The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Spring 2022

INTRODUCTION TO COMPUTER VISION

Atlas Wang
Assistant Professor, The University of Texas at Austin

Visual Informatics Group@UT Austin
https://vita-group.github.io/

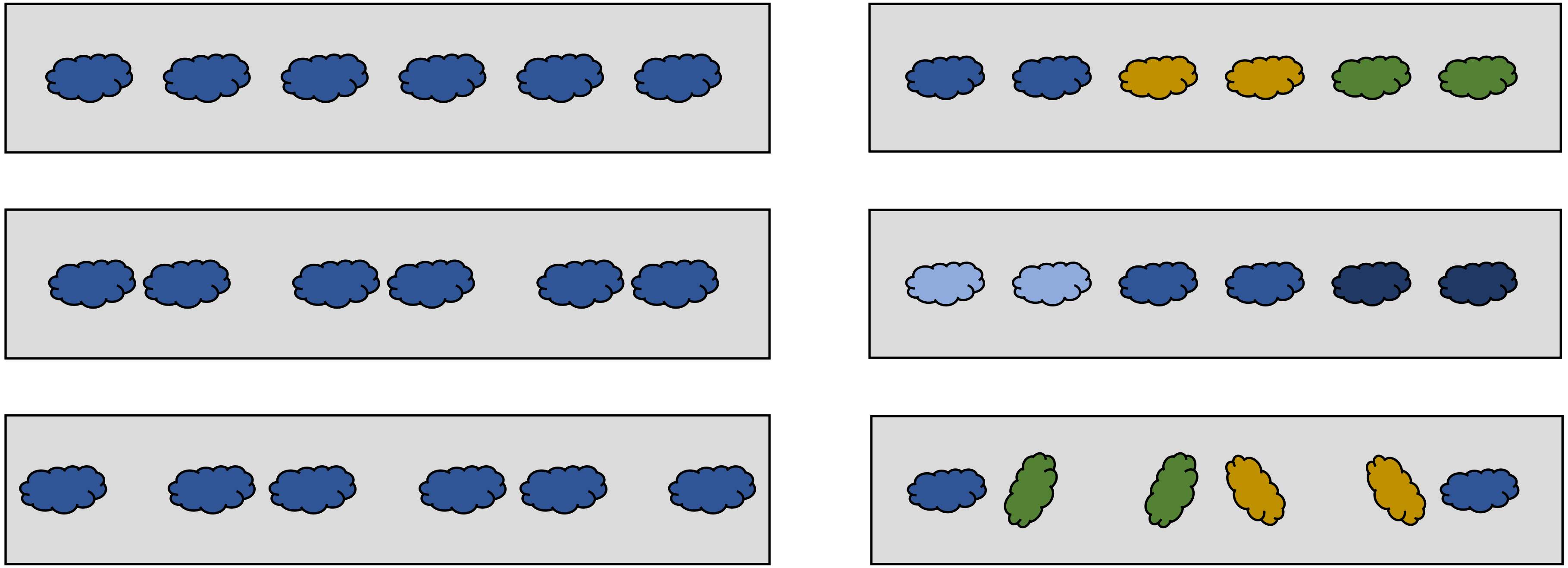
Many slides here were adapted from CMU 16-385

Gestalt Psychology

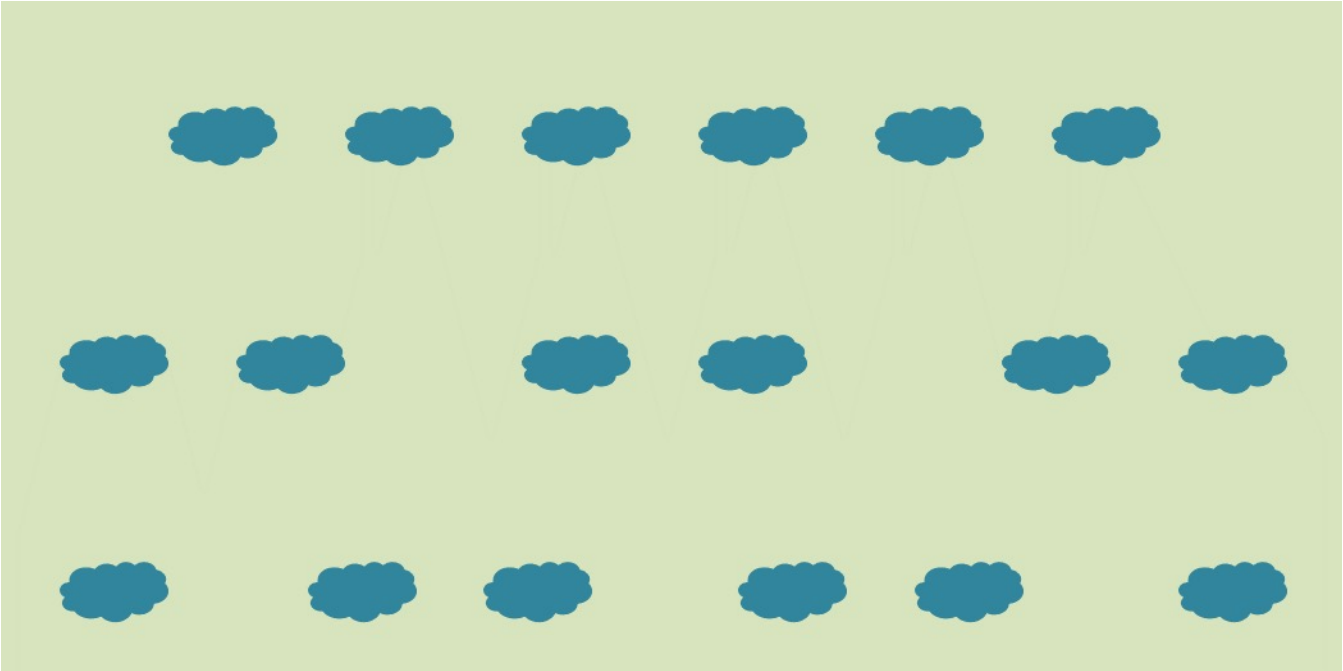
We perceive objects in their entirety before their individual parts.

Closer objects are grouped together

Similar objects are grouped together

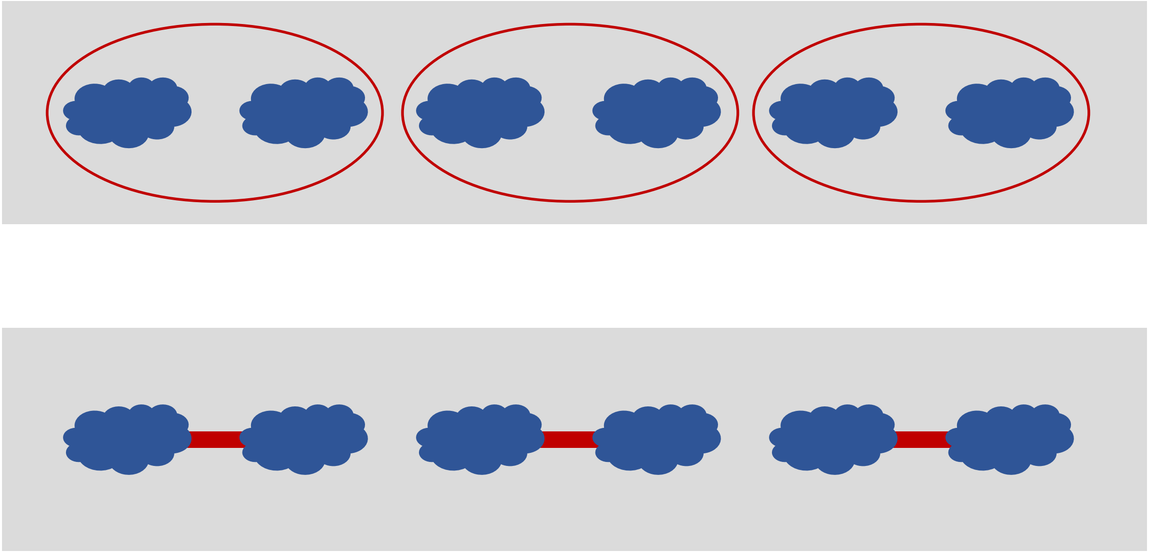


Common Fate



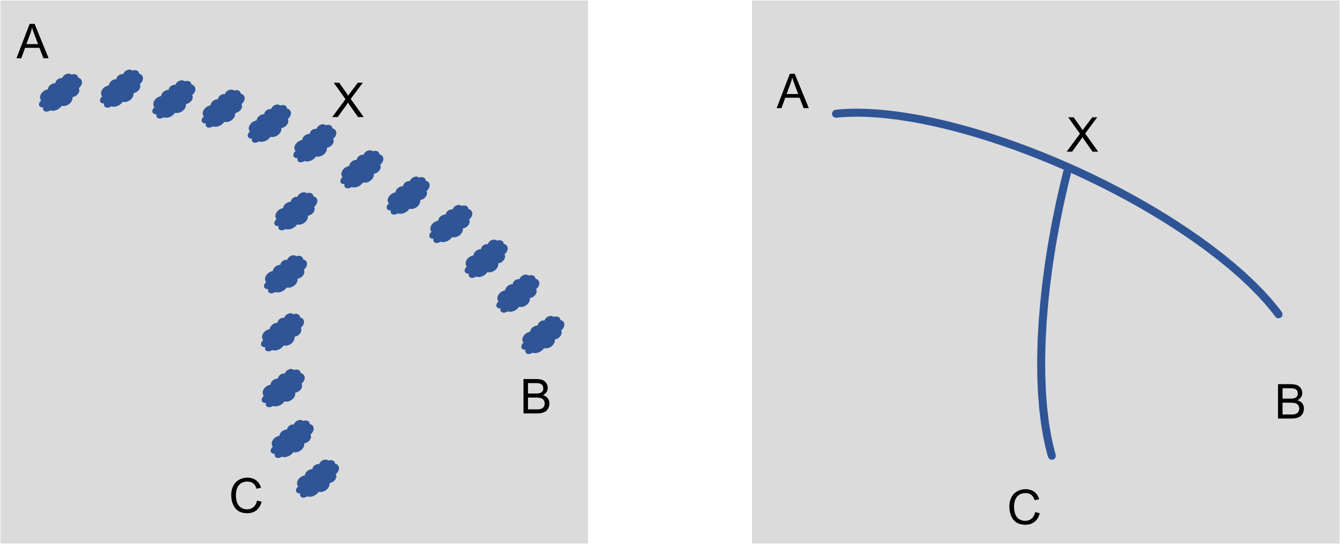
Objects with similar motion or change in appearance are grouped together

