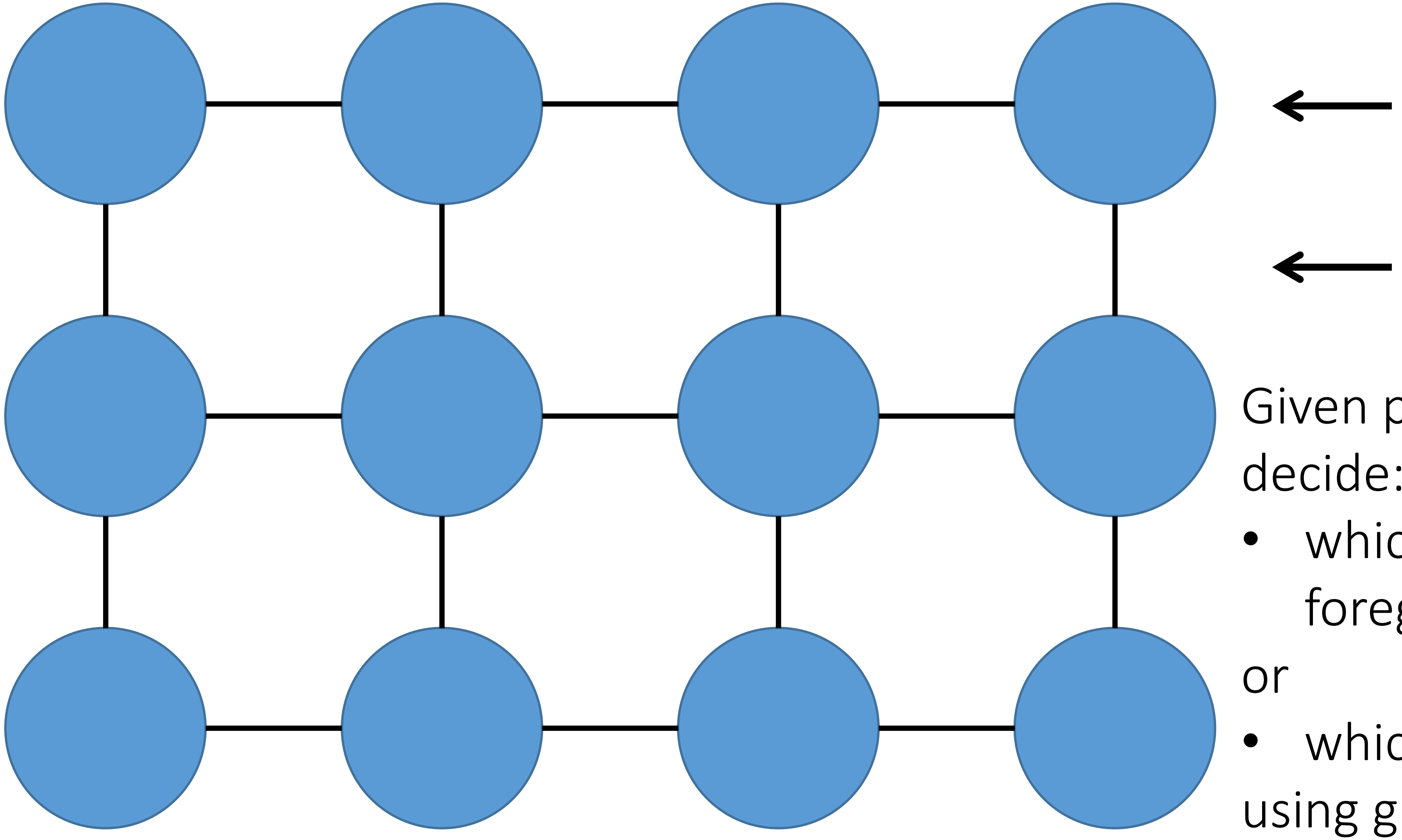


Nodes: pixels

Edges: Constraints between neighboring pixels

Graph-view of segmentation problem

Segmentation is node-labeling



Nodes: pixels

Edges: Constraints between neighboring pixels

Given pixel values and neighborhoods, decide:

- which nodes to label as foreground/background

or

- which nodes to label as seams

using graph algorithms

Graph-view of segmentation problem

Today we will cover:

Method	Labeling problem	Algorithm	Intuition
Intelligent scissors	label pixels as seams	Dijkstra’s shortest path (dynamic programming)	short path is a good **boundary**
GrabCut	label pixels as foreground/background	max-flow/min-cut (graph cutting)	good **region** has low cutting cost

Shortest graph paths and intelligent scissors

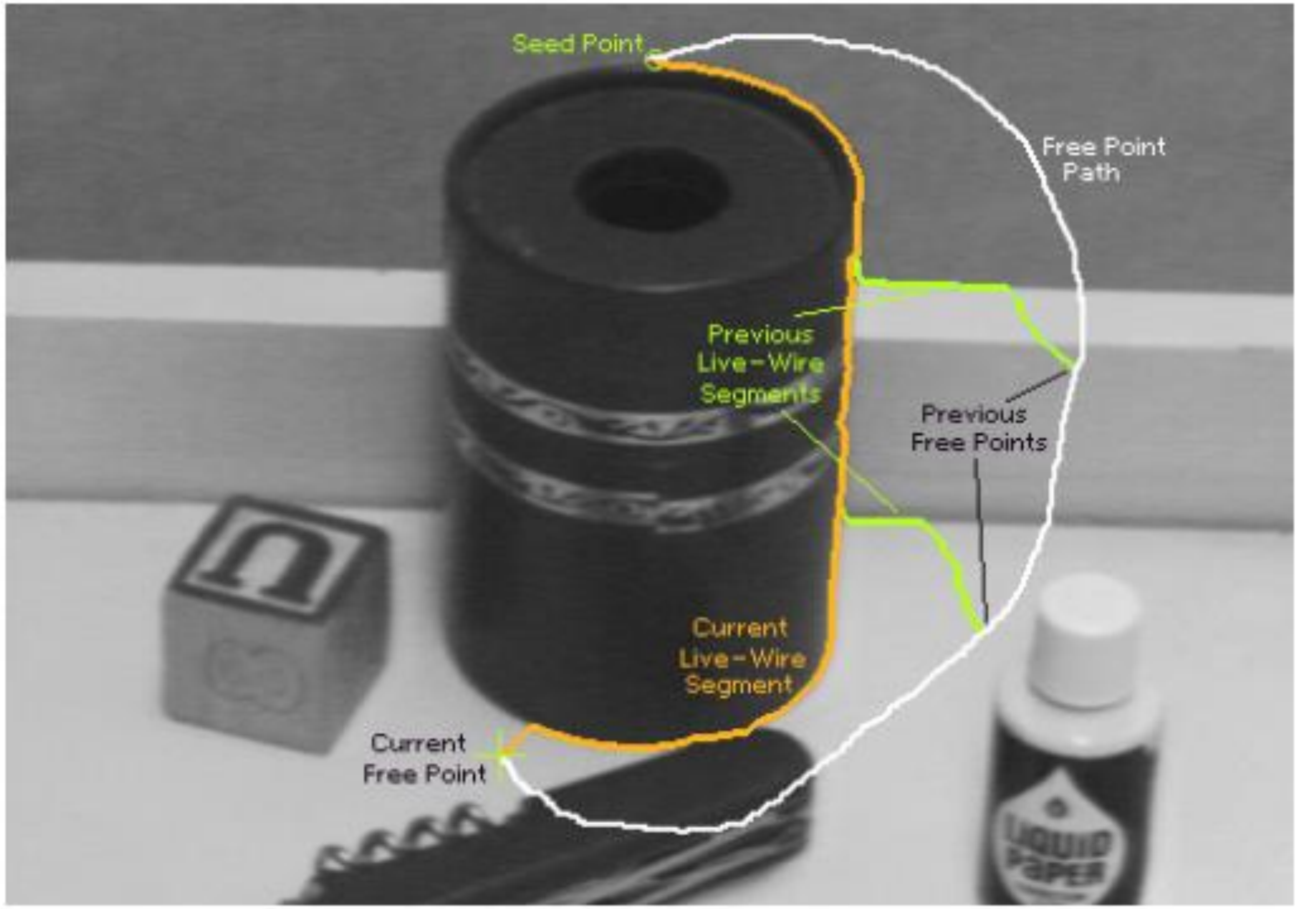
Intelligent scissors

Problem statement:

Given <u>two seed points</u>, find a good boundary connecting them

Challenges:

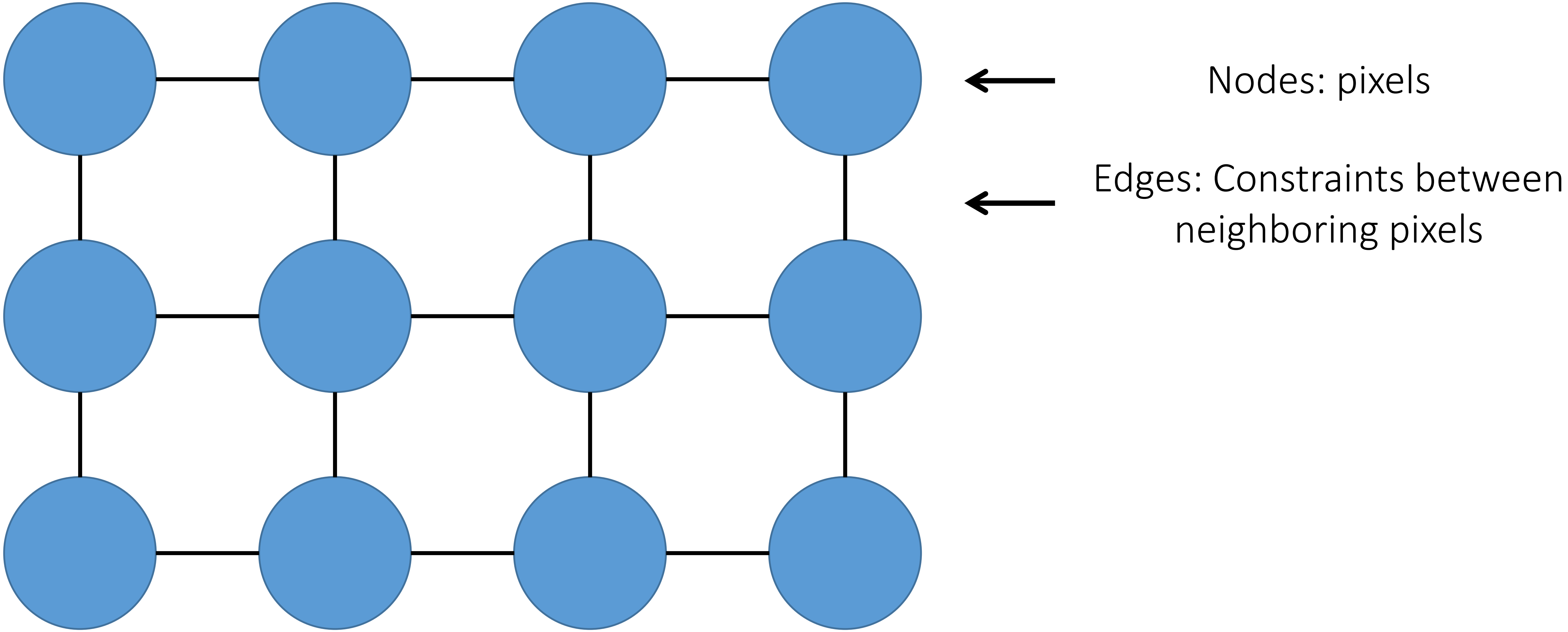
- Make this real-time for interaction
- Define what makes a good boundary



Mortenson and Barrett (SIGGRAPH 1995)
(you can tell it's old from the paper's low quality teaser figure)