Common Region/Connectivity



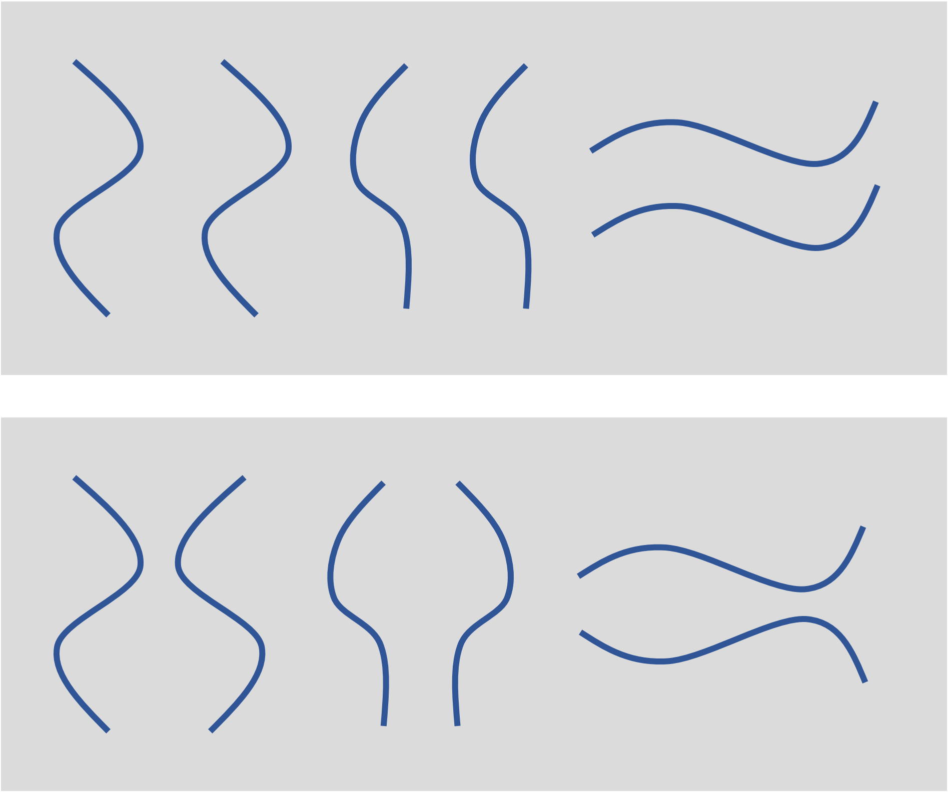
Connected objects are grouped together

Continuity Principle

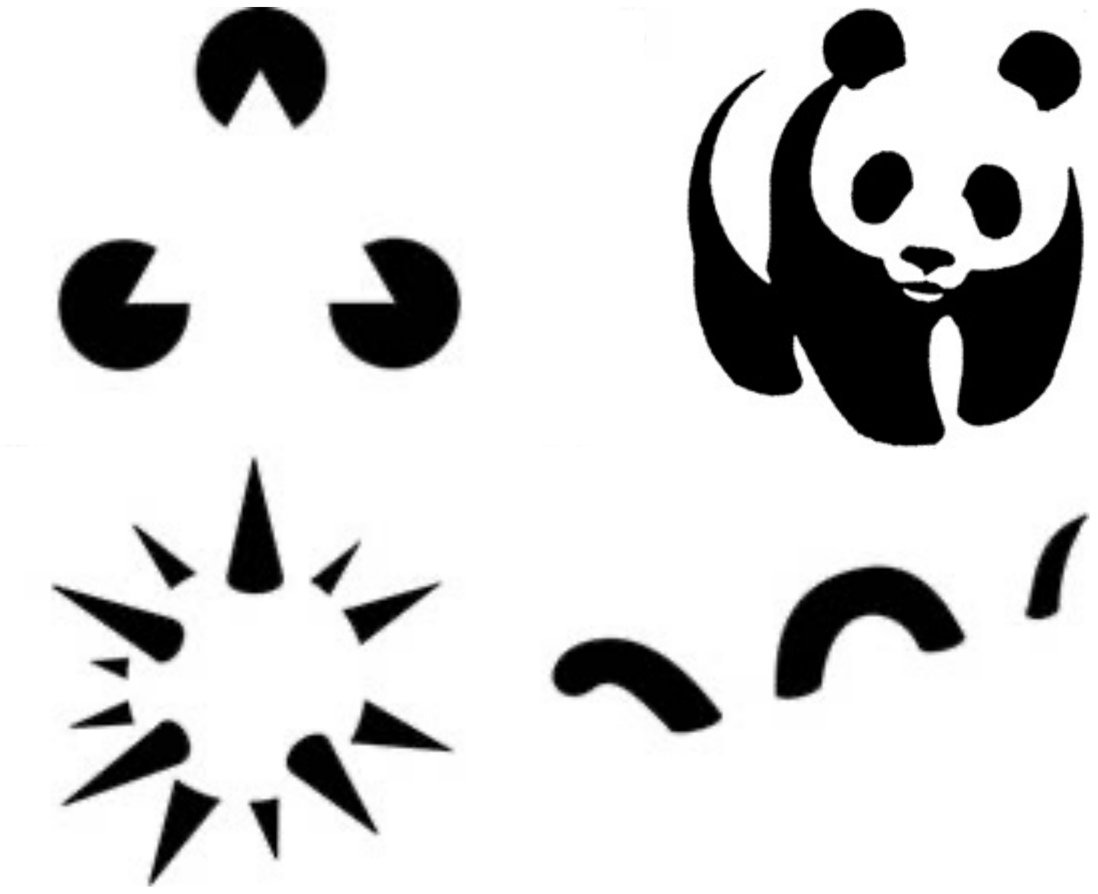


Features on a continuous curve are grouped together

Symmetry Principle

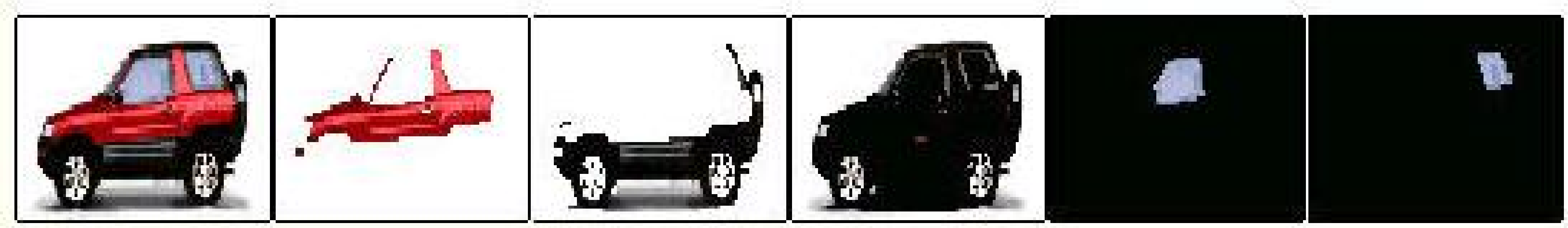


Completion

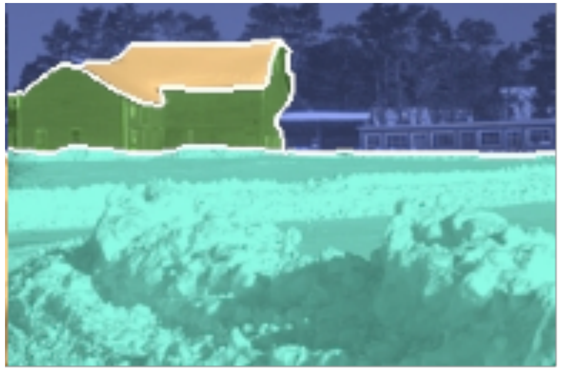


Illusory or subjective contours are perceived

Segmentation/Clustering

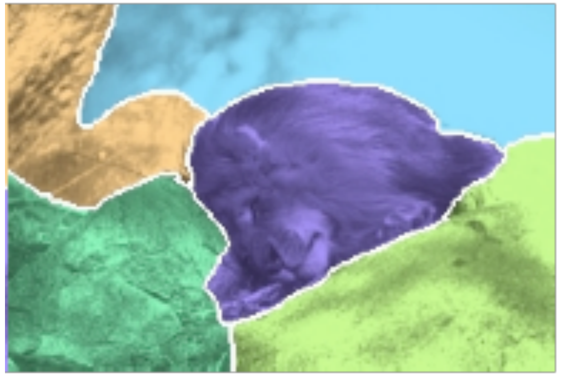


$k = 4$



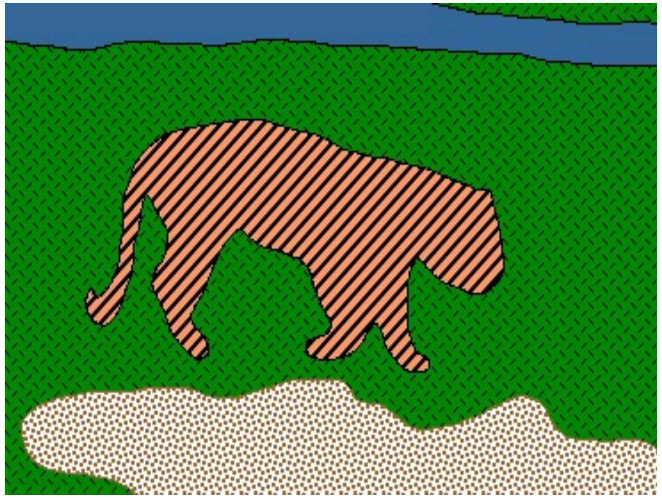
$nc = .0017$

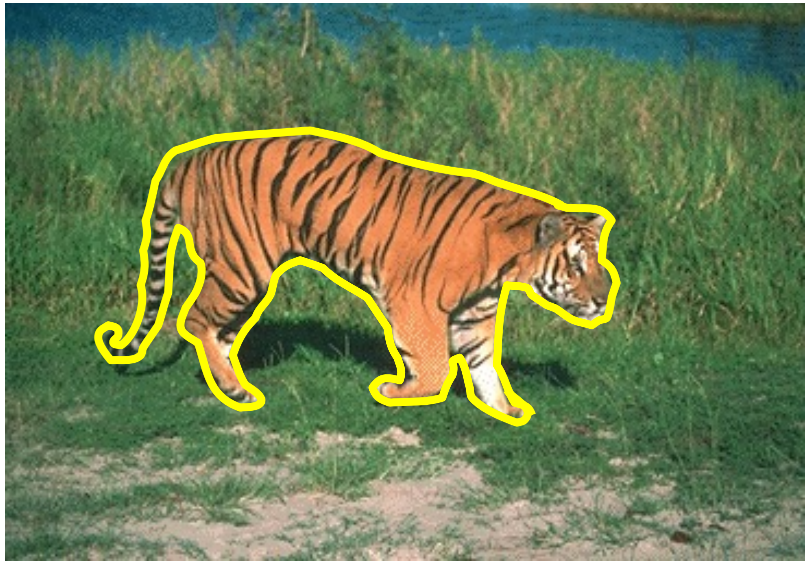
$k = 5$



$nc = .0060$

$k = 11$

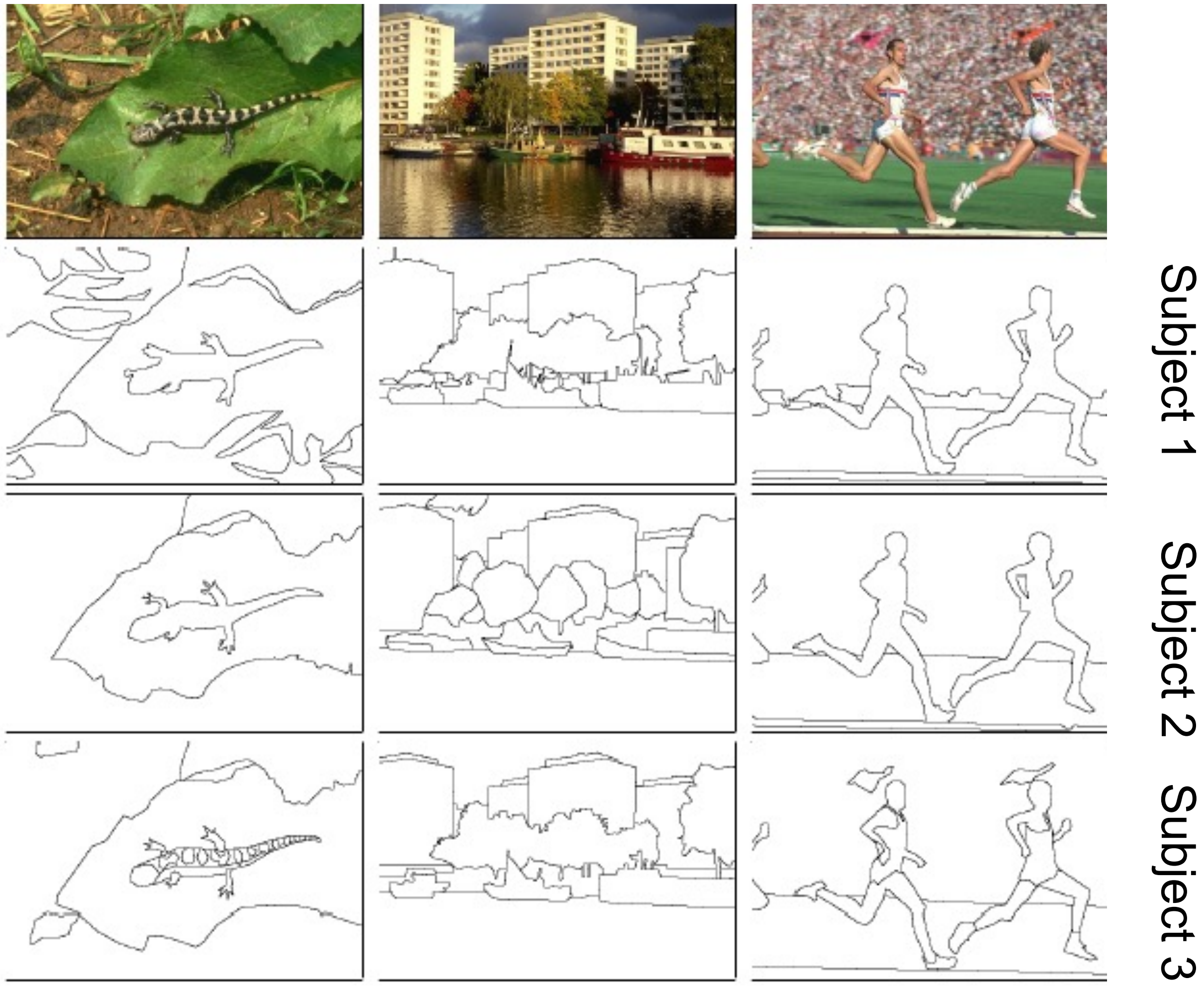





What is a “good” segmentation??

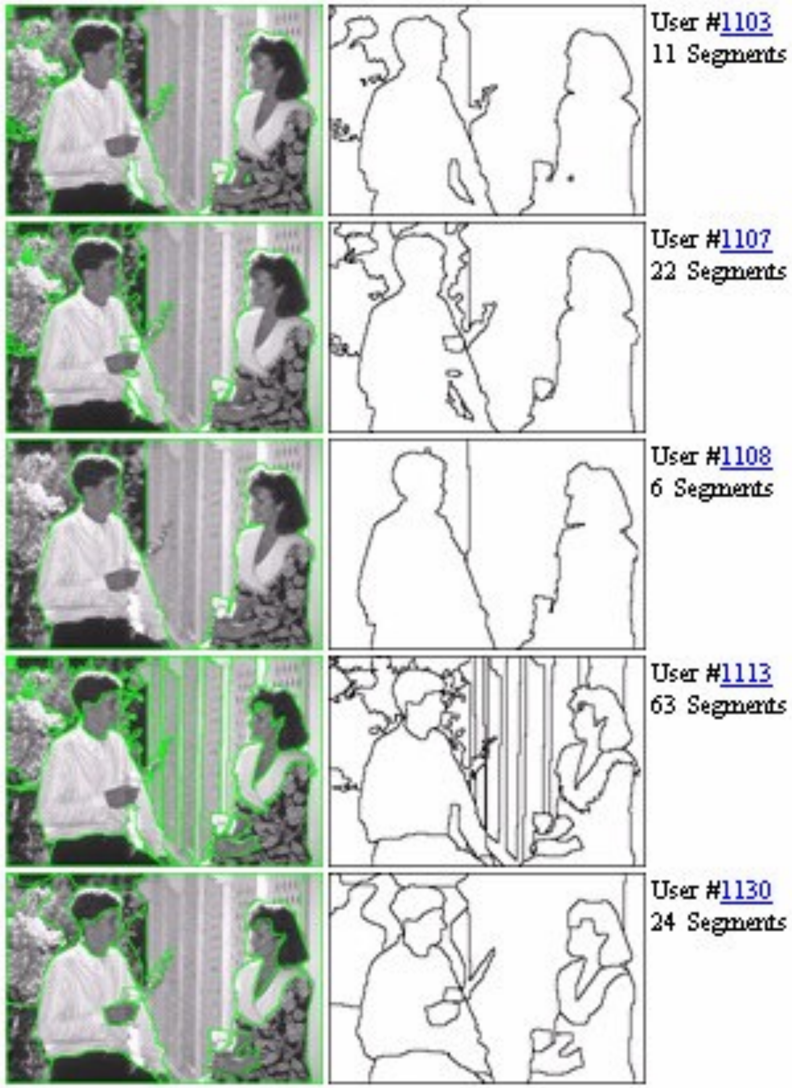
First idea: Compare to human “ground truth”