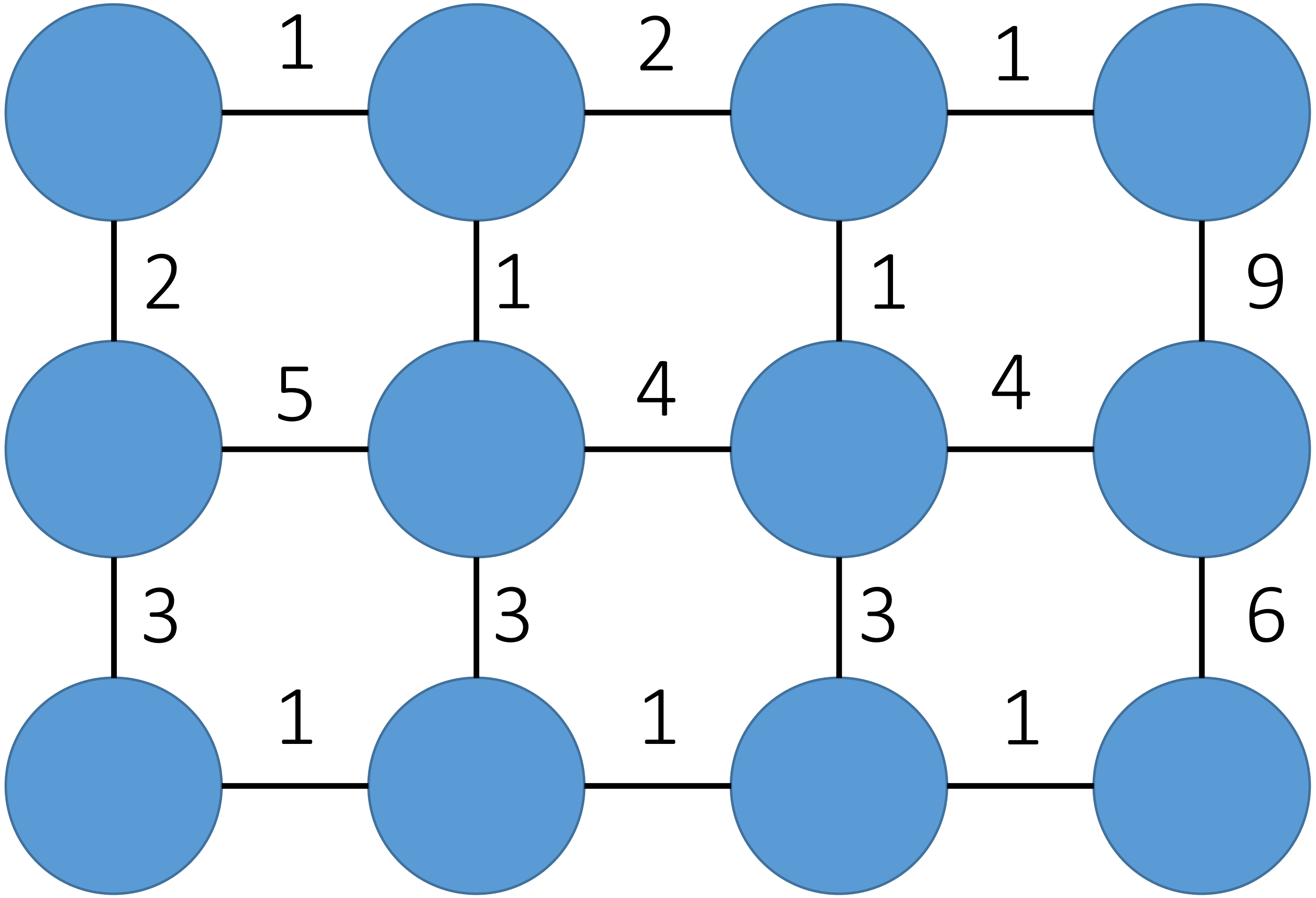
Graph-view of this problem

Images can be viewed as graphs



Graph-view of this problem

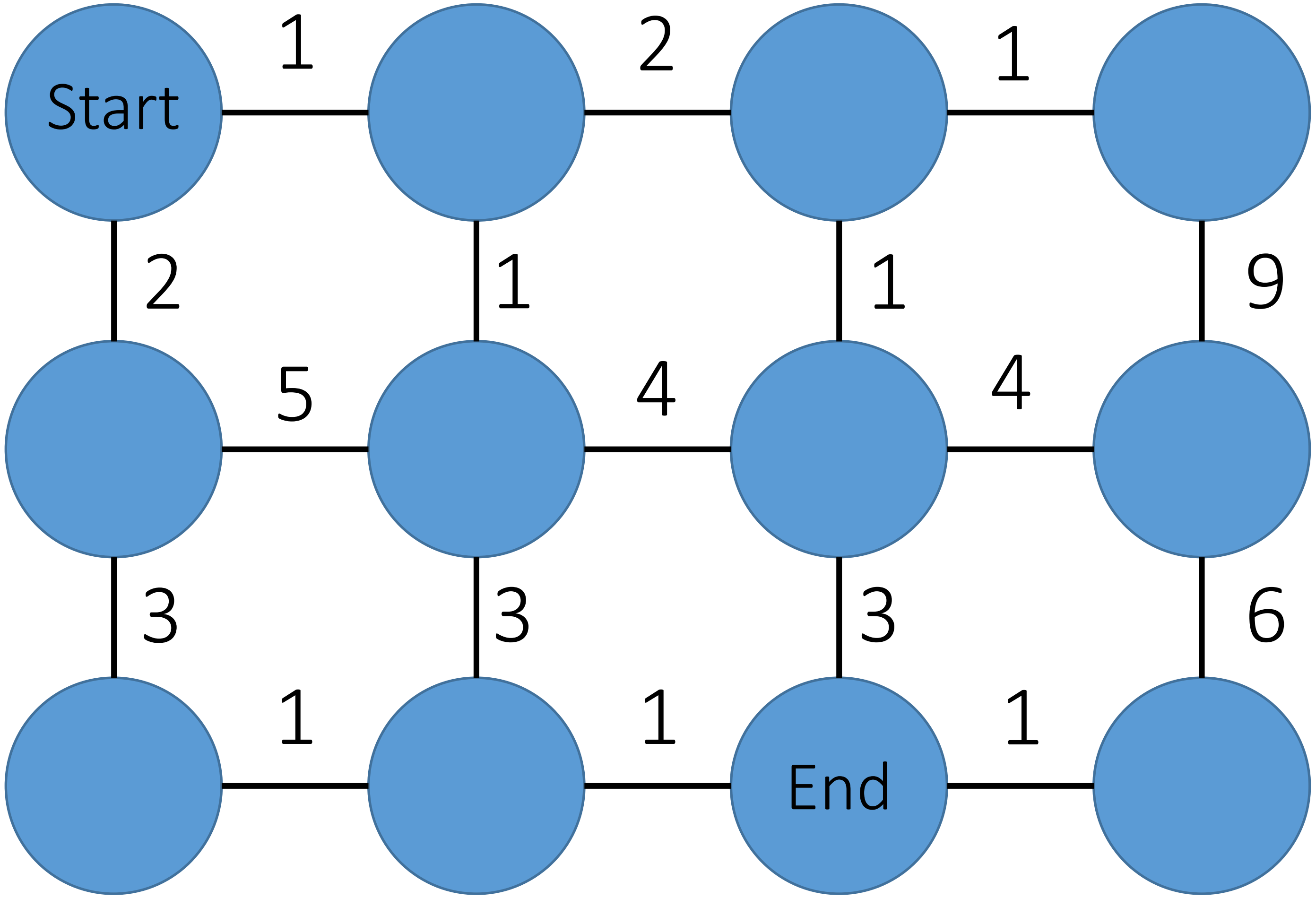
Graph-view of intelligent scissors:



1. Assign weights (costs) to edges

Graph-view of this problem

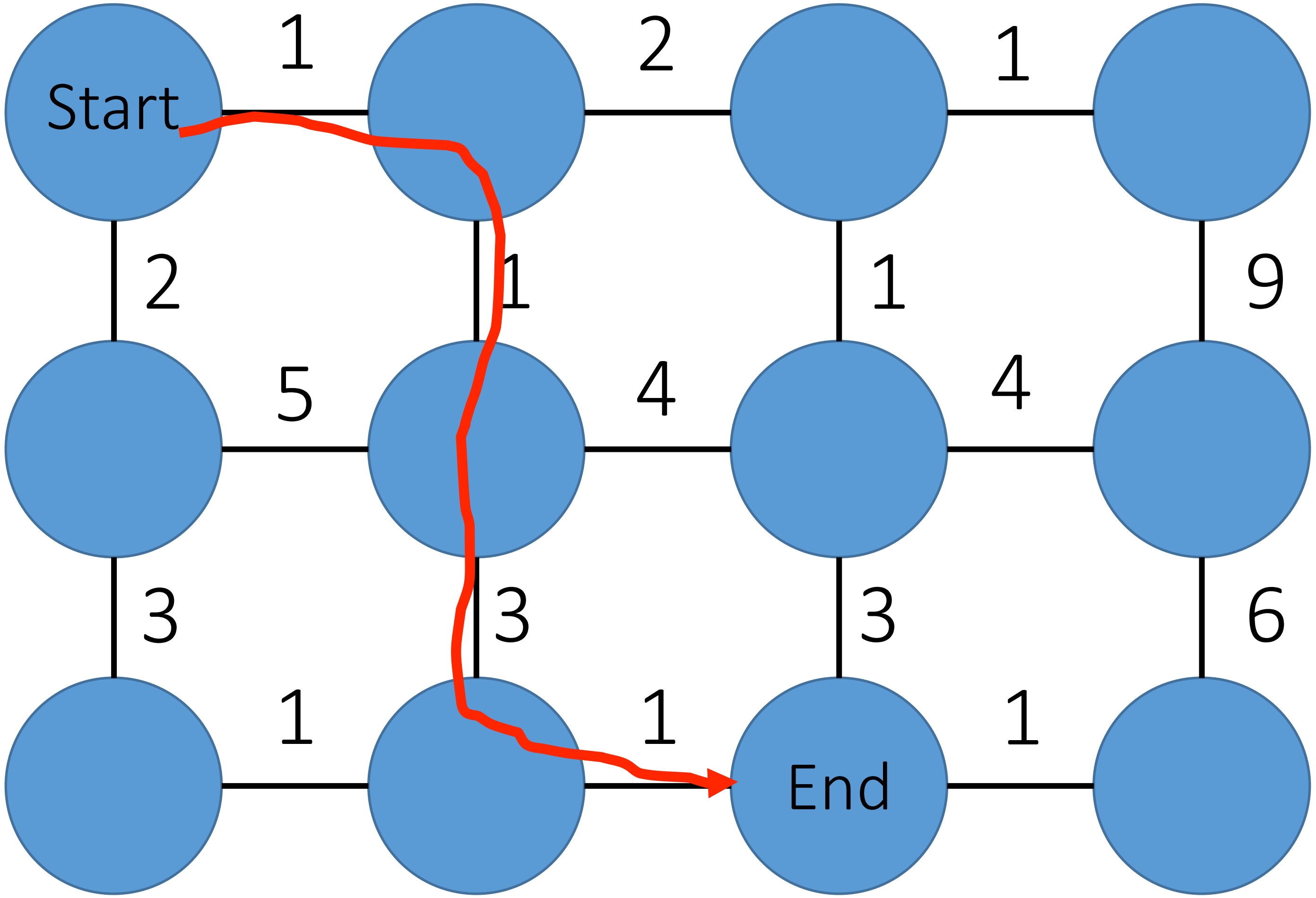
Graph-view of intelligent scissors:



1. Assign weights (costs) to edges
2. Select the seed nodes

Graph-view of this problem

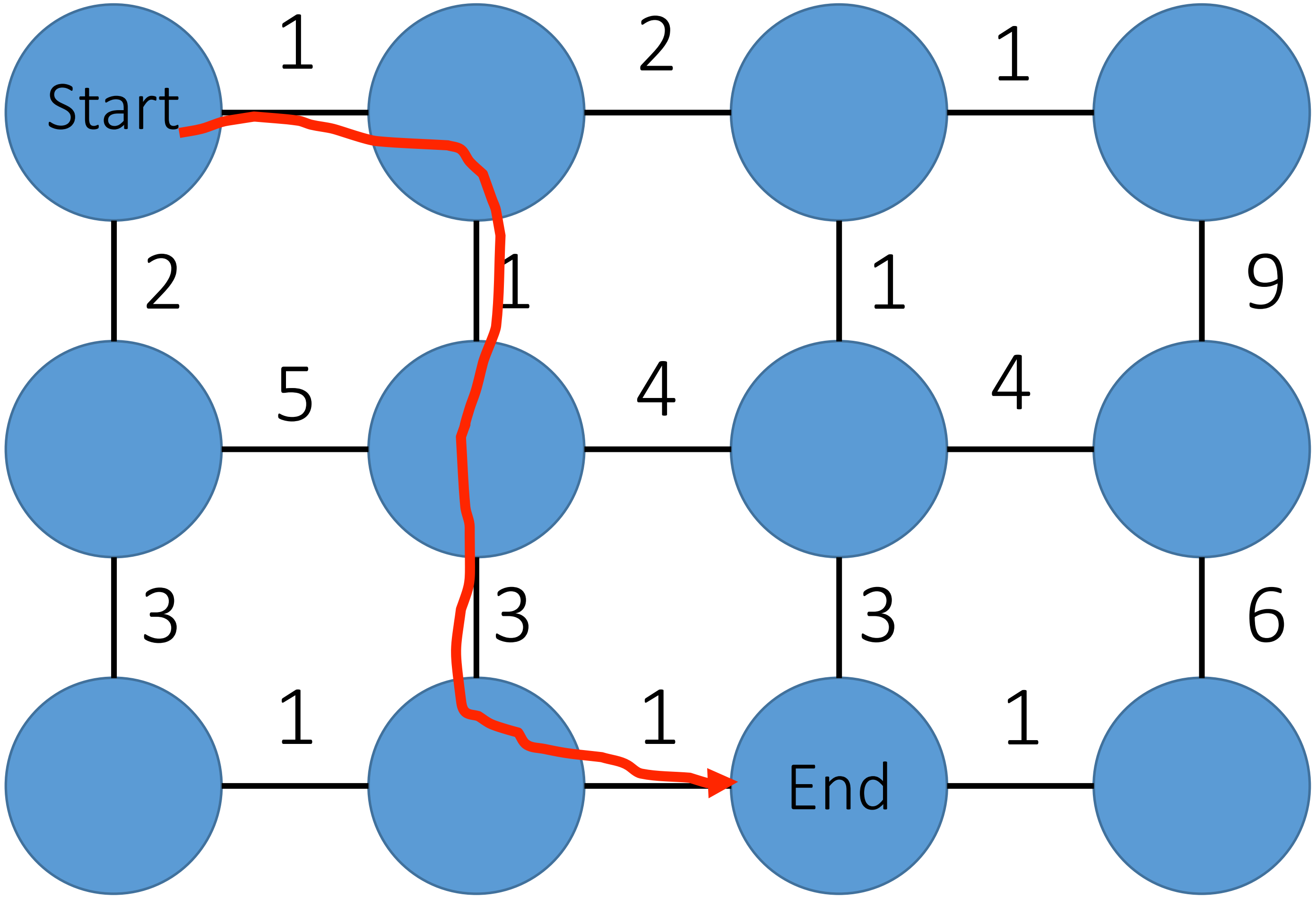
Graph-view of intelligent scissors:



1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

Graph-view of this problem

Graph-view of intelligent scissors:

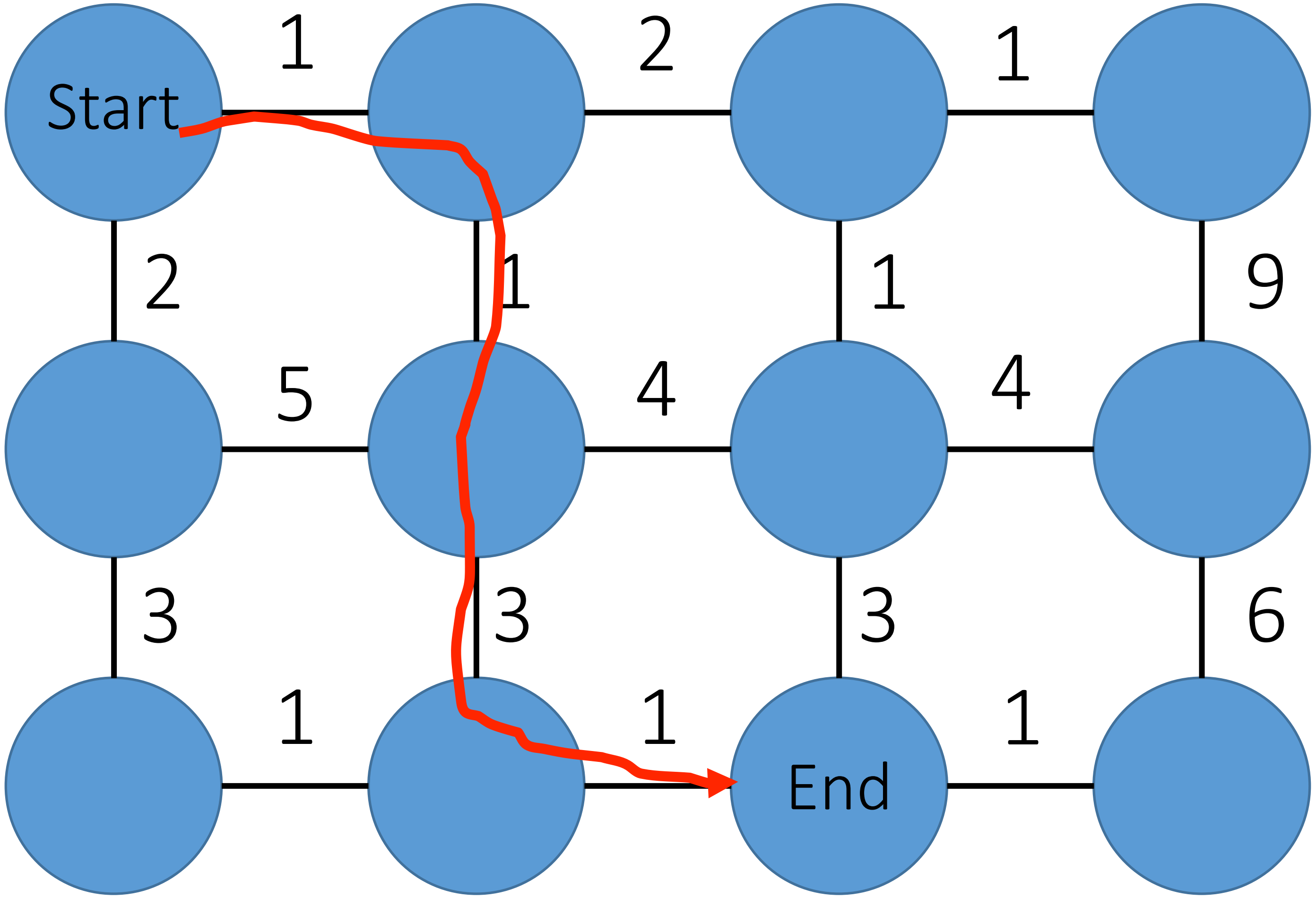


1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

What algorithm can we use to find the shortest path?

Graph-view of this problem

Graph-view of intelligent scissors:



1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

What algorithm can we use to find the shortest path?

- Dijkstra's algorithm (dynamic programming)

Dijkstra's shortest path algorithm

Initialize, given seed *s (pixel ID)*:

- cost(*s*) = 0 % total cost from seed to this point
- cost(!*s*) = big
- **A** = {*all pixels*} % set to be expanded
- **prev**(s)=undefined % pointer to pixel that leads to q=s

Precompute cost_2(q, r) % cost between *q* to neighboring pixel *r*

Loop while **A** is not empty

1.q = pixel in **A** with lowest cost

2.Remove q from **A**

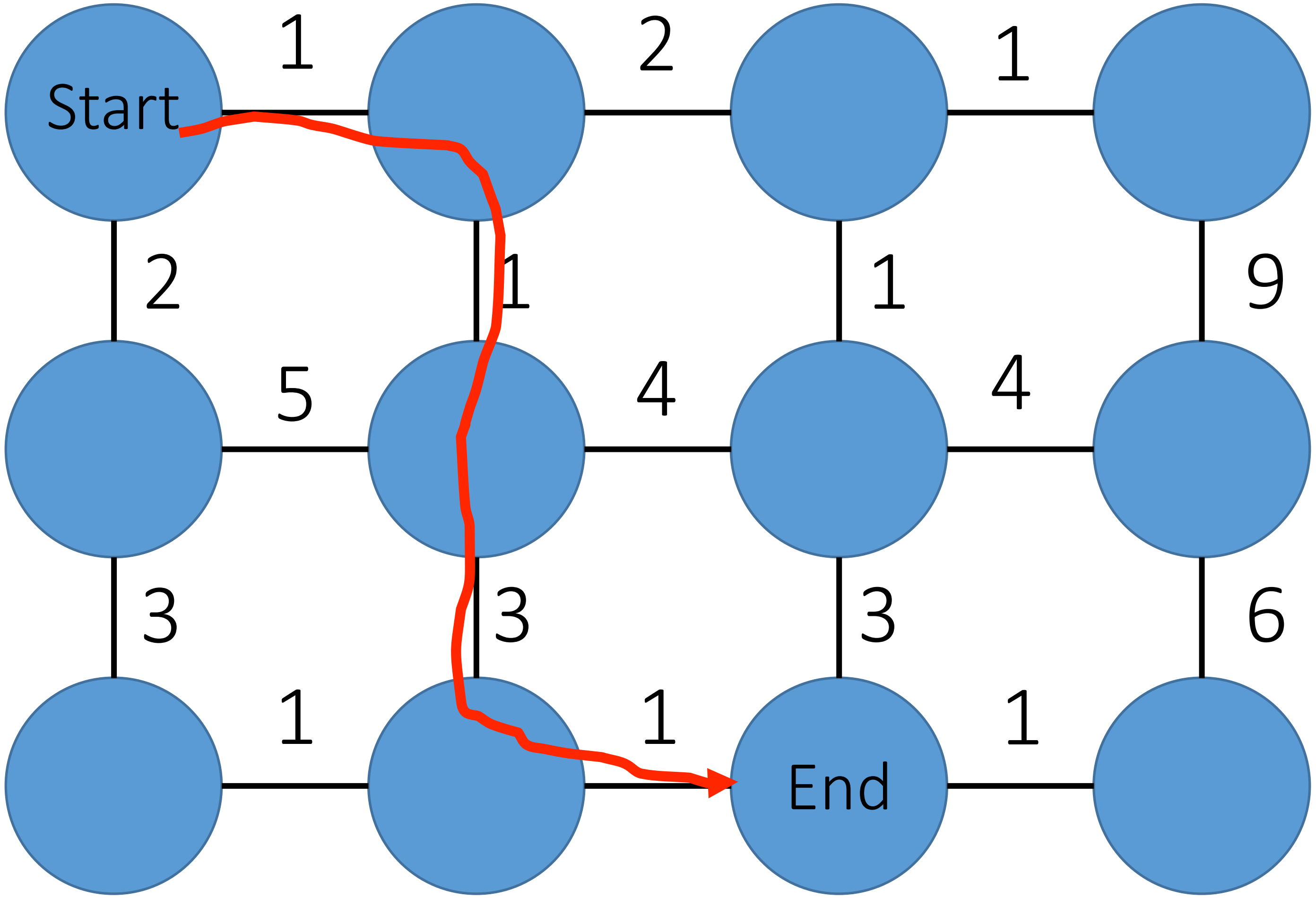
3.For each pixel *r* in neighborhood of *q* that is in **A**

  a) cost_tmp = cost(*q*) + cost_2(*q*,*r*) %this updates the costs

  b) if (cost_tmp < cost(*r*))
    i. cost(*r*) = cost_tmp
    ii. **prev**(*r*) = *q*

Graph-view of this problem

Graph-view of intelligent scissors:



1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

What algorithm can we use to find the shortest path?

- Dijkstra's algorithm (dynamic programming)

How should we select the edge weights to get good boundaries?

Selecting edge weights

Define boundary cost between neighboring pixels:

1. Lower if an image edge is present (e.g., as found by Sobel filtering).

2. Lower if the gradient magnitude at that point is strong.

3. Lower if gradient is similar in boundary direction.

Selecting edge weights

Gradient magnitude

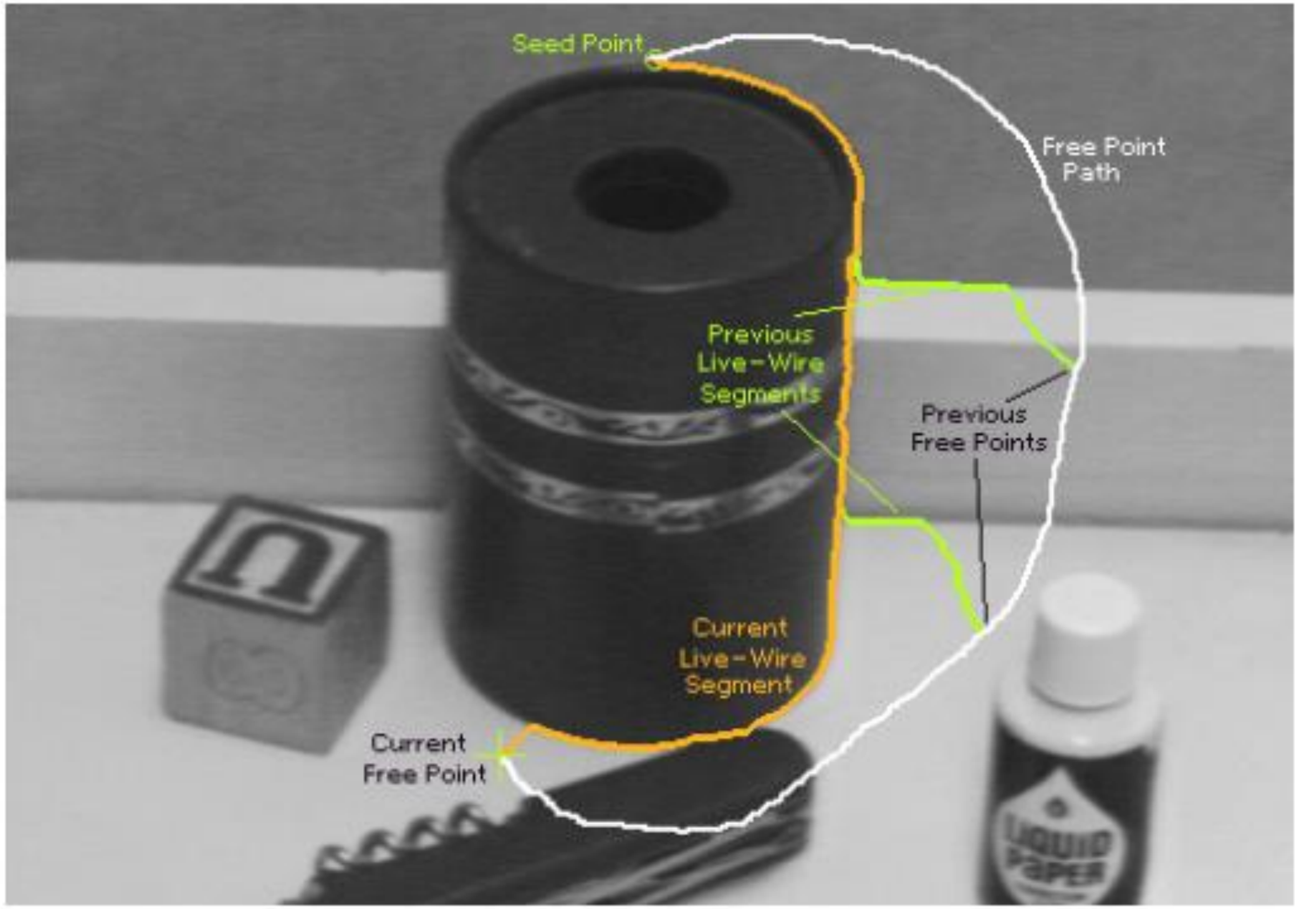
Pixel-wise cost

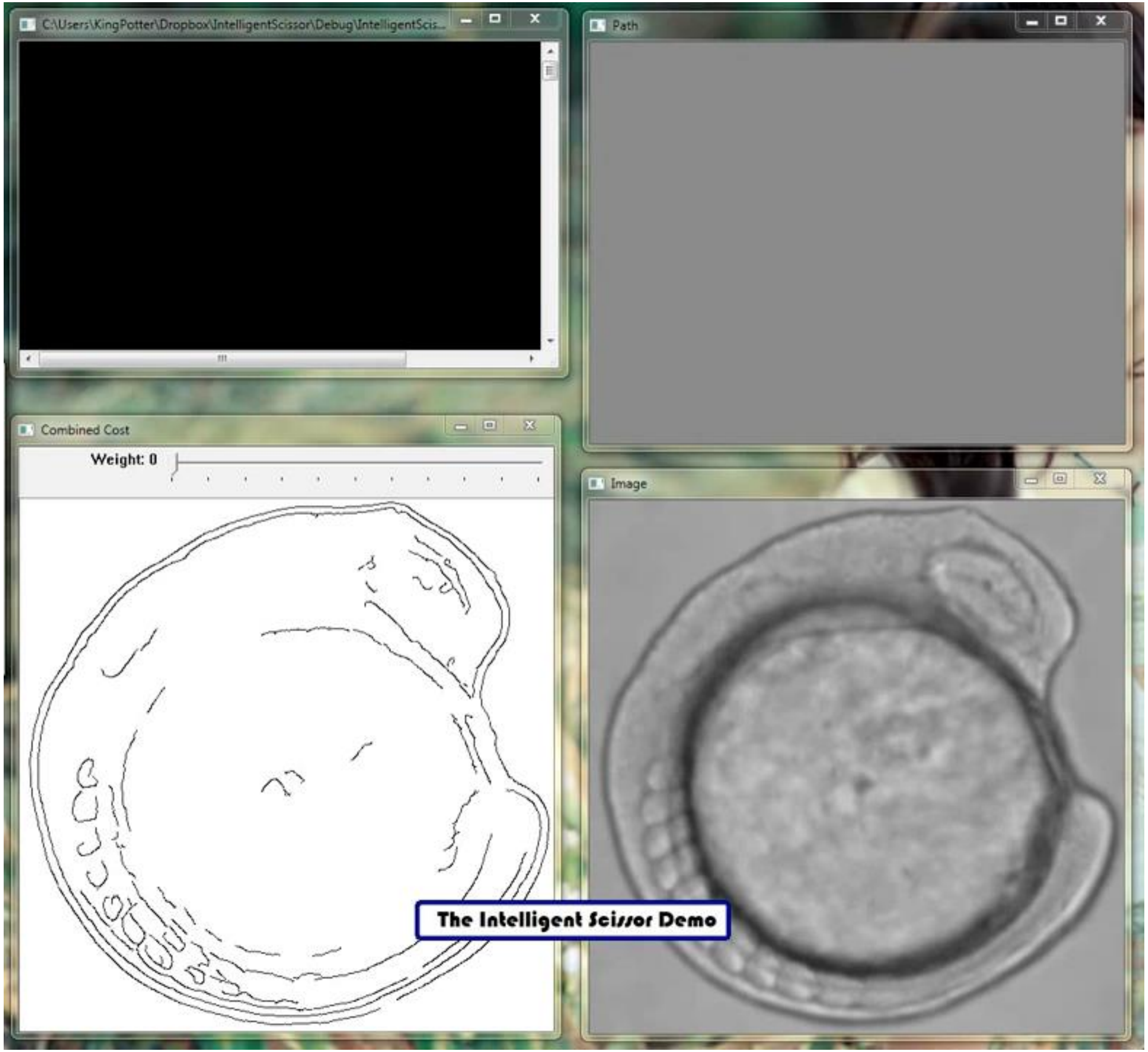
Edge image

Making it more interactive

1. Use cursor as the "end" seed, and always connect start seed to that

2. Every time the user clicks, use that point as a new starting seed and repeat

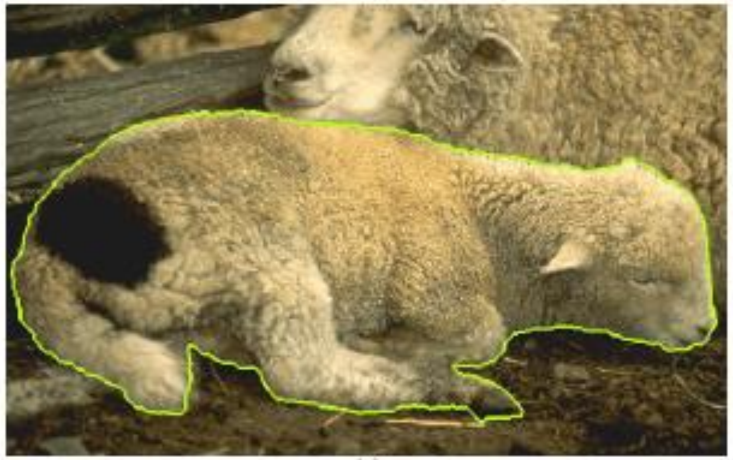



C:\Users\KingPotter\Dropbox\IntelligentScissor\Debug\IntelligentScis...
Path
Combined Cost
Weight: 0
Image
The Intelligent Scissor Demo

Examples

(a)

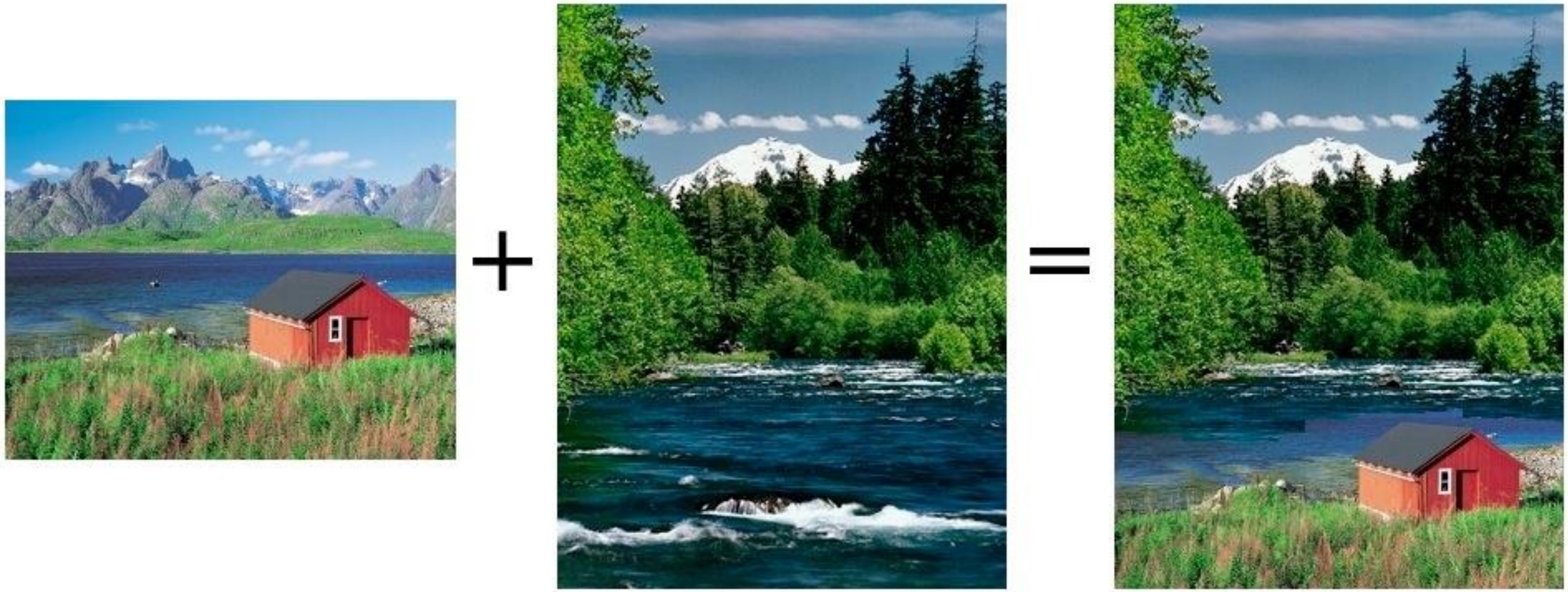
(b)



(c)

Seam collaging

Another use for image seam selection



Kwatra et al., Graphcut Textures: Image and Video Synthesis using Graph Cuts, SIGGRAPH 2003

Selecting edge weights for seam collaging

Good places to cut:
- similar color in both images
- high gradient in both images

Seam carving

Another use for image seam selection

Avidan and Shamir, Seam Carving for Content-Aware Image Resizing, SIGGRAPH 2007

Shai Avidan

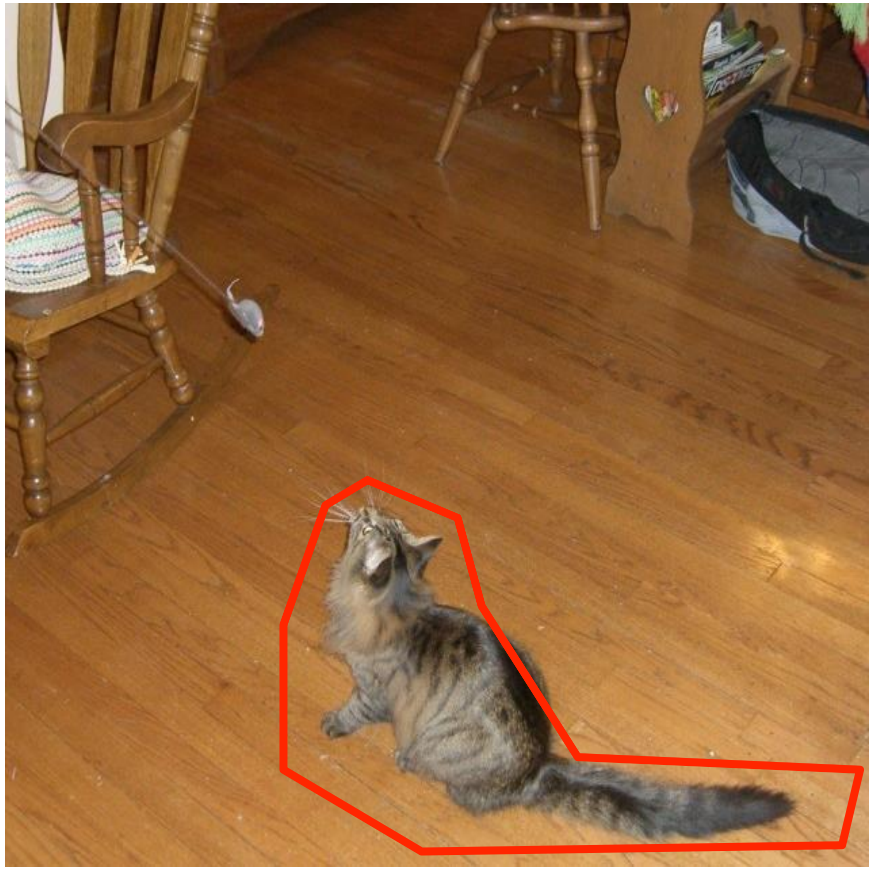
Mitsubishi Electric Research Lab

Ariel Shamir

The interdisciplinary Center & MERL

Examples

Where will intelligent scissors work well, or have problems?


Graph-cuts and GrabCut

GrabCut

Only user input is the box!

grab cut paste

Rother et al., “Interactive Foreground Extraction with Iterated Graph Cuts,” SIGGRAPH 2004