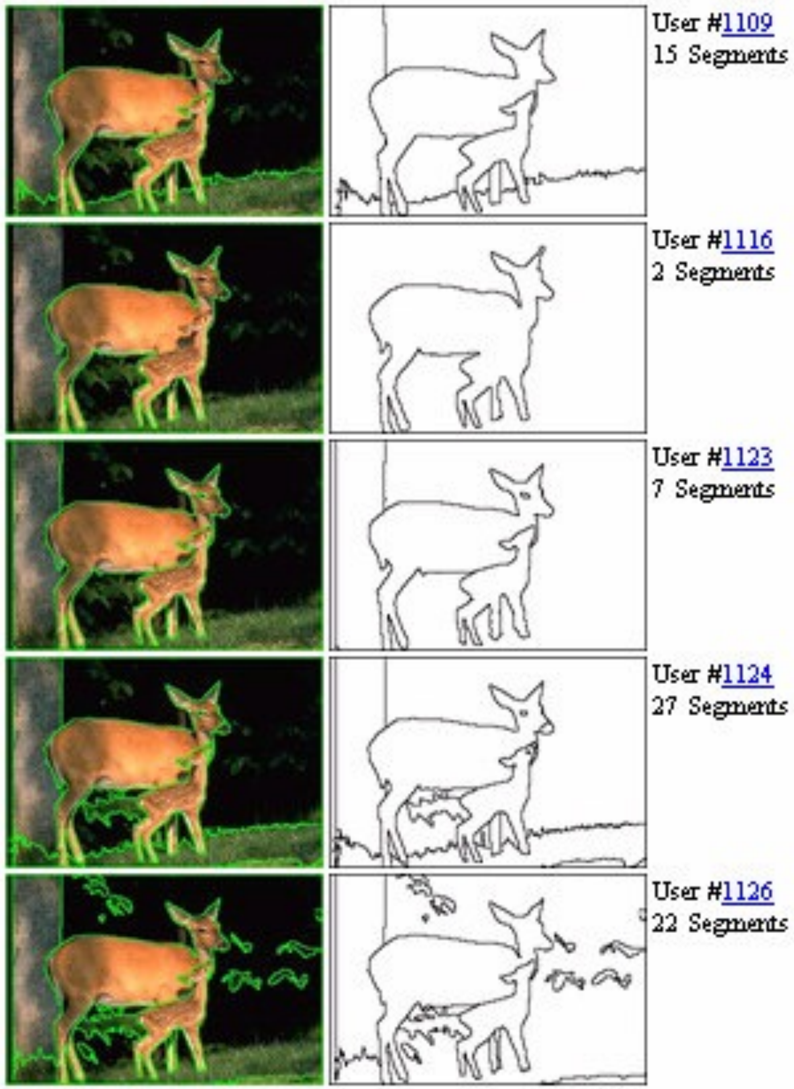
No objective definition of segmentation!



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

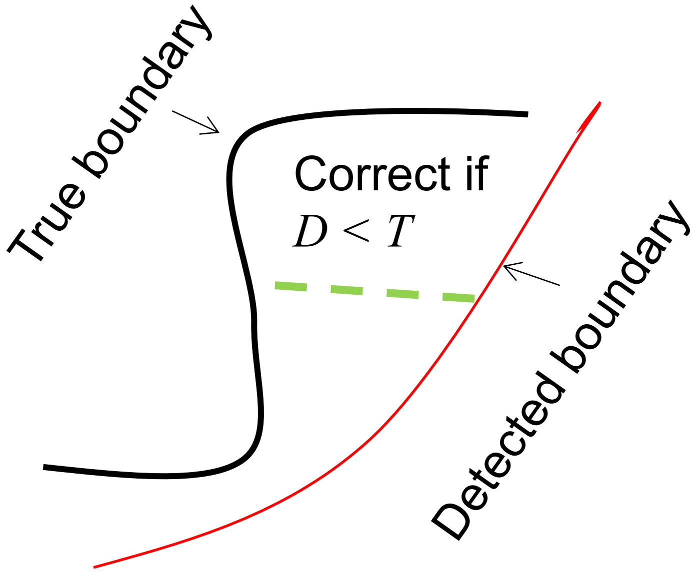
No objective definition of segmentation!

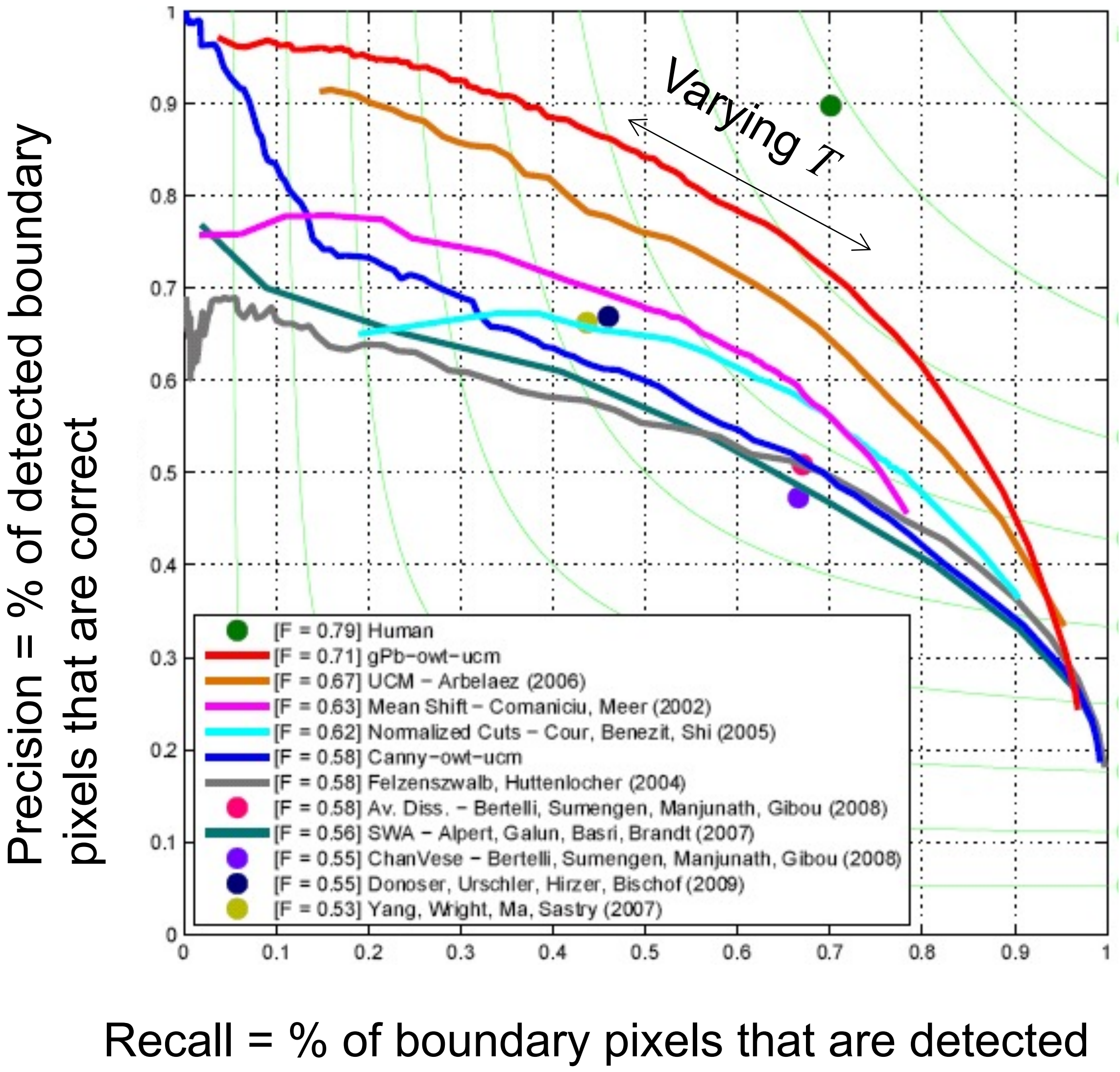




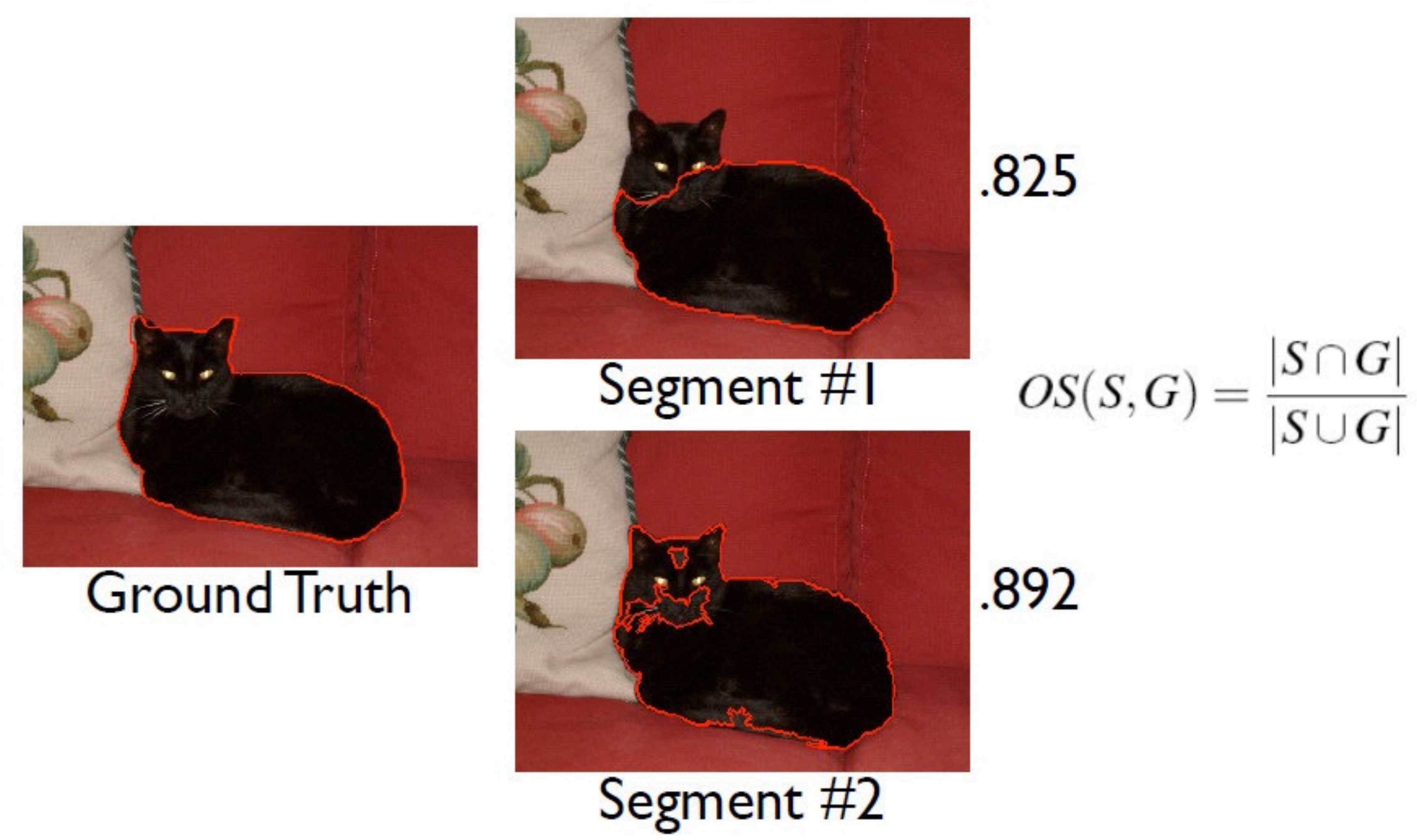
- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/color/317080.html