Combining region and boundary information

Magic Wand (198?) | Intelligent scissors | GrabCut

user input

result

regions | boundary | regions & boundary

GrabCut is a mixture of two components

1. Segmentation using graph cuts

2. Foreground-background modeling using unsupervised clustering

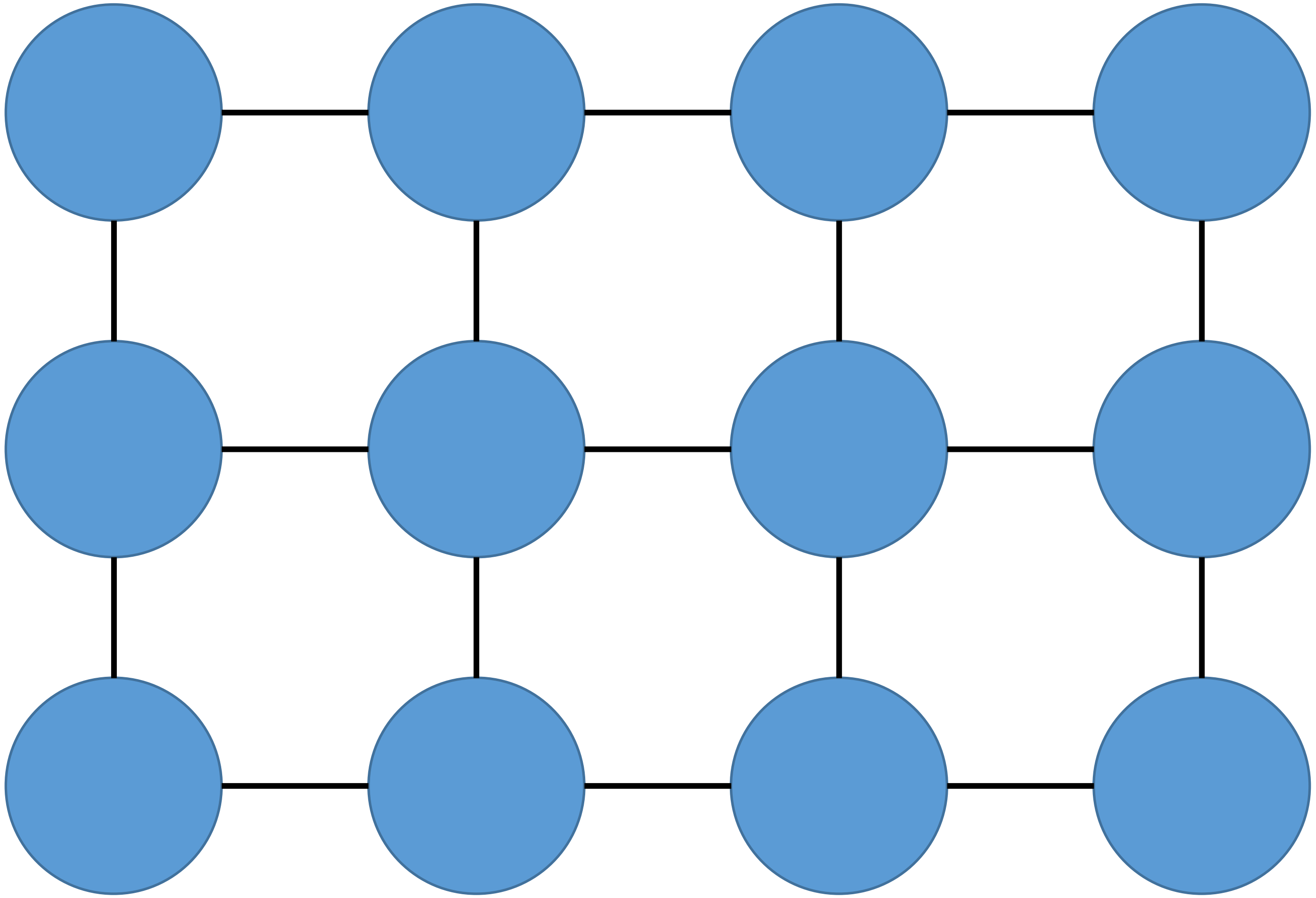
GrabCut is a mixture of two components

1. $\boxed{\text{Segmentation using graph cuts}}$

2. Foreground-background modeling using unsupervised clustering

Segmentation using graph cuts

Remember: Graph-based view of images



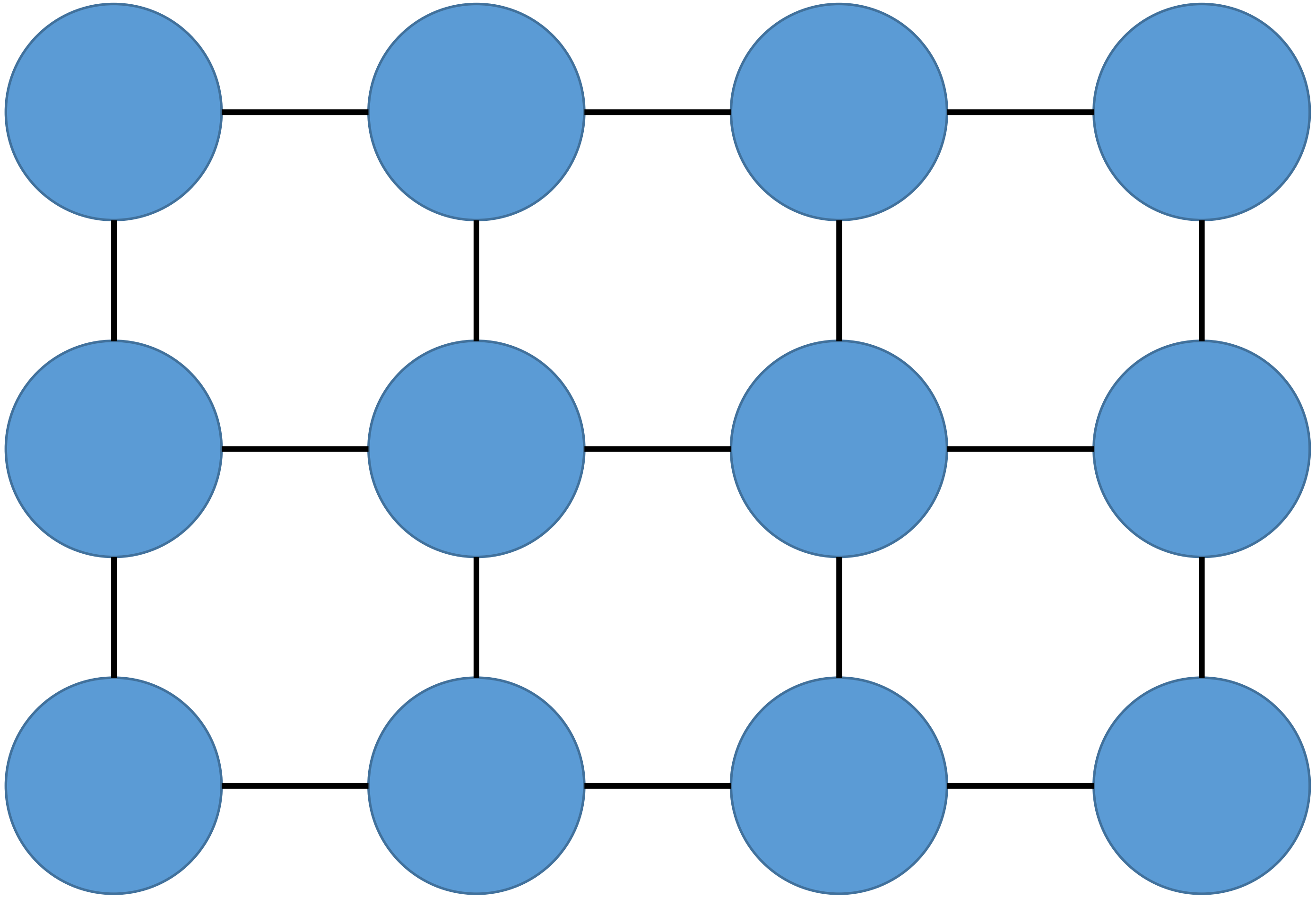
Nodes: pixels

Edges: Constraints between neighboring pixels

Markov Random Field (MRF)

Assign foreground/background labels based on:

$$Energy(\mathbf{y};\theta,data)=\sum_{i}\psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

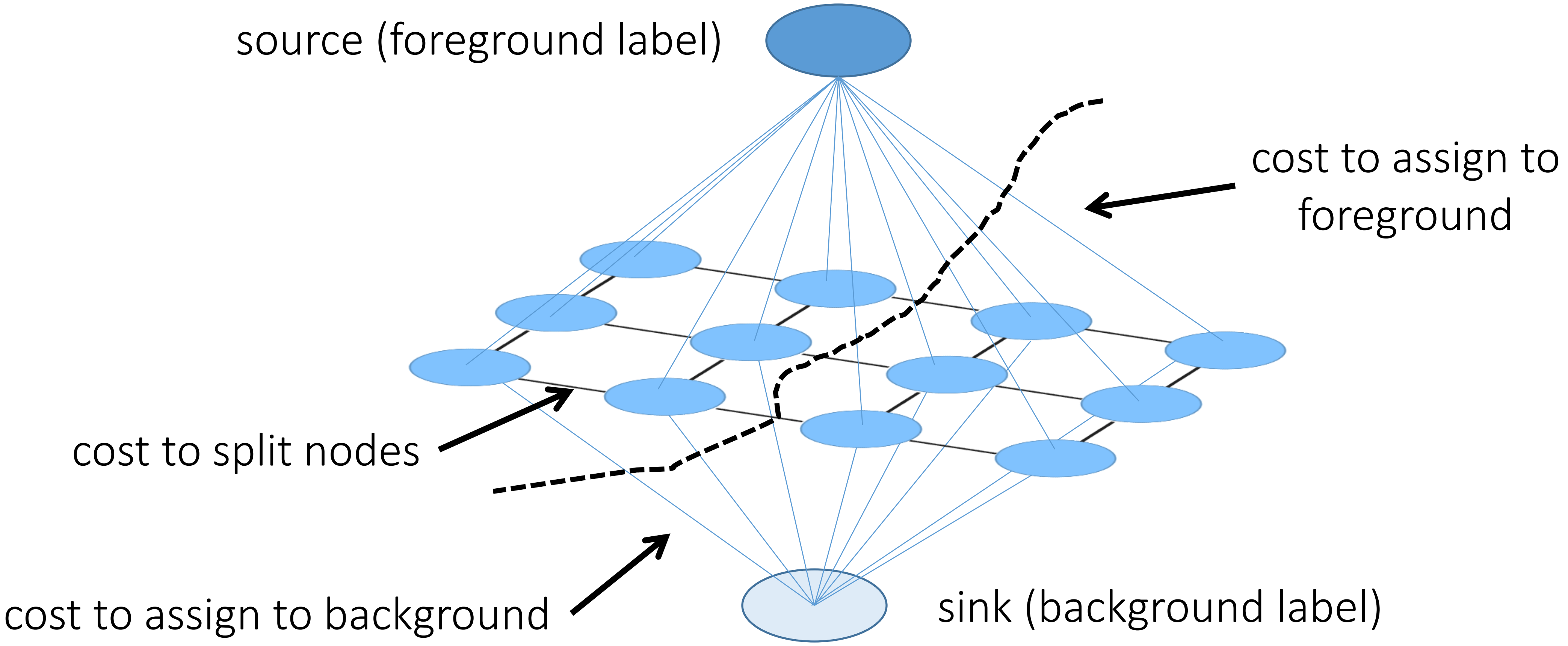


Given its intensity value, how likely is a pixel to be foreground or background?

Given their intensity values, how likely are two neighboring pixels to have two labels?

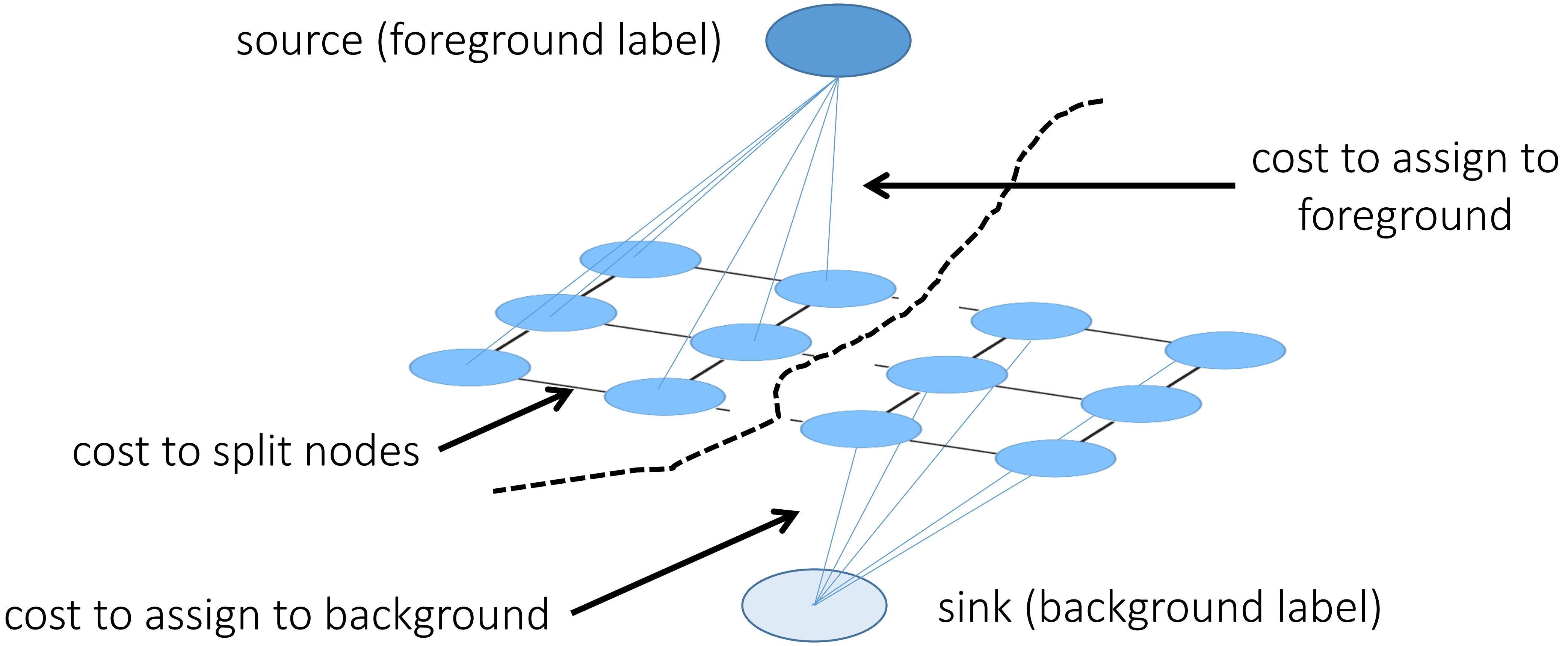
What kind of cost functions would you use for GrabCut?