Evaluation: Boundary agreement

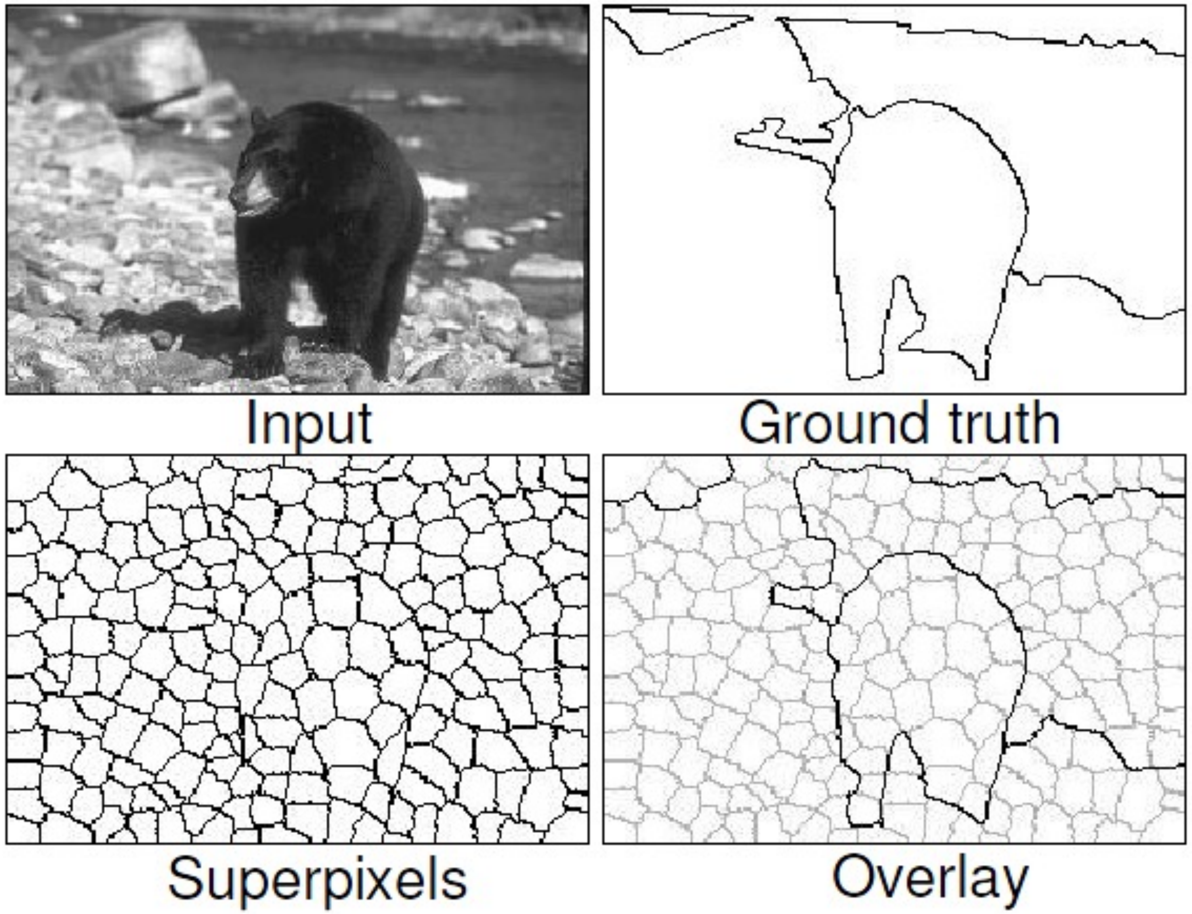


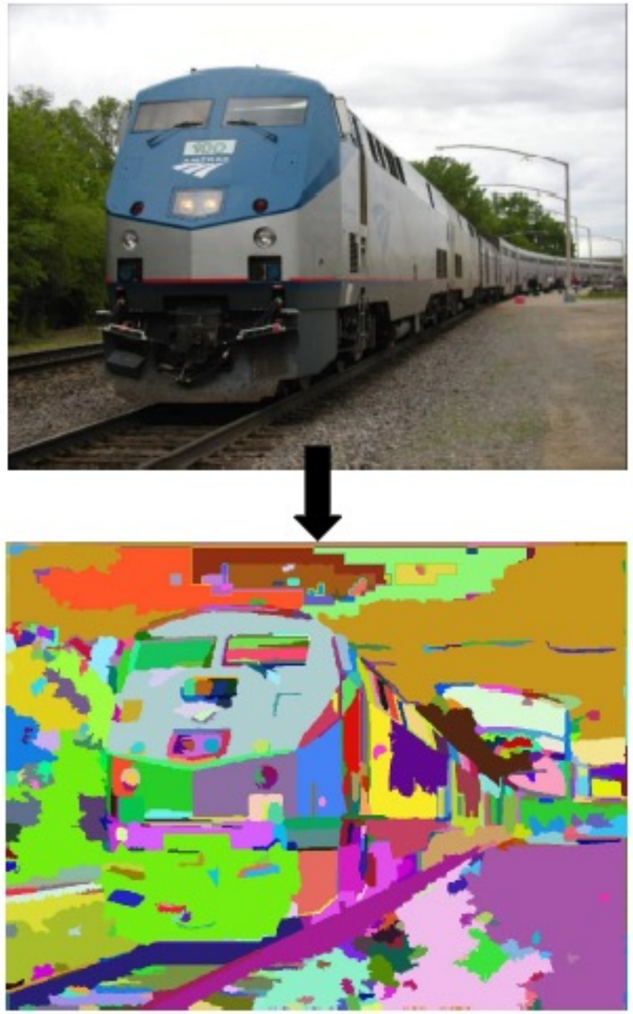


Evaluation: Region overlap with ground truth



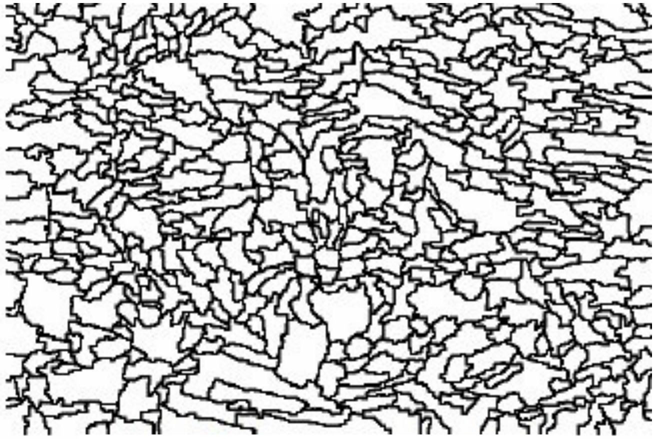
Second idea: Superpixels



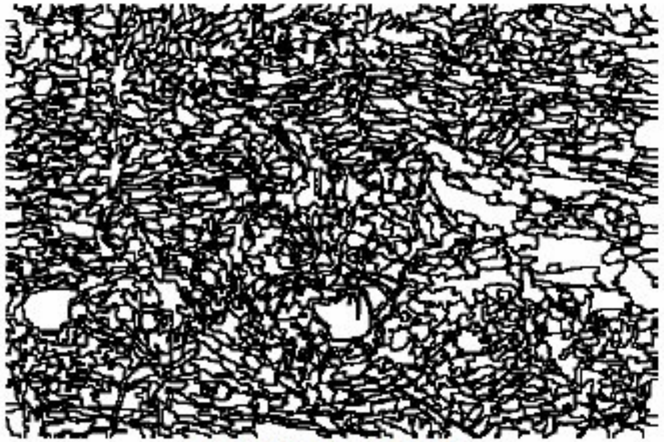


- Let’s not even try to compute a “correct” segmentation
- Let’s be content with an *oversegmentation* in which each region is very likely (formal guarantees are hard) to be uniform

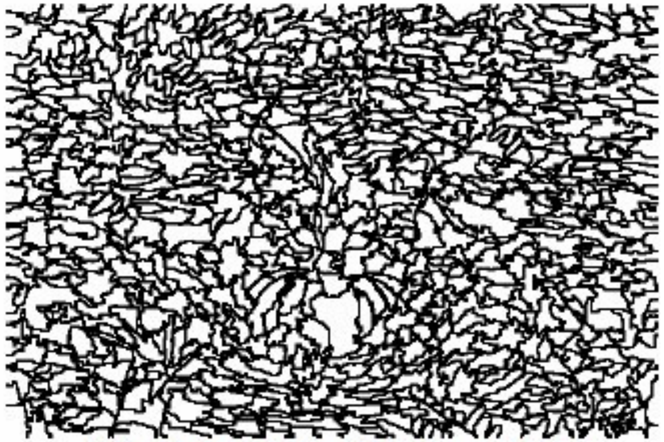
Second idea: Superpixels



Watershed



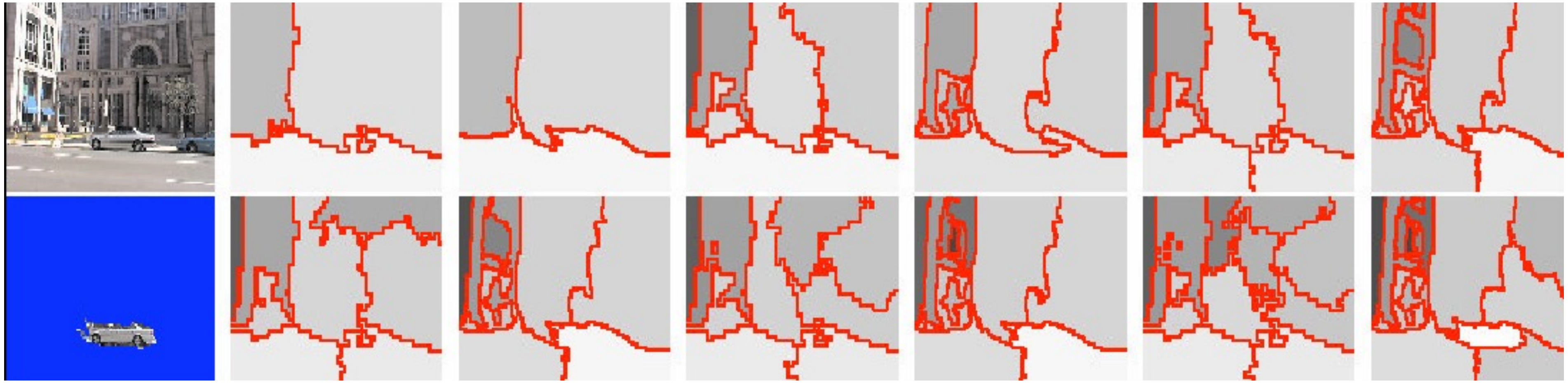
Mean shift



Graph-based

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

Third idea: Multiple segmentations



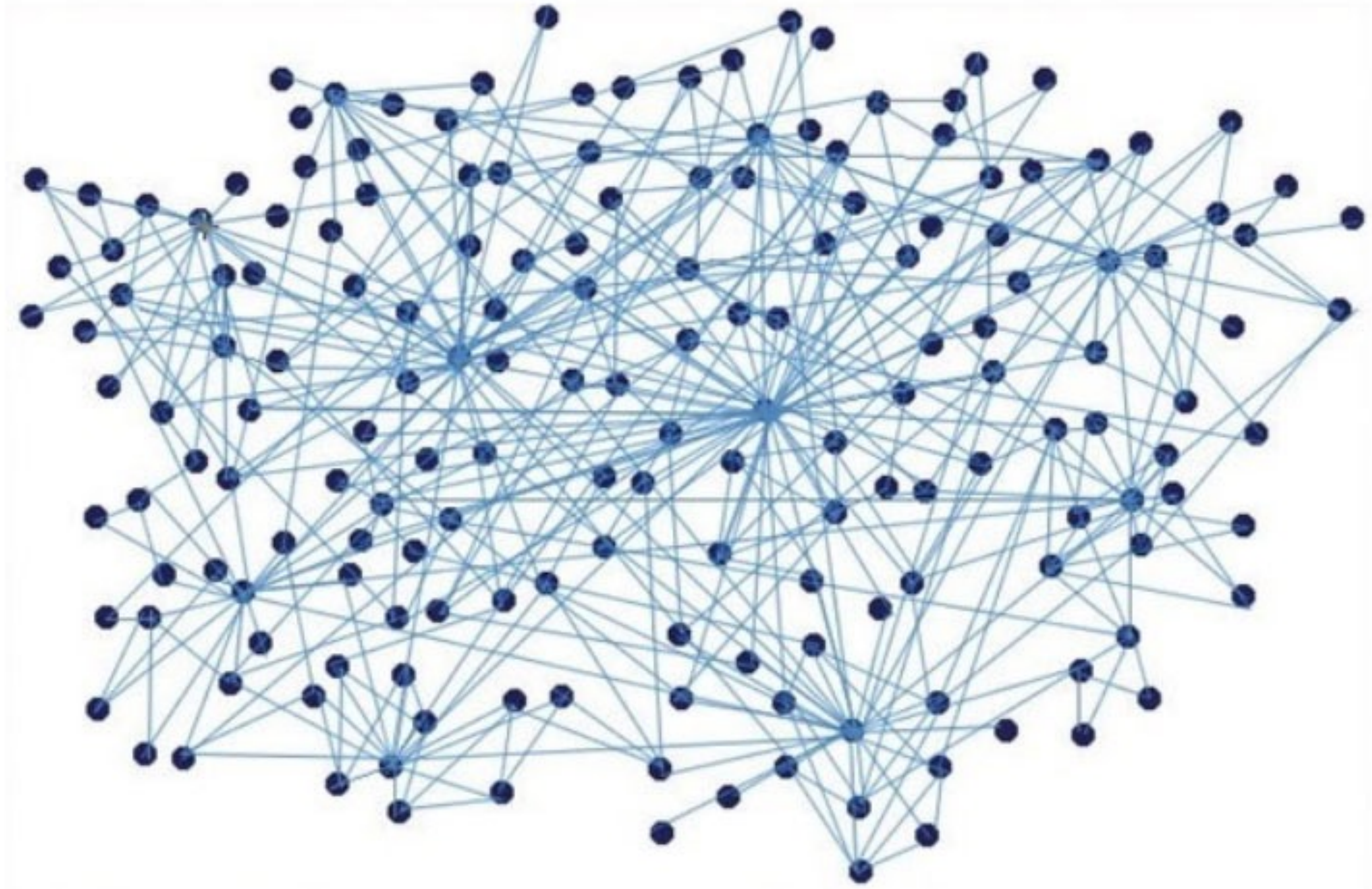
- Generate many segmentations of the same image
- Even though many regions are “wrong”, some consensus should emerge

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

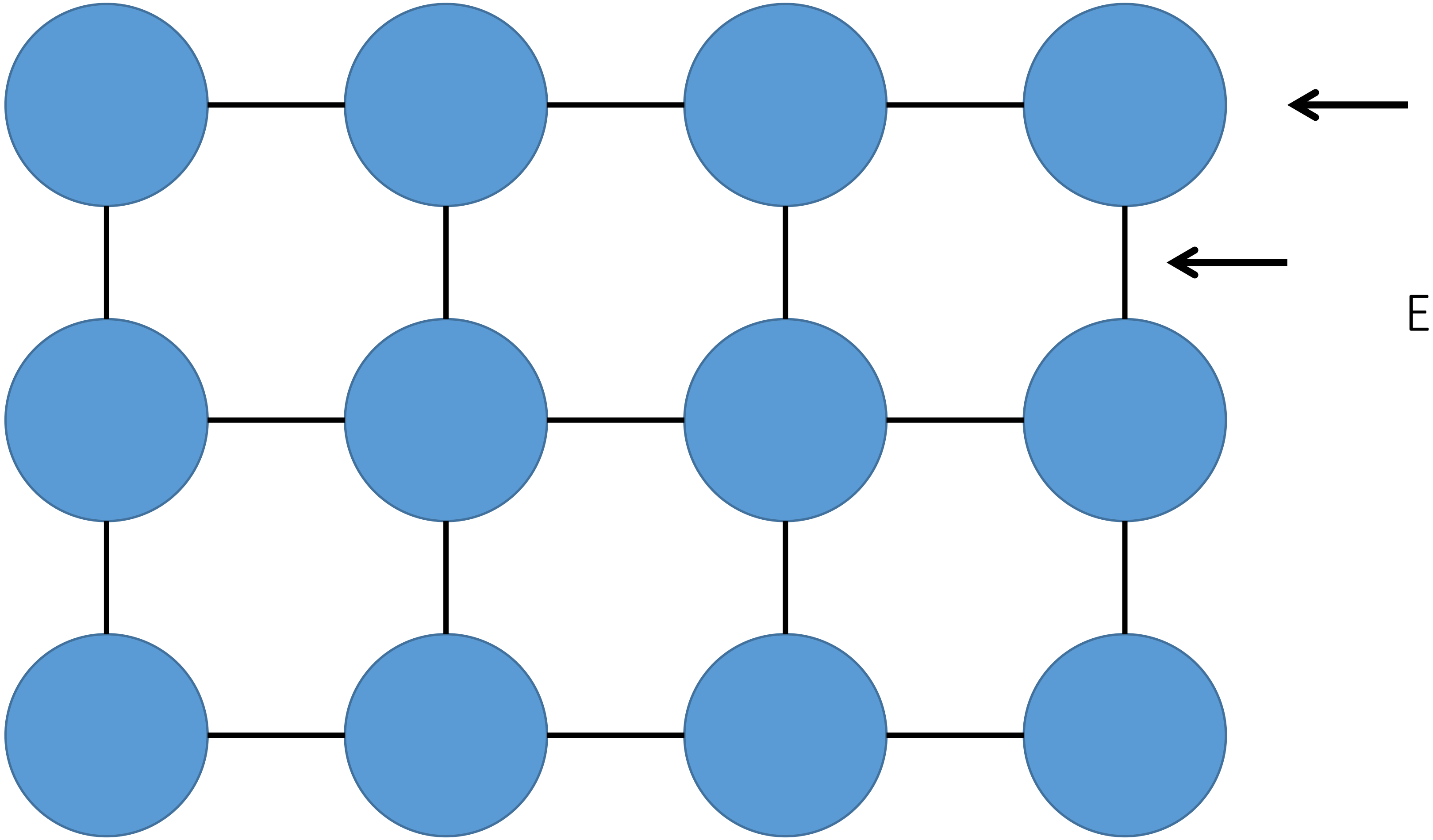
Main approaches

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

Image as a graph



Images can be viewed as graphs

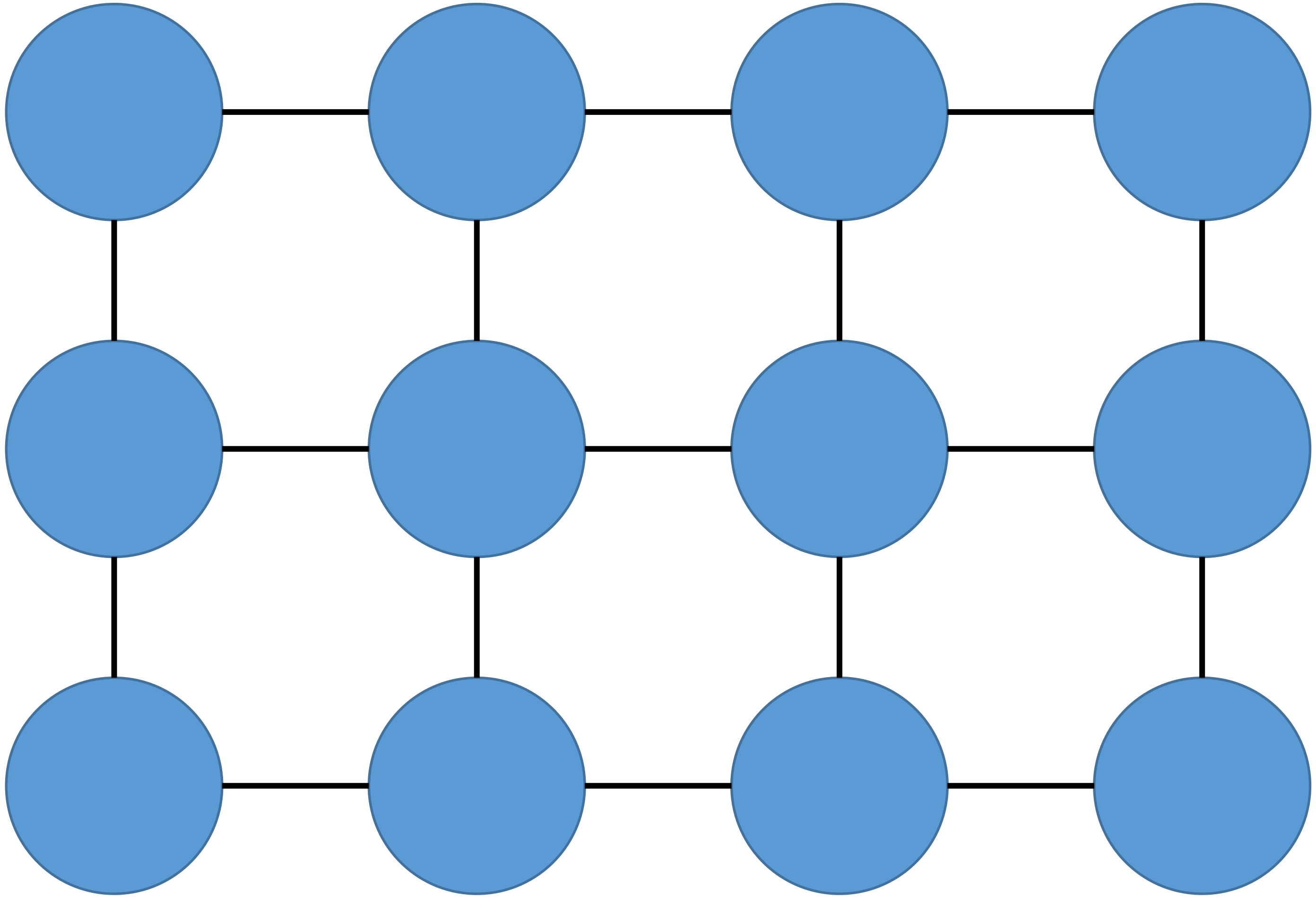


Nodes: pixels

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

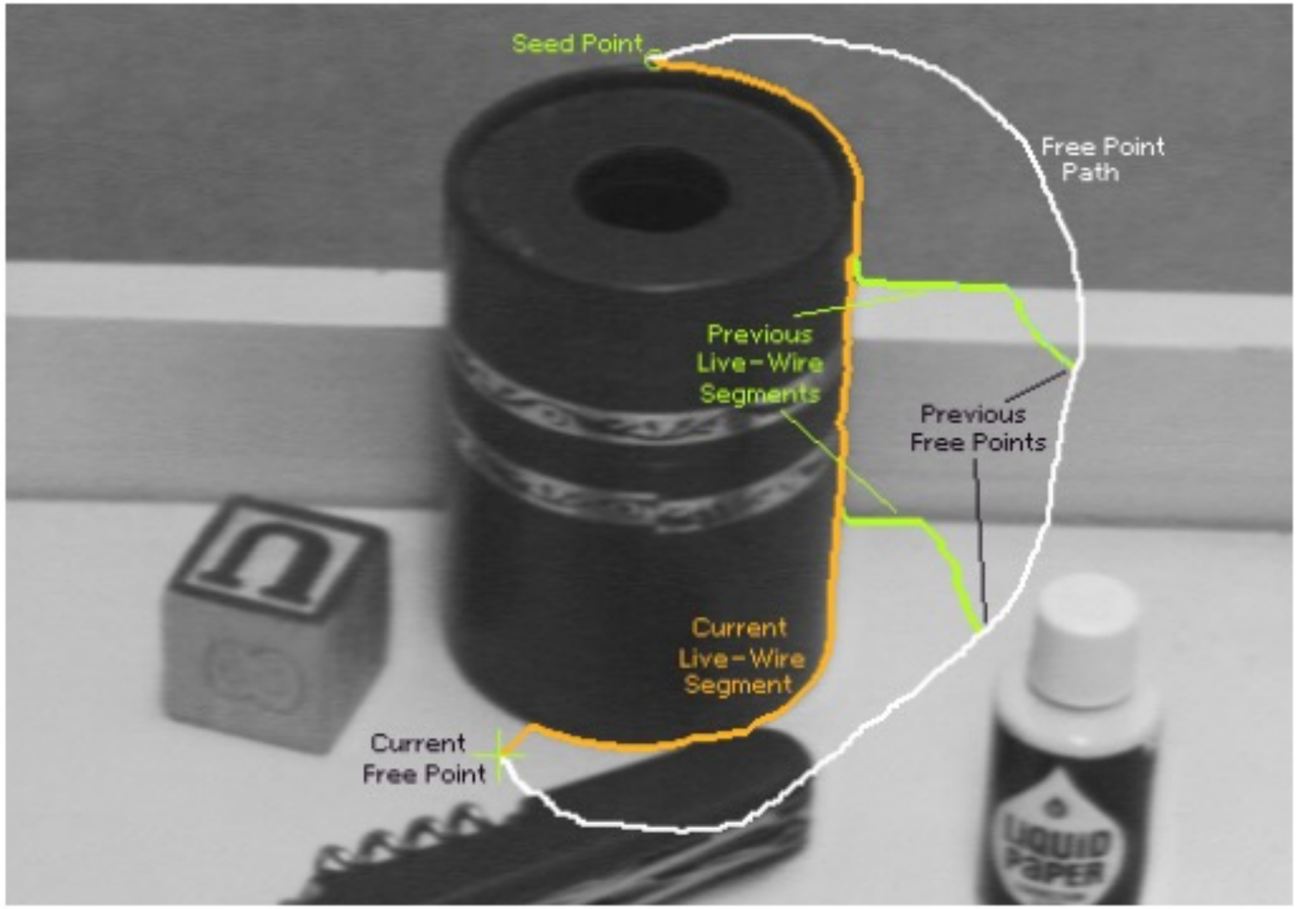
Intelligent scissors

Problem statement:

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

Challenges:

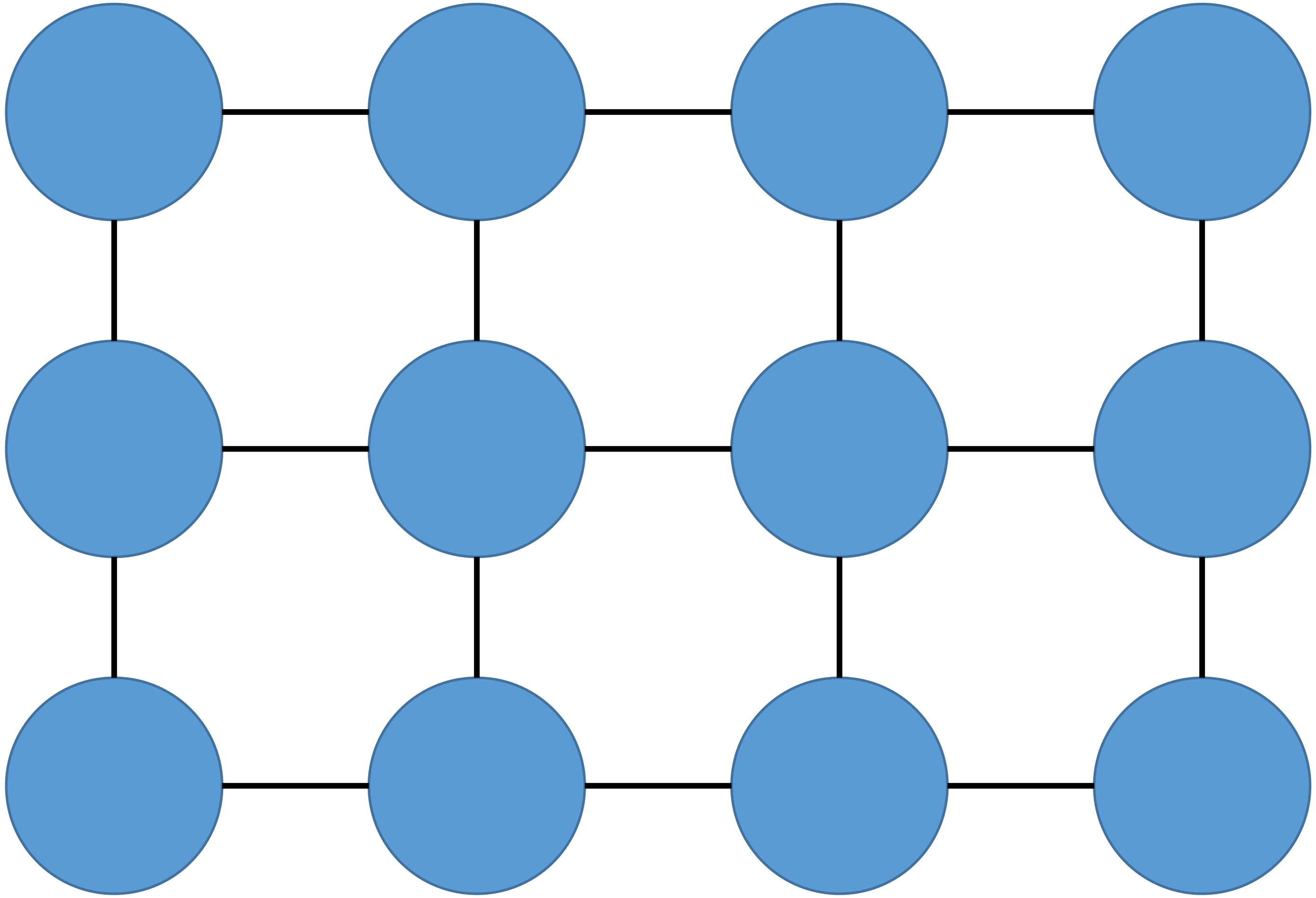
- Make this real-time for interaction
- Define what makes a good boundary