Solving MRFs using max-flow/min-cuts (graph cuts)



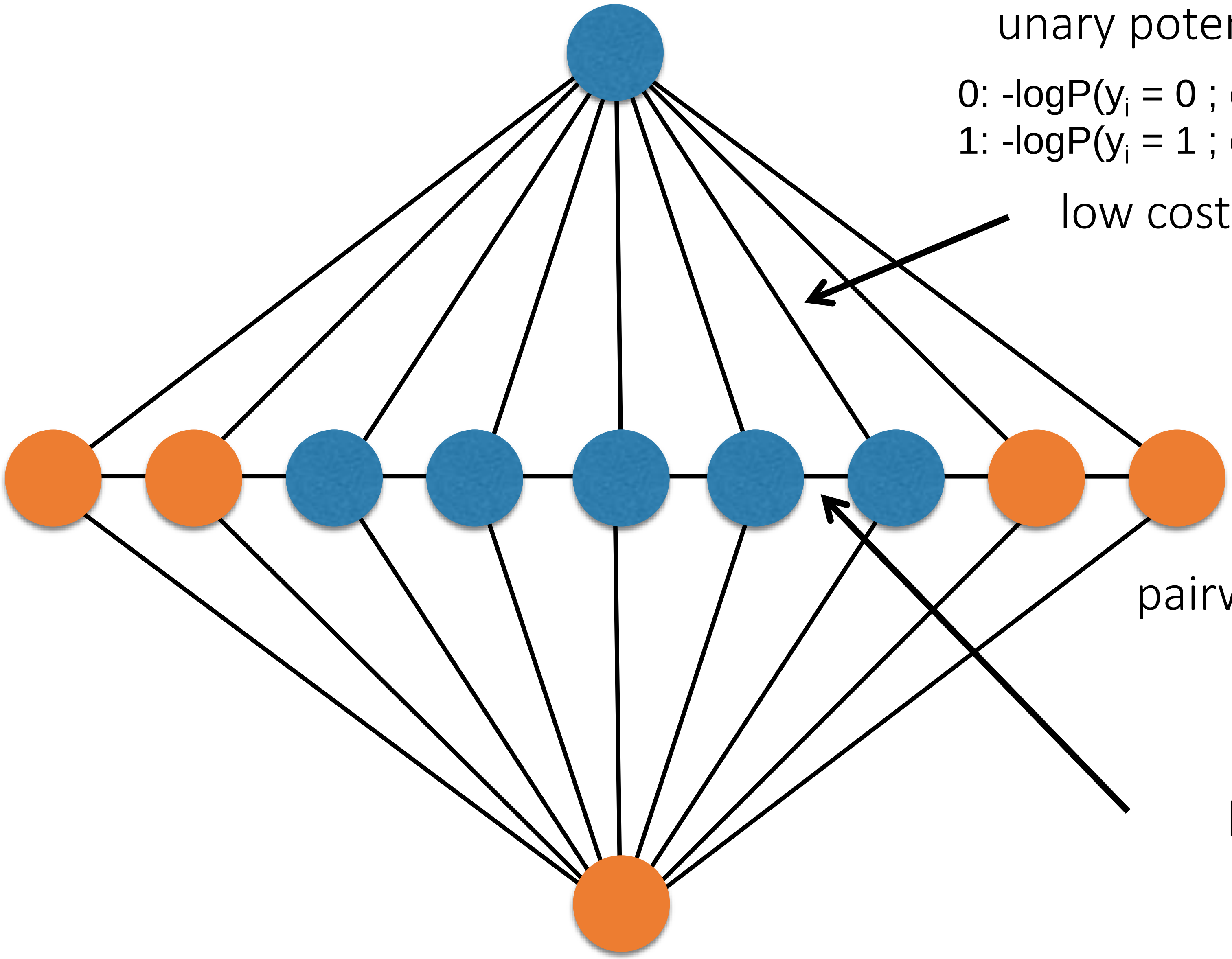
$$Energy(\mathbf{y};\theta,data)=\sum_{i}\psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

Solving MRFs using max-flow/min-cuts (graph cuts)



$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

A toy visual example



unary potential

0: -logP(y_i = 0 ; data)
1: -logP(y_i = 1 ; data)

low cost

pairwise potential

	0	1
0	0	K
1	K	0

low cost

Graph-cuts segmentation

1. Define graph
   - usually 4-connected or 8-connected
2. Set weights to foreground/background

How would you determine these for GrabCut?

$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Set weights for edges between pixels

$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\left\|c(x)-c(y)\right\|^2}{2\sigma^2}\right\}$$

4. GraphCut: Apply min-cut/max-flow algorithm

GrabCut is a mixture of two components

1. Segmentation using graph cuts

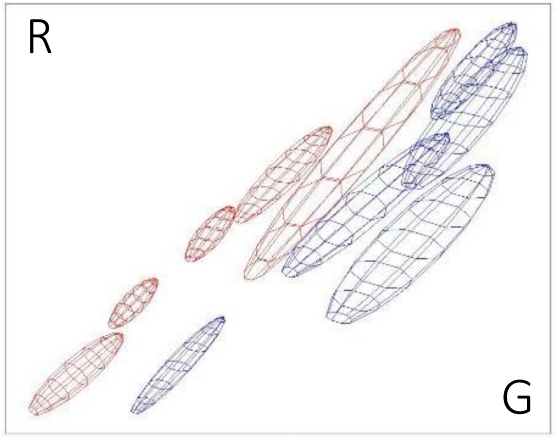
2. $\boxed{\text{Foreground-background modeling using unsupervised clustering}}$

Foreground-background modeling

Given foreground/background labels

build a color model for both

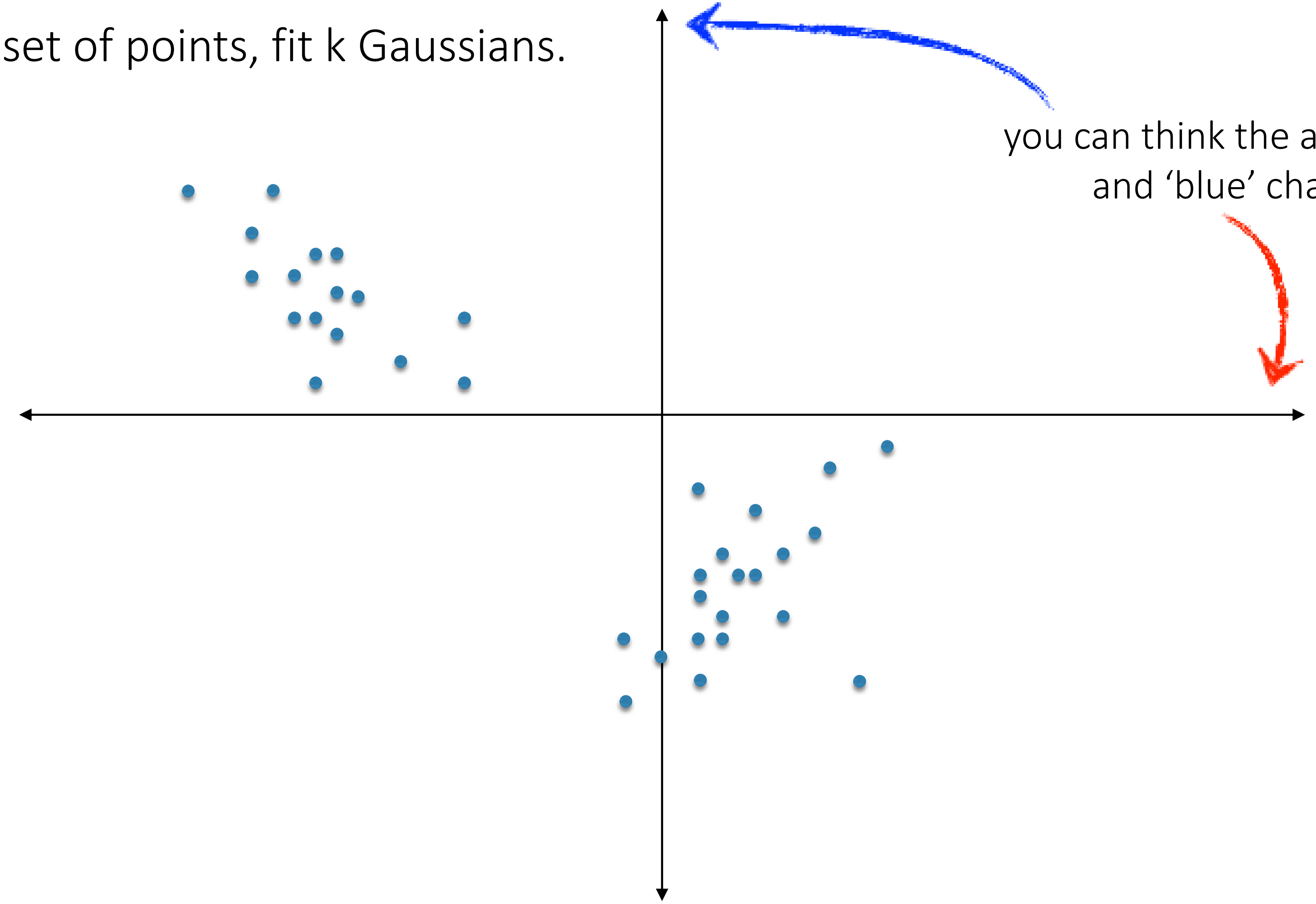
foreground



background

Learning color models

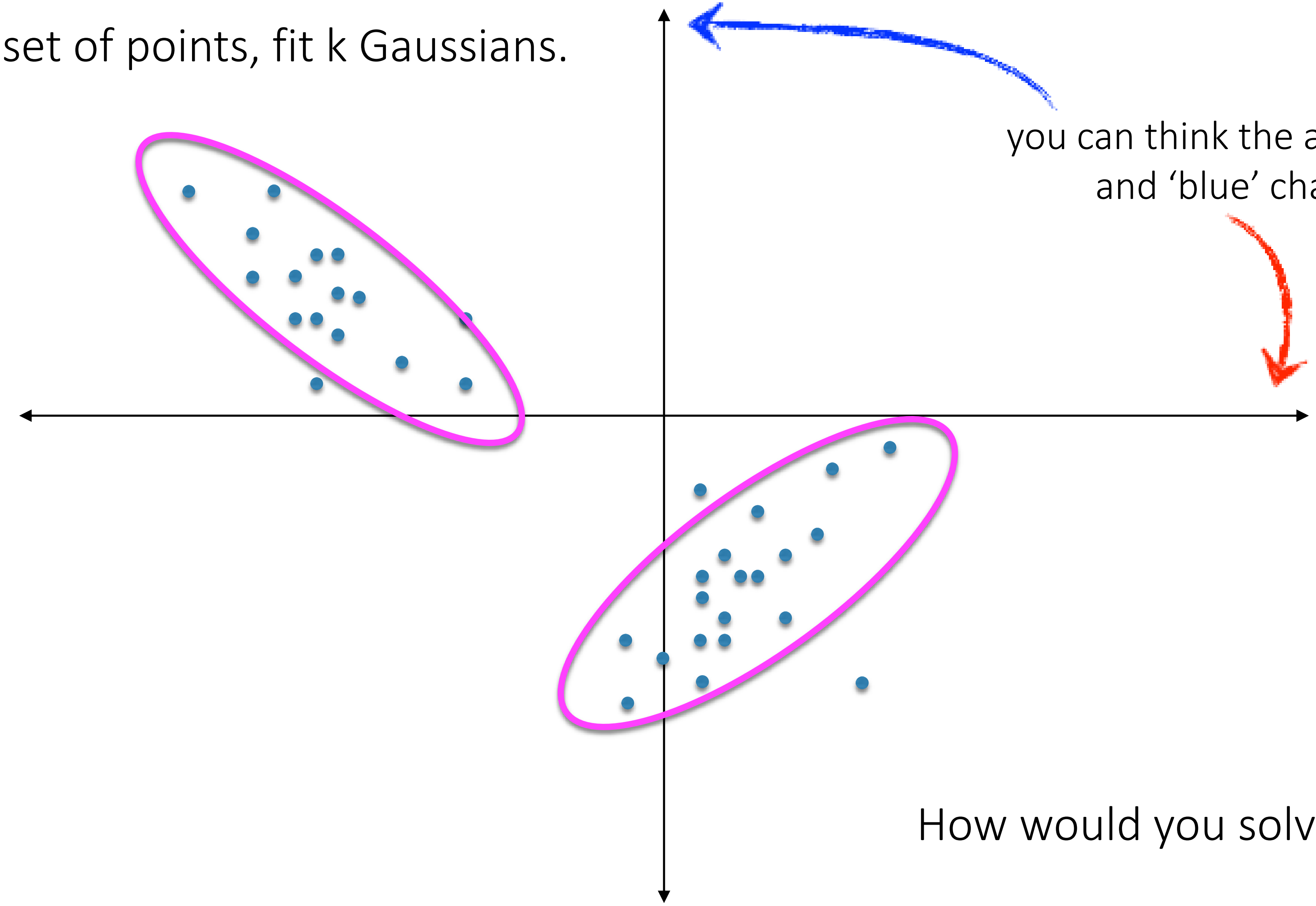
Given a set of points, fit k Gaussians.



you can think the axes as ‘red’ and ‘blue’ channels

Learning color models

Given a set of points, fit k Gaussians.



you can think the axes as ‘red’ and ‘blue’ channels

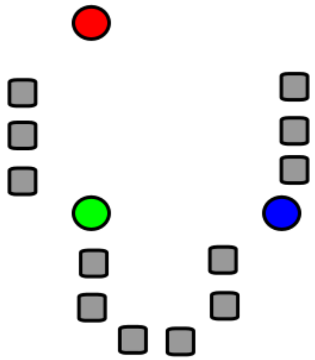
How would you solve this problem?