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

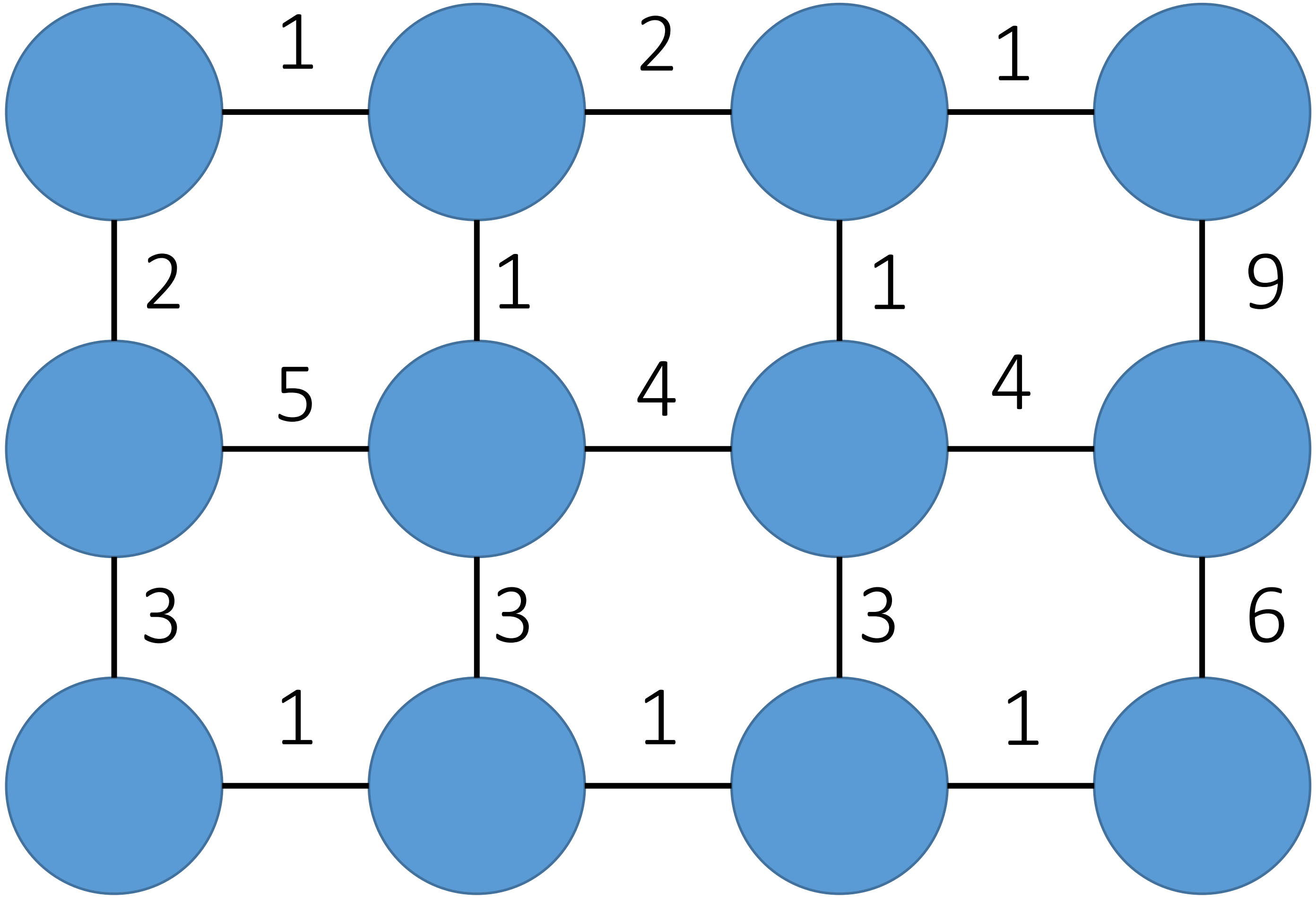


Nodes: pixels

Edges: Constraints between neighboring pixels

Graph-view of this problem

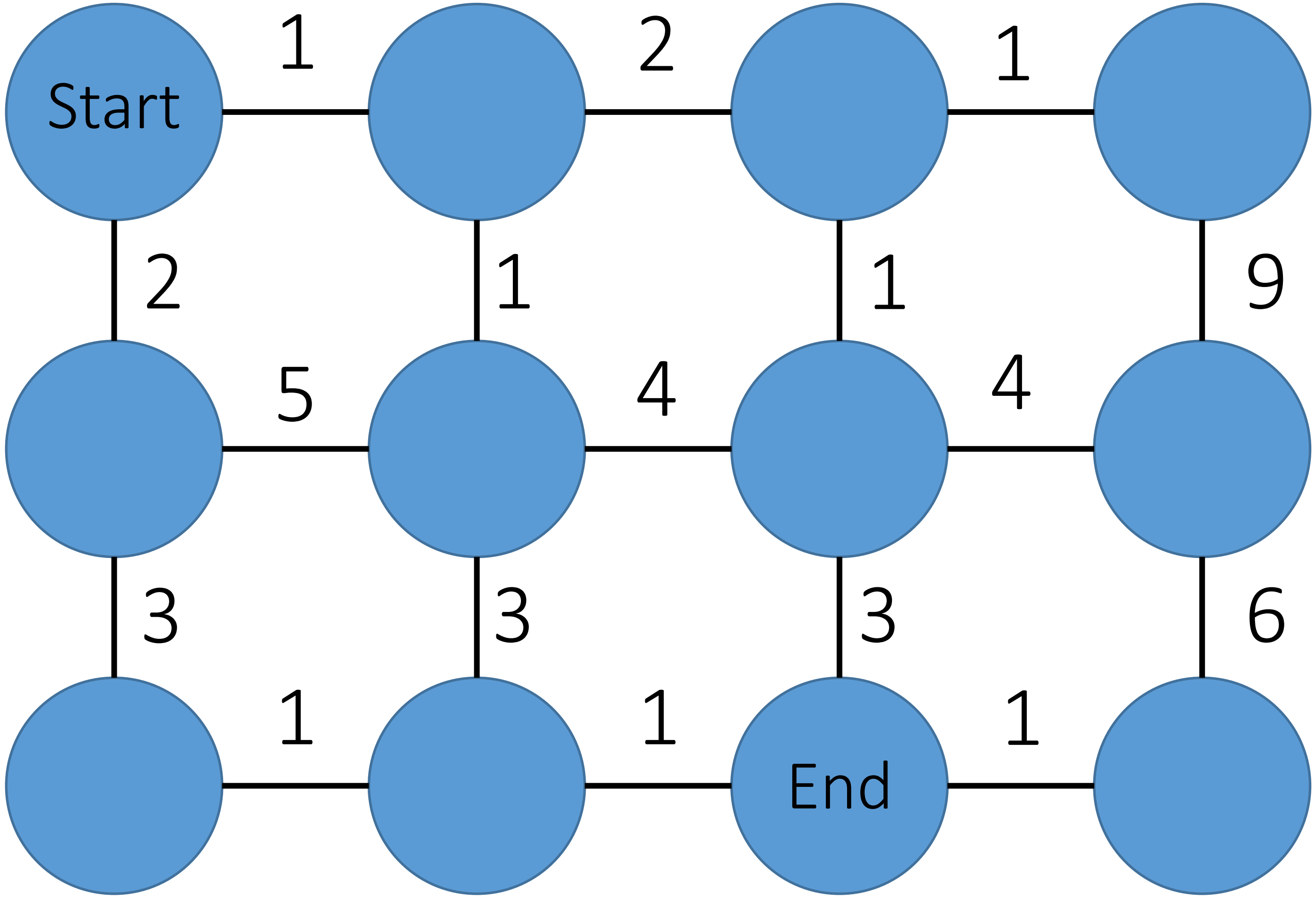
Graph-view of intelligent scissors:



1. Assign weights (costs) to edges

Graph-view of this problem

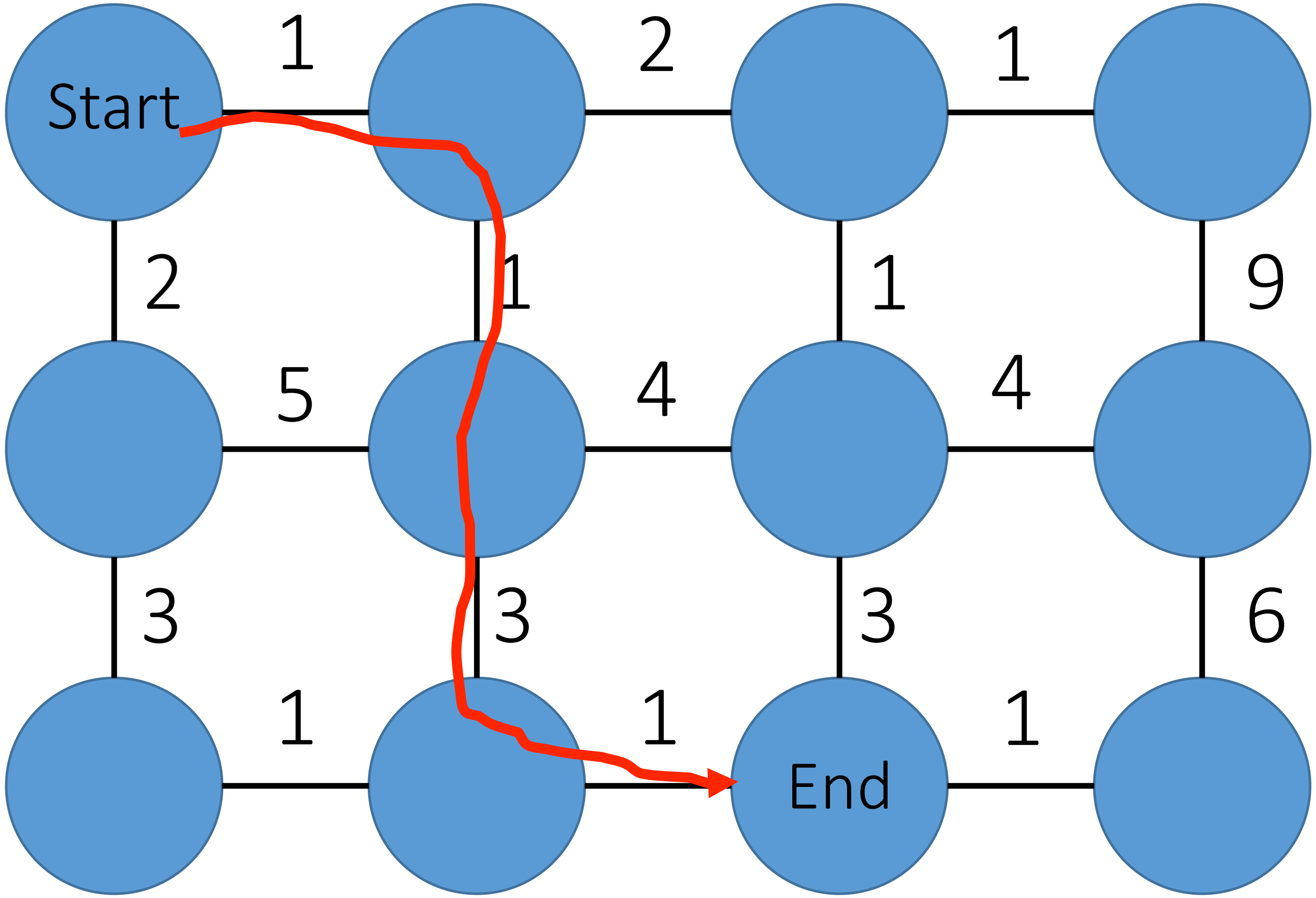
Graph-view of intelligent scissors:



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

Graph-view of this problem

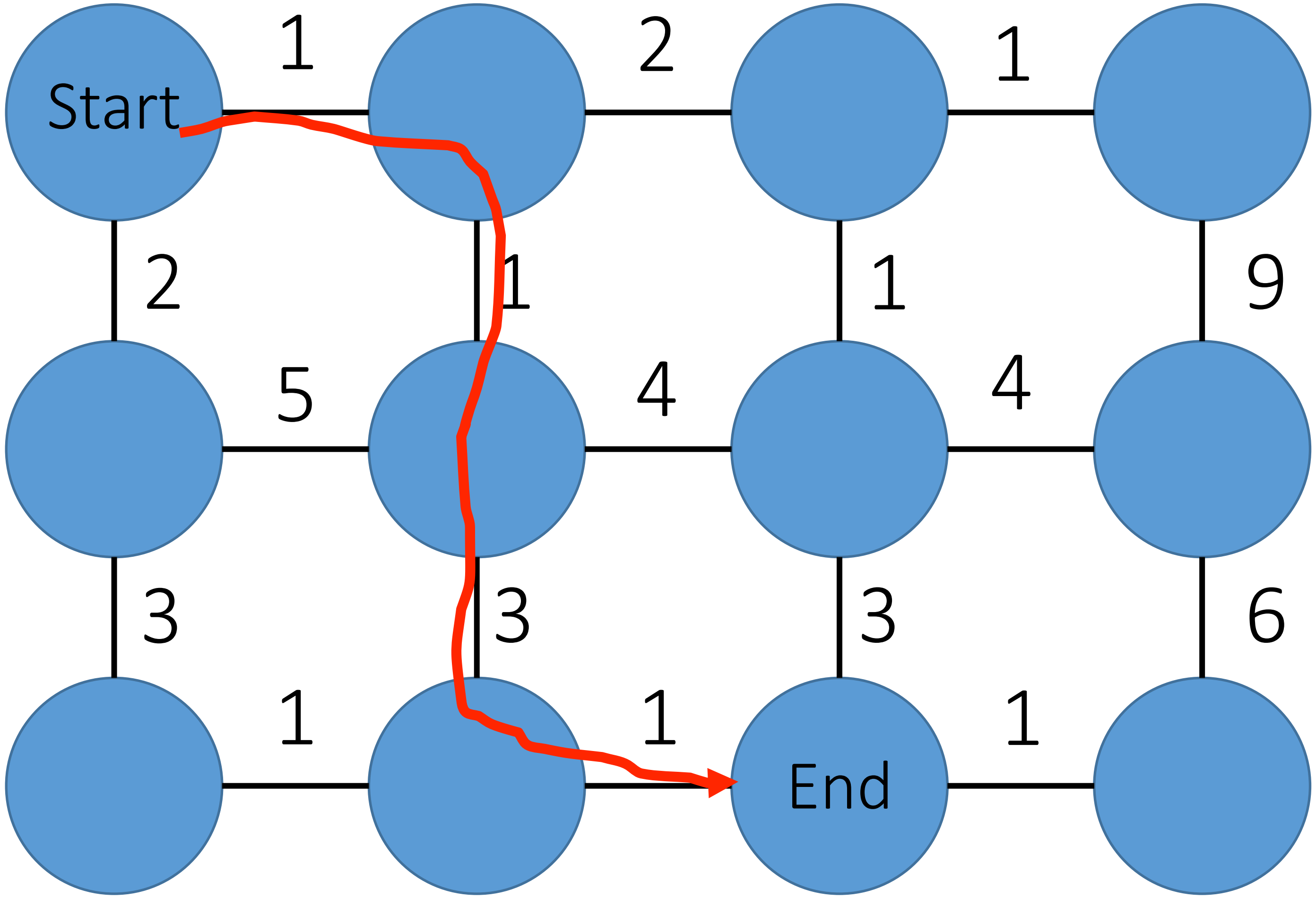
Graph-view of intelligent scissors:



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:

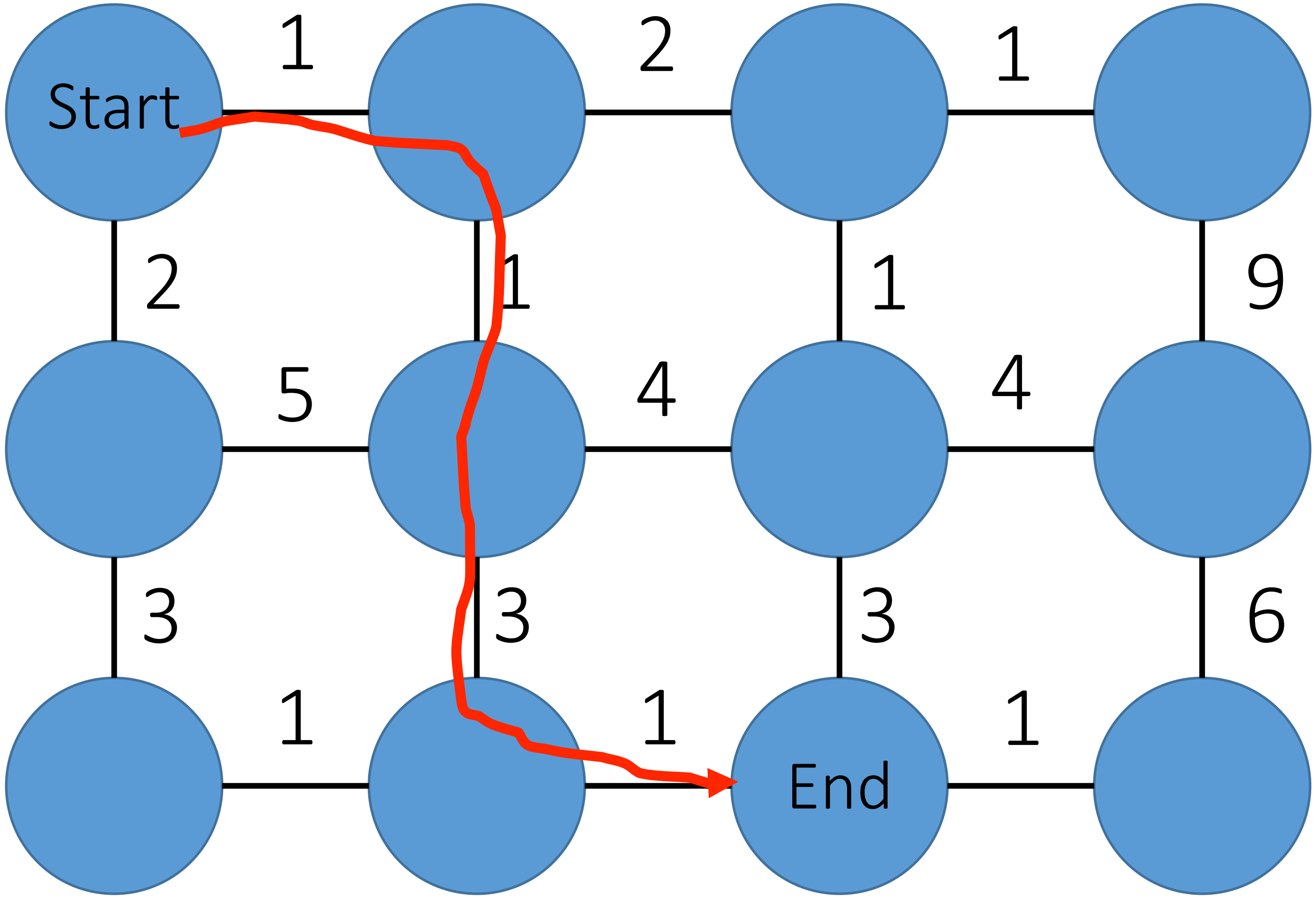


1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

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 $\text{cost}_2(q, r)$ % 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**

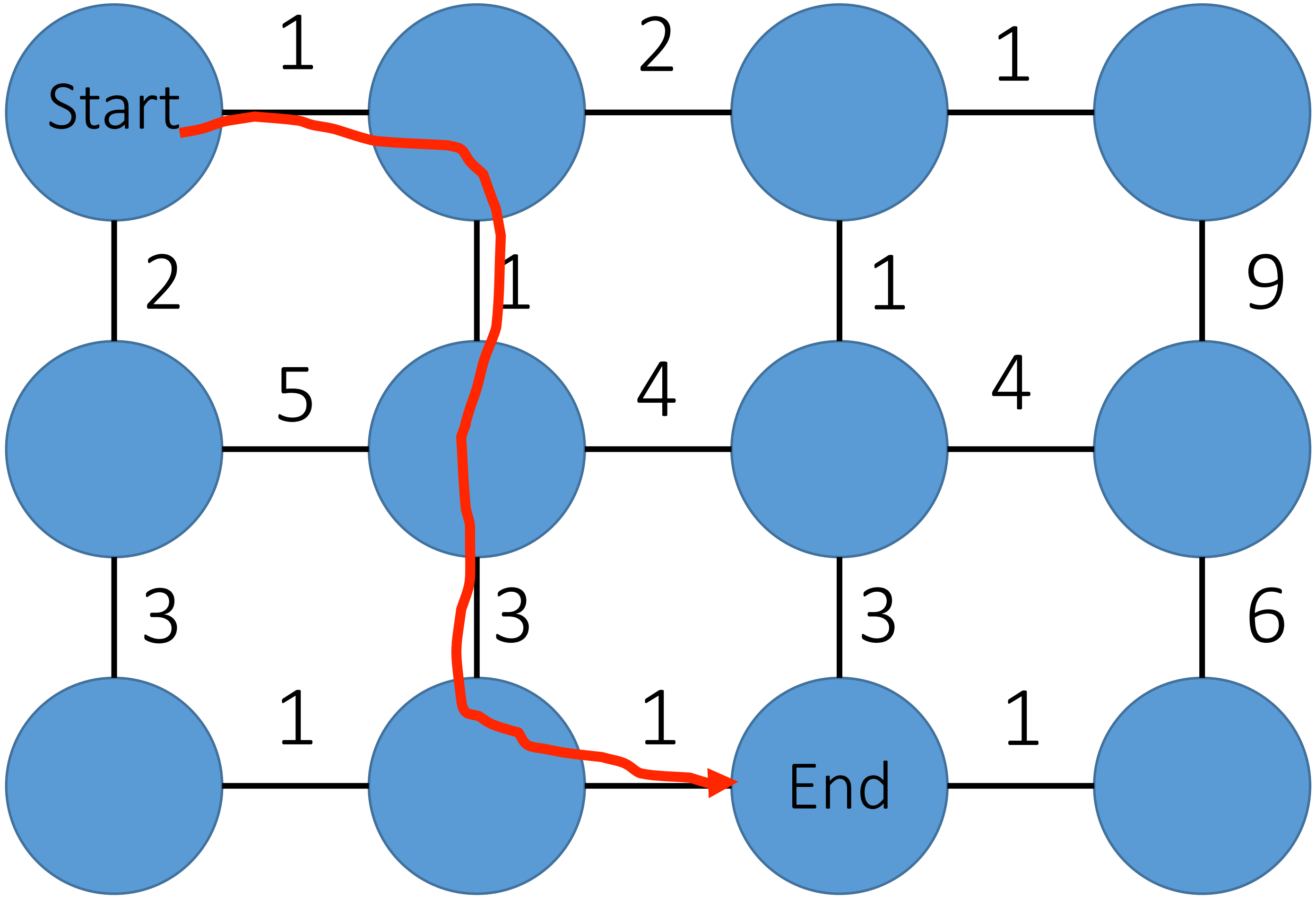
3.For each pixel *r* in neighborhood of *q* that is in **A**

a) cost_tmp = cost(*q*) + $\text{cost}_2(q,r)$ %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

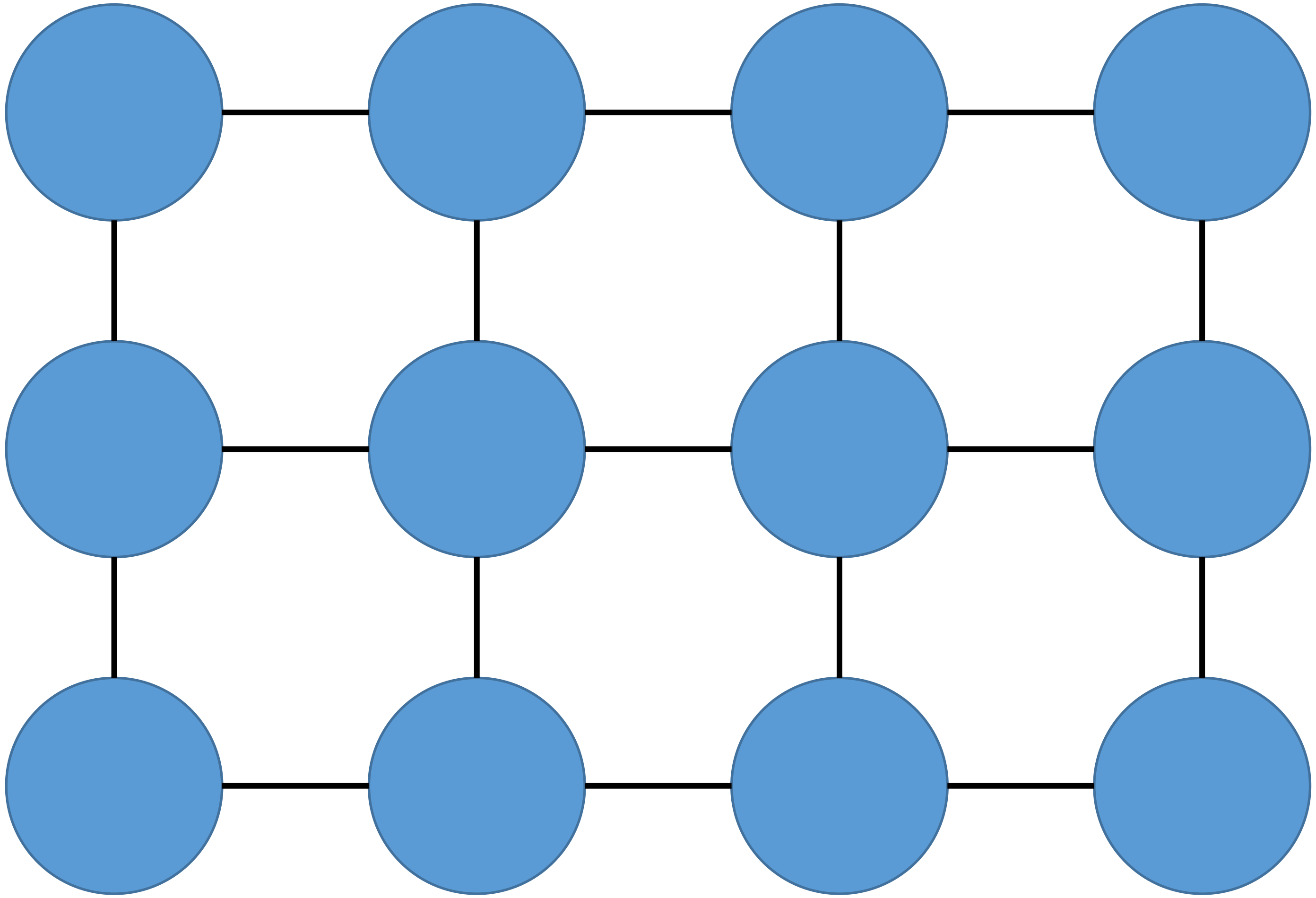
Gradient magnitude

Edge image

Pixel-wise cost

Segmentation using graph cuts

Remember: Graph-based view of images



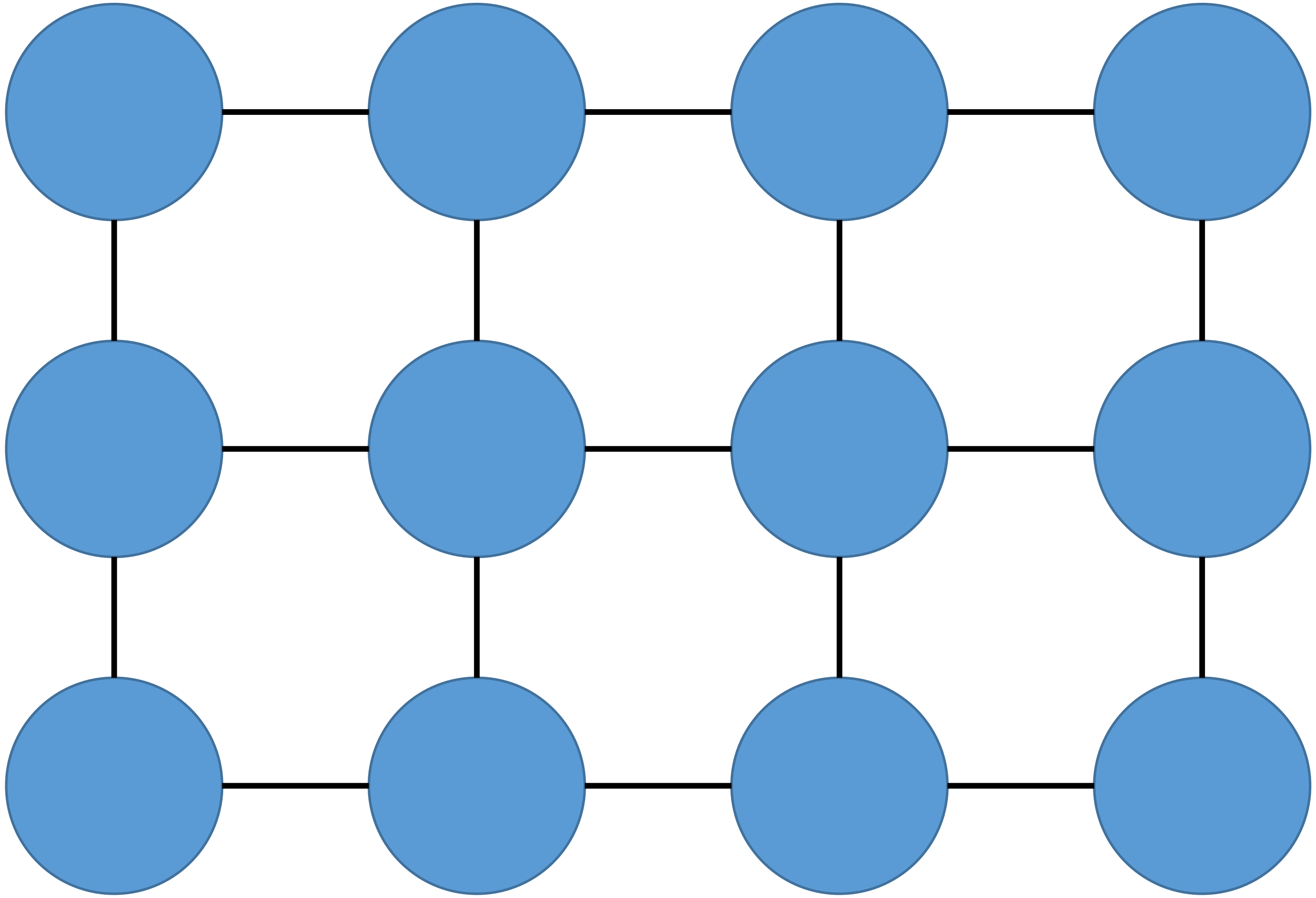
Nodes: pixels

Edges: Constraints between neighboring pixels

Markov Random Field (MRF)