Intuition: “hard” clustering using K-means

Given k:

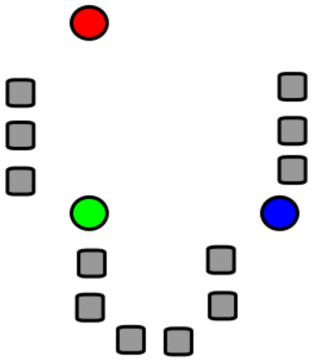
1. Select initial centroids at random.
2. Assign each object to the cluster with the nearest centroid.
3. Compute each centroid as the mean of the objects assigned to it.
4. Repeat previous 2 steps until no change.

K-means visualization

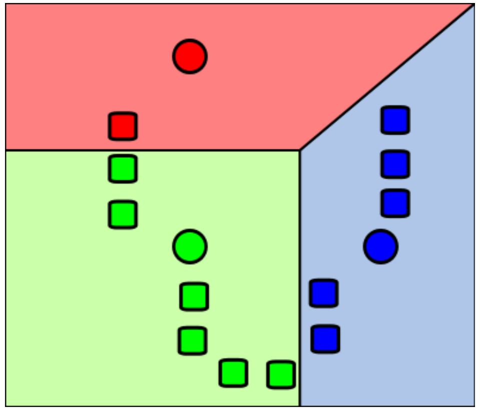


1. Select initial centroids at random

K-means visualization

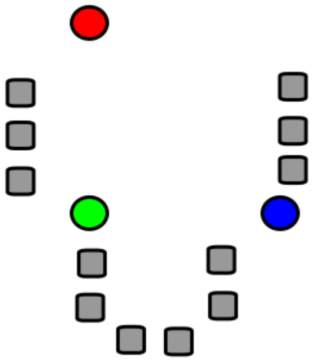


1. Select initial centroids at random

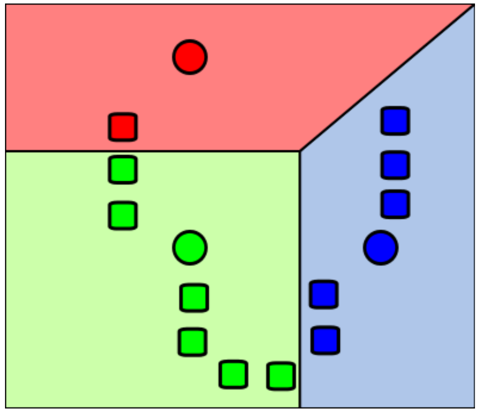


2. Assign each object to the cluster with the nearest centroid.

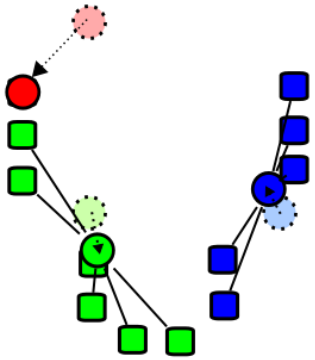
K-means visualization



1. Select initial centroids at random

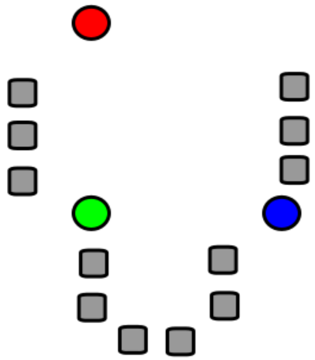


2. Assign each object to the cluster with the nearest centroid.

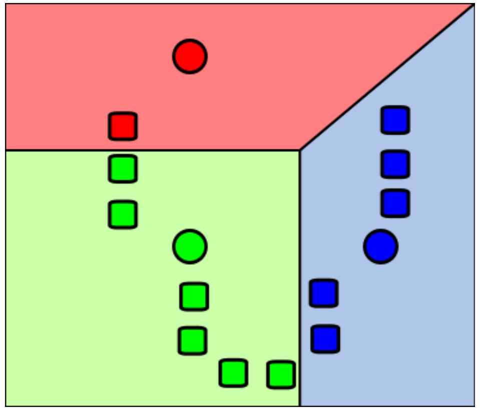


3. Compute each centroid as the mean of the objects assigned to it (go to 2)

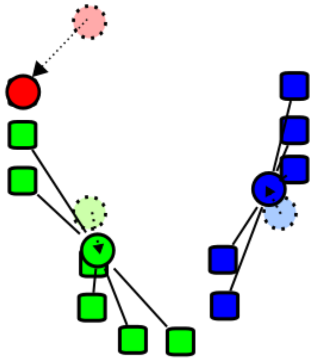
K-means visualization



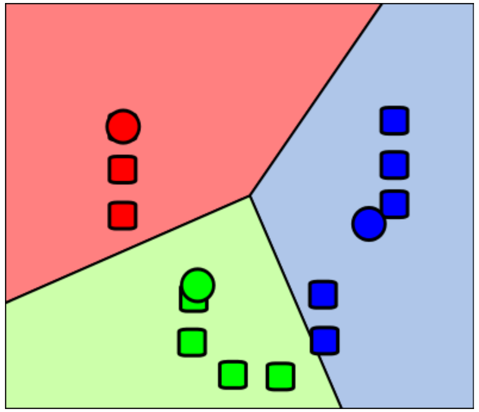
1. Select initial centroids at random



2. Assign each object to the cluster with the nearest centroid.



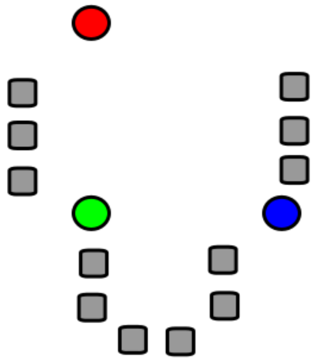
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



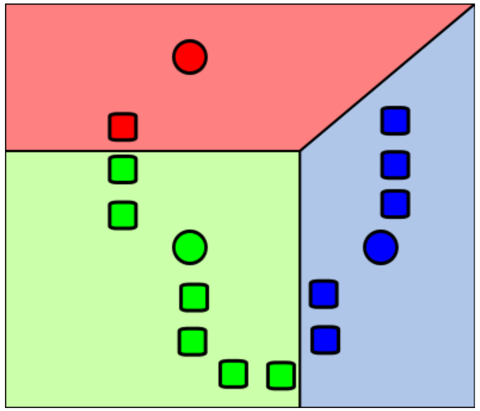
2. Assign each object to the cluster with the nearest centroid.

Repeat previous 2 steps until no change

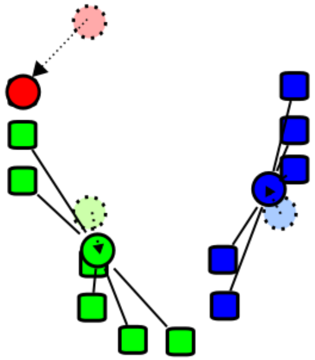
K-means visualization



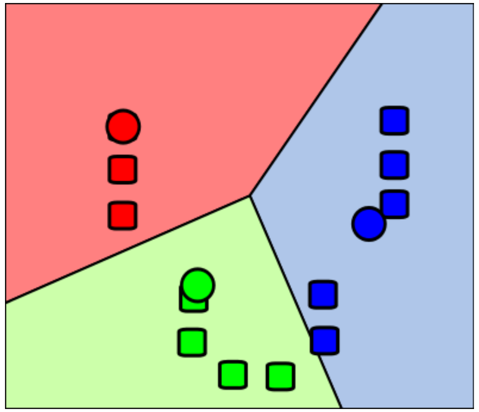
1. Select initial centroids at random



2. Assign each object to the cluster with the nearest centroid.



3. Compute each centroid as the mean of the objects assigned to it (go to 2)



2. Assign each object to the cluster with the nearest centroid.

Expectation-Maximization: "soft" version of K-means

E-step

M-step

```
Given k:
1.Select initial centroids at
 random.
```

compute the probability of each object being in a cluster

```
2.Assign each object to the cluster
 with the nearest centroid.
```

and covariance

```
3.Compute each centroid as the mean
 of the objects assigned to it.
```

weighed by the probability of being in that clust

```
4.Repeat previous 2 steps until no
 change.
```

Unsupervised clustering

Model: **Mixture of Gaussians**

Algorithm: **Expectation Maximization**

E step

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta};\mathbf{X},\mathbf{Z})\right]$$

Compute the expected log-likelihood

M step

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\arg\max}\ Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

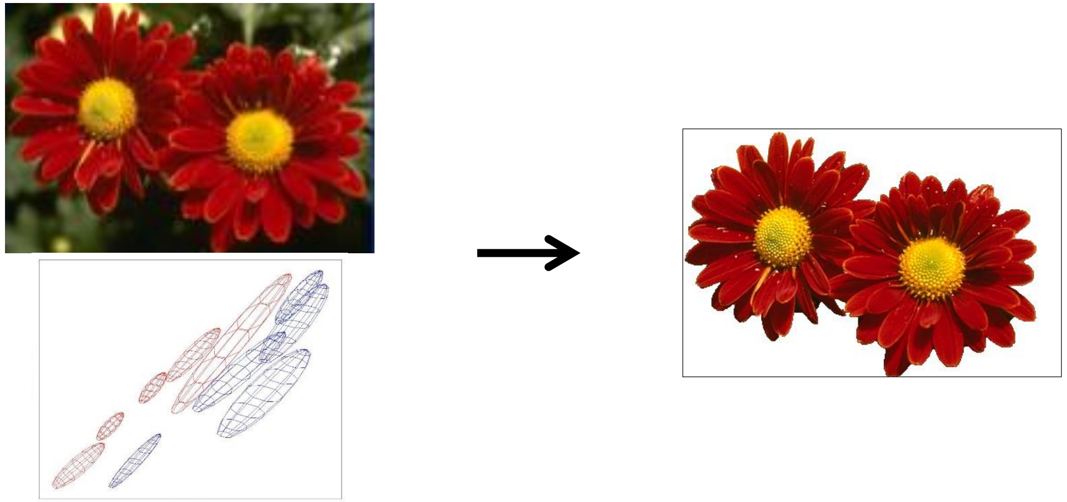
Update parameters based on likelihood

Important result for GrabCut:
we can compute the **likelihood** of a **pixel** belonging to the **foreground** or **background** as:

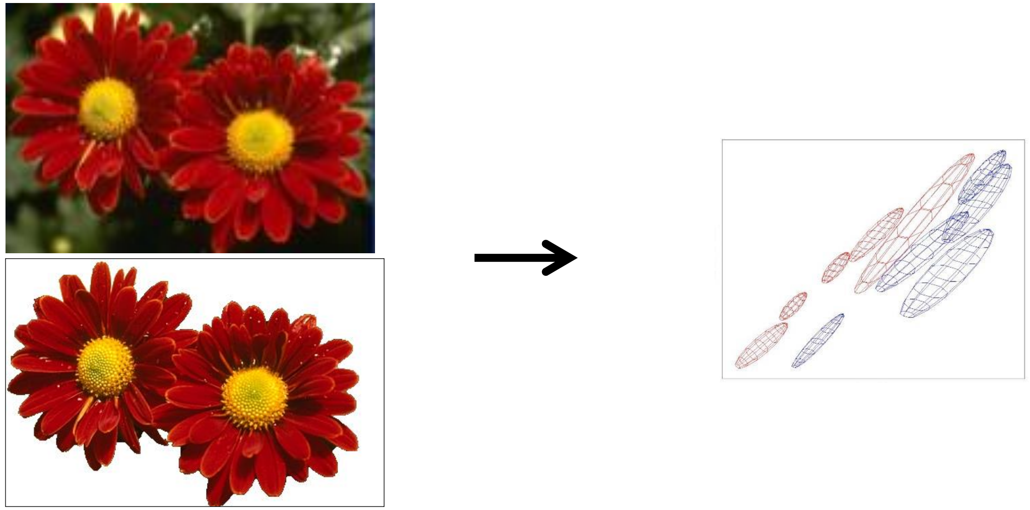
$$p(c(x);\boldsymbol{\theta}) = \prod_{k=1}^{K} \alpha_k \cdot \mathcal{N}(c(x);\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

GrabCut is a mixture of two components

1. Segmentation using graph cuts
    - Requires having foreground model



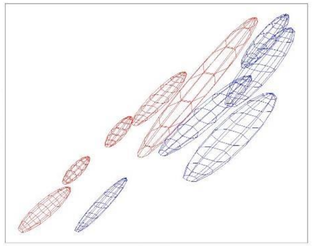
2. Foreground-background modeling using unsupervised clustering
    - Requires having segmentation



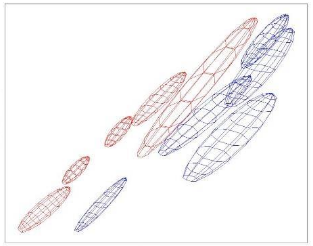
What do we do?

GrabCut: iterate between two steps

1. Segmentation using graph cuts
   - Requires having foreground model

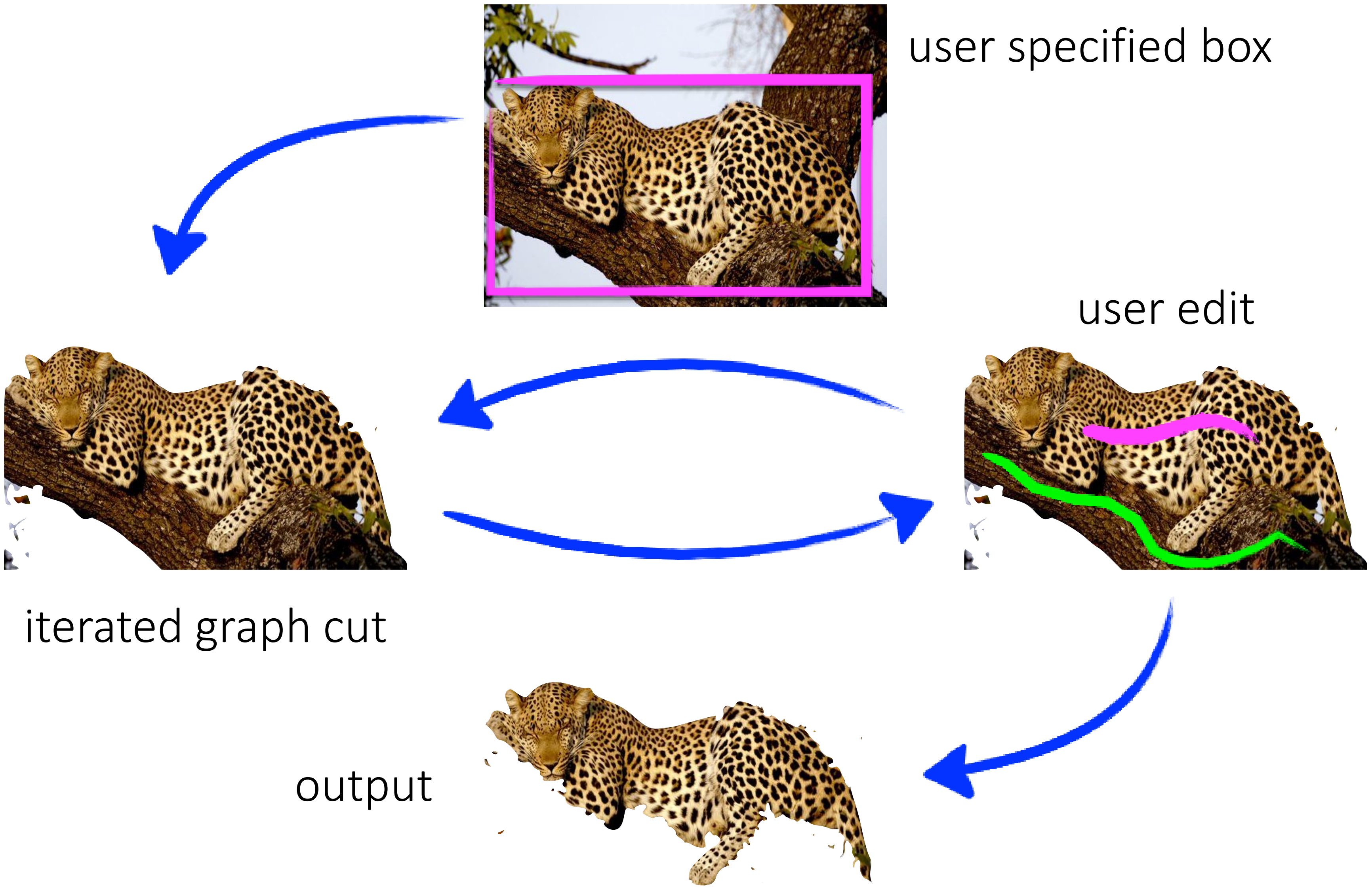


2. Foreground-background modeling using unsupervised clustering
   - Requires having segmentation

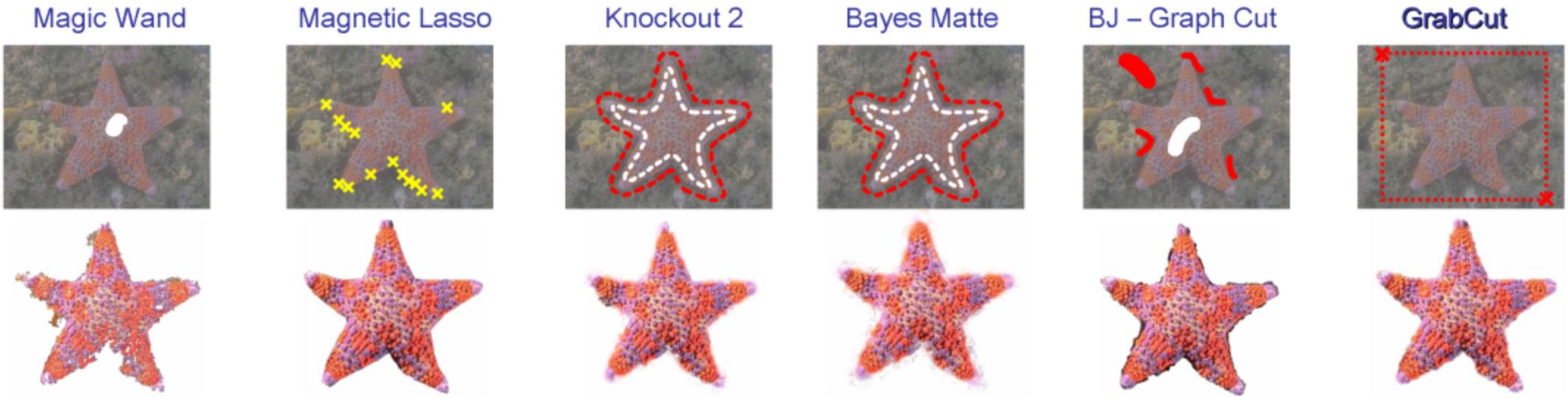


What do we do?

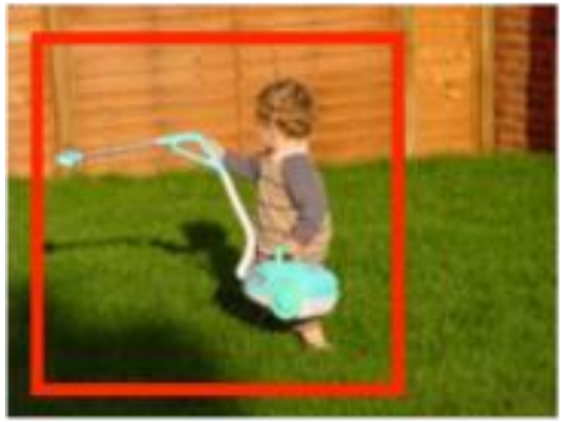
Iteration can be interactive



Examples


Magic Wand
Magnetic Lasso
Knockout 2
Bayes Matte
BJ – Graph Cut
GrabCut

Examples

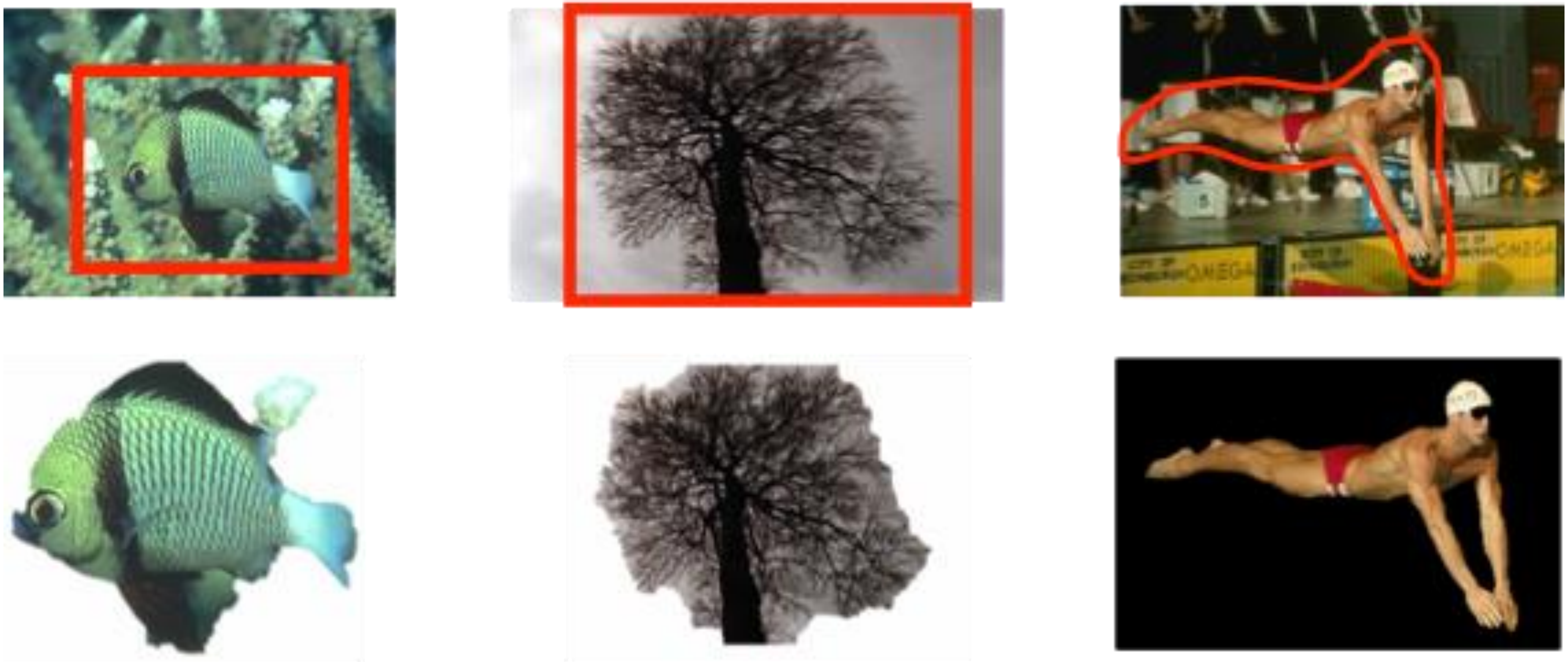





What is easy or hard about these cases for graph cut-based segmentation?

Examples

Examples

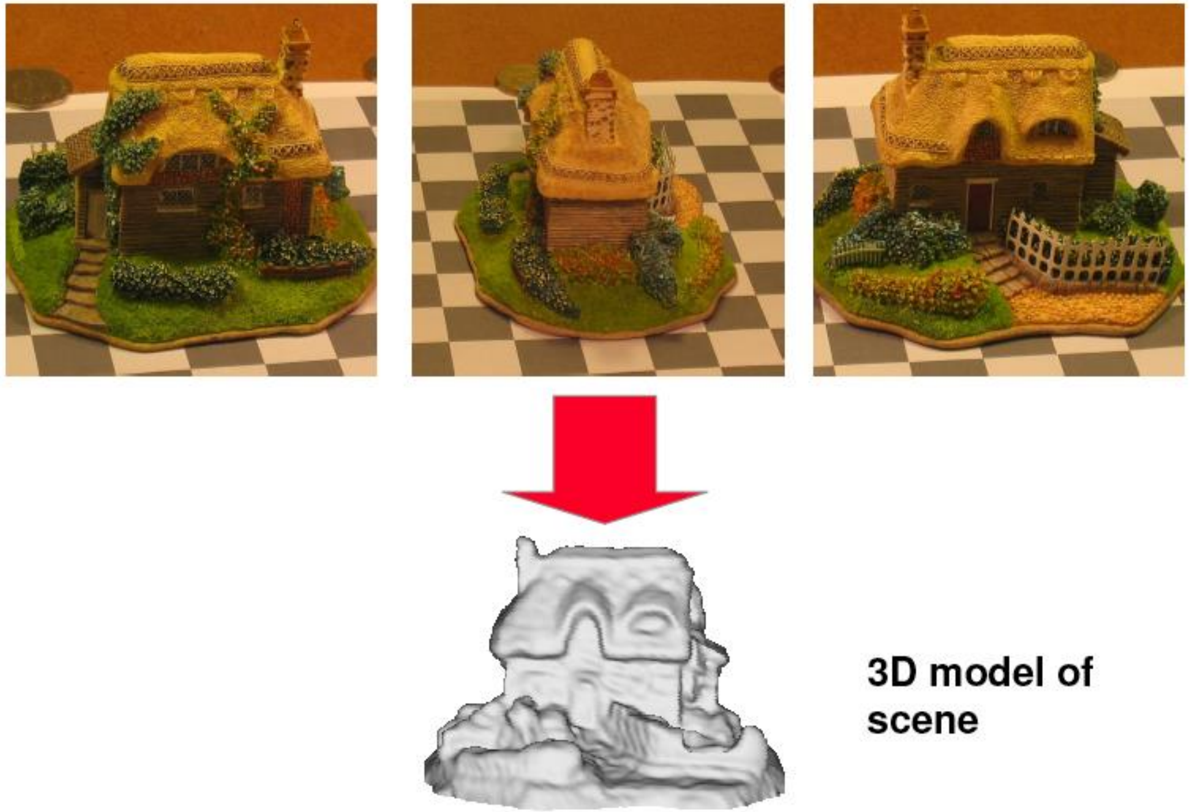


Examples

Lazy Snapping
[Li et al. SIGGRAPH 2004]

Graph-cuts are a very general, very useful tool

- denoising
- stereo
- texture synthesis
- segmentation
- classification
- recognition
- …



3D model of scene

References

Basic reading:
- Szeliski textbook, Sections 5.1.3, 5.3.1, 9.3.2, 9.3.3, 10.4.3.