Assign foreground/background labels based on:

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

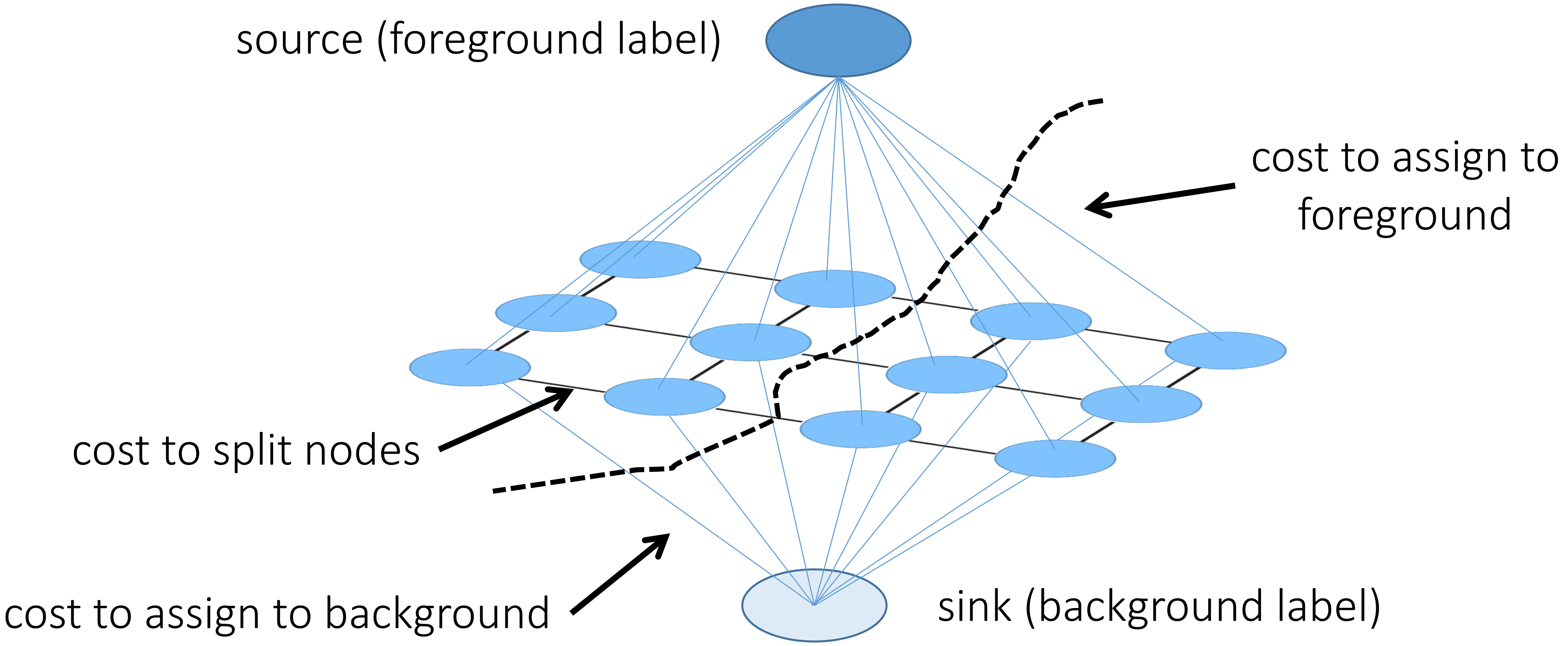


Given its intensity value, how likely is a pixel to be foreground or background?

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

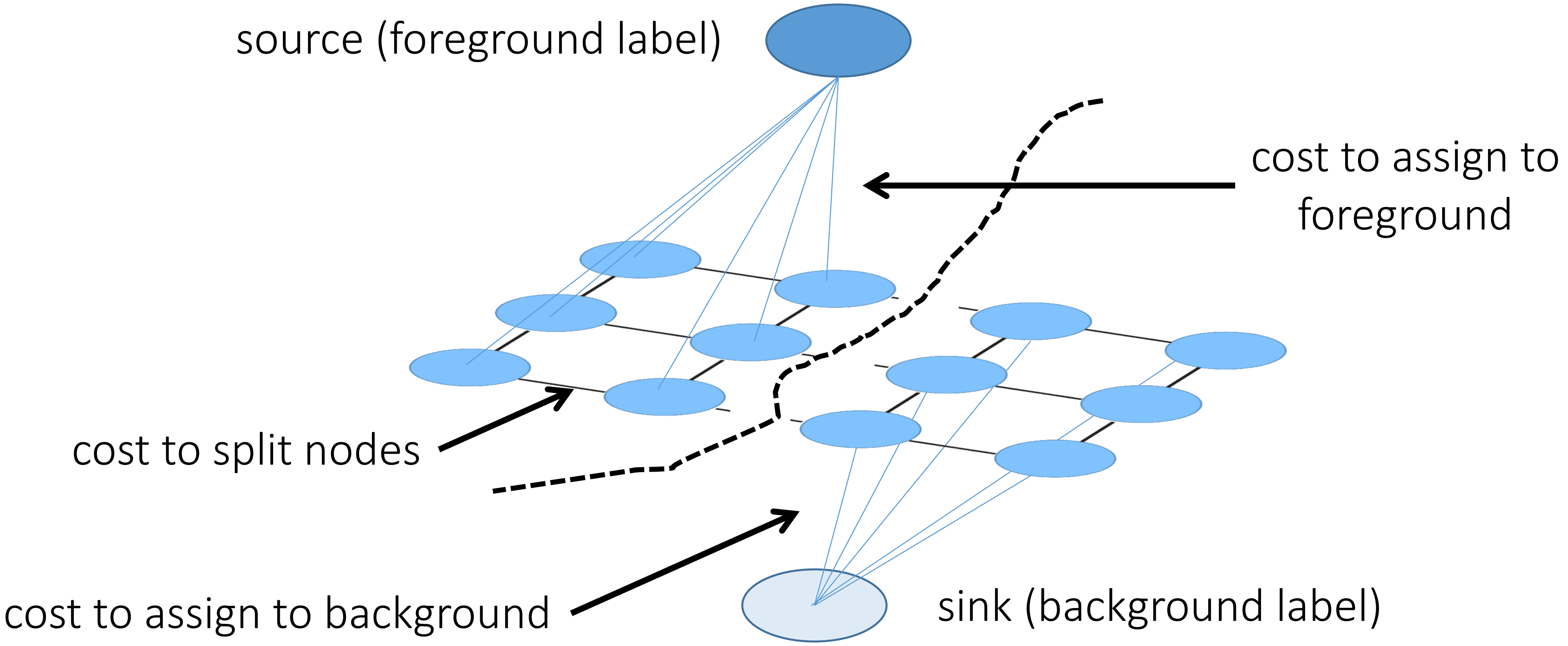
What kind of cost functions would you use for GraphCut?

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)



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

Graph-cuts segmentation

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

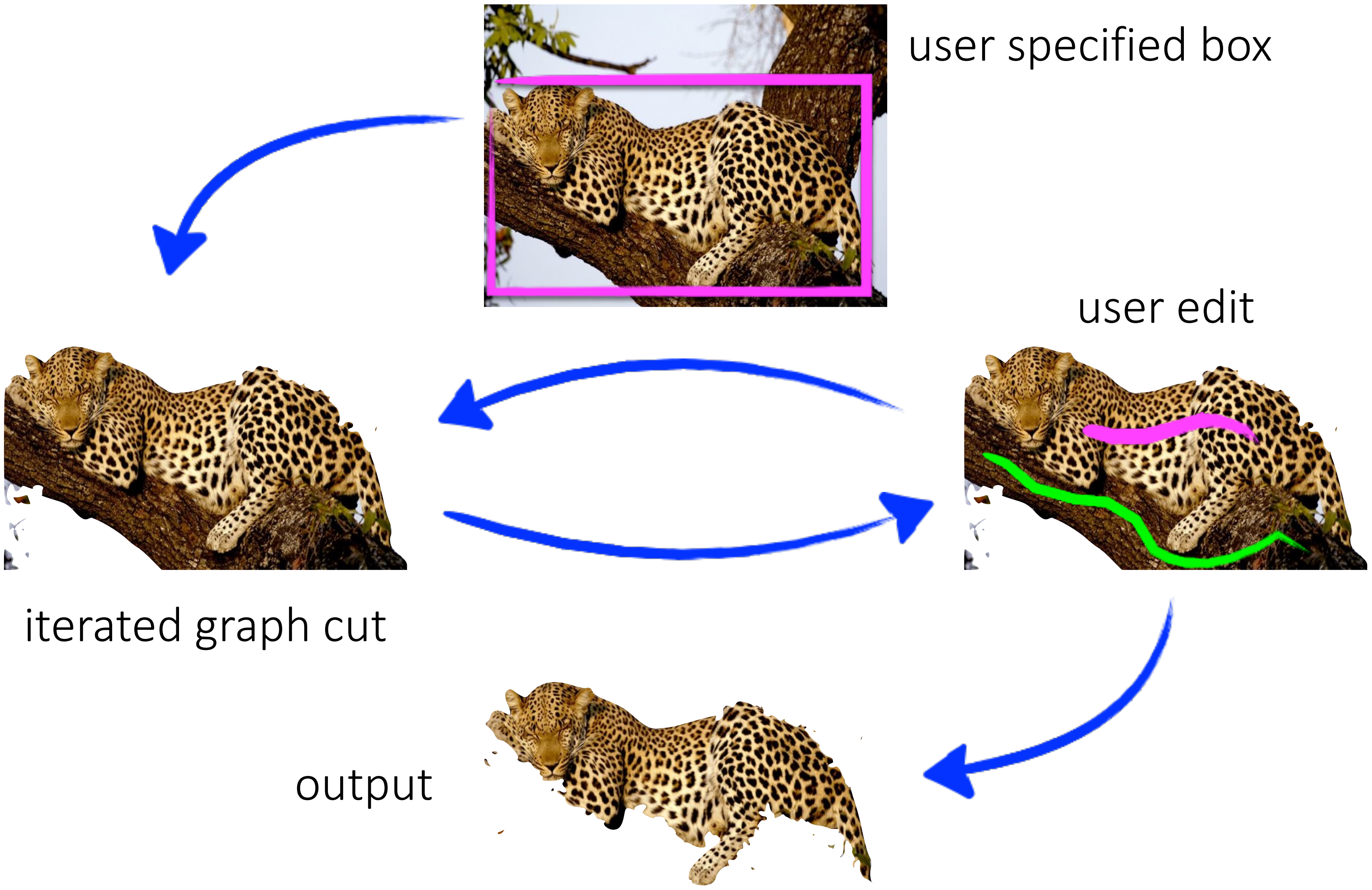
$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Set weights for edges between pixels

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

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

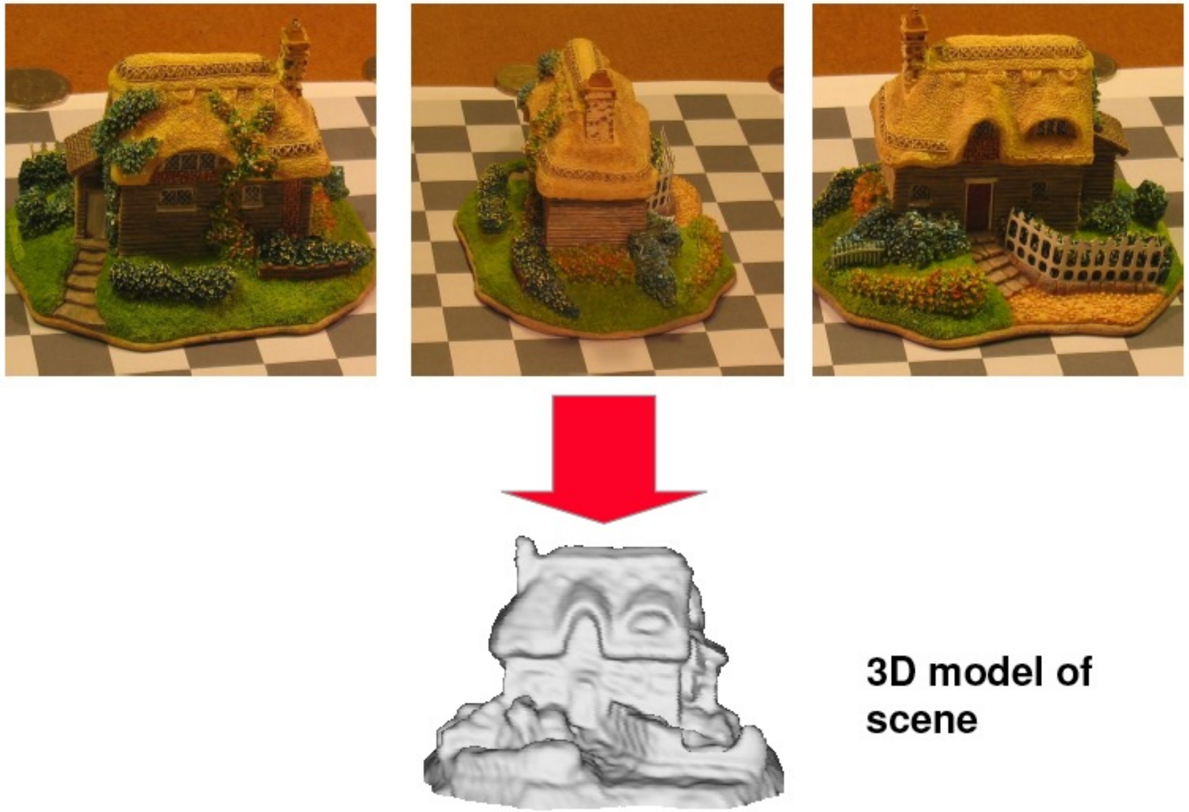
Iteration can be interactive



Examples

Graph-cuts are a very general, very useful tool

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



3D model of scene

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering