Looking for seams

15-463, 15-663, 15-862
Computational Photography
Fall 2017, Lecture 8

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

Course announcements

- Apologies for canceling Monday’s lecture.

- Homework 3 will be posted tonight.
  - Due October 12th.

- Comments on Homework 2?

Overview of today's lecture

- Back to cutting-and-pasting (and other motivating examples).
- Image as a graph.
- Shortest graph paths and Intelligent scissors.
- Graph-cuts and GrabCut.
- Some notes about cutting-and-pasting.

Slide credits

Most of these slides were adapted from:

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

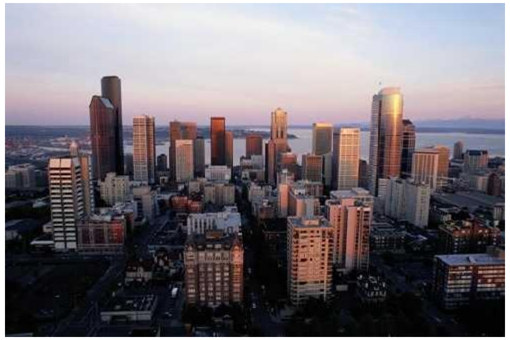
Some slides were inspired or taken from:

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

Back to cutting-and-pasting (and other motivating examples)

Cut and paste procedure

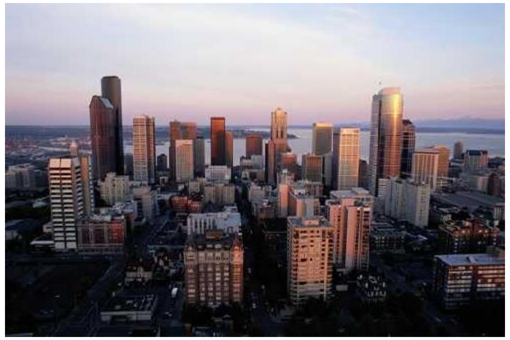
1. Extract Sprites



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:

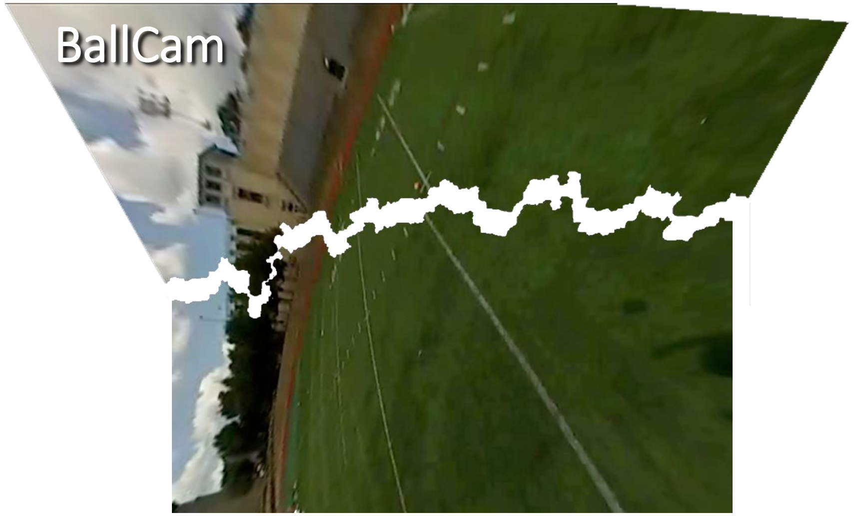


image stitching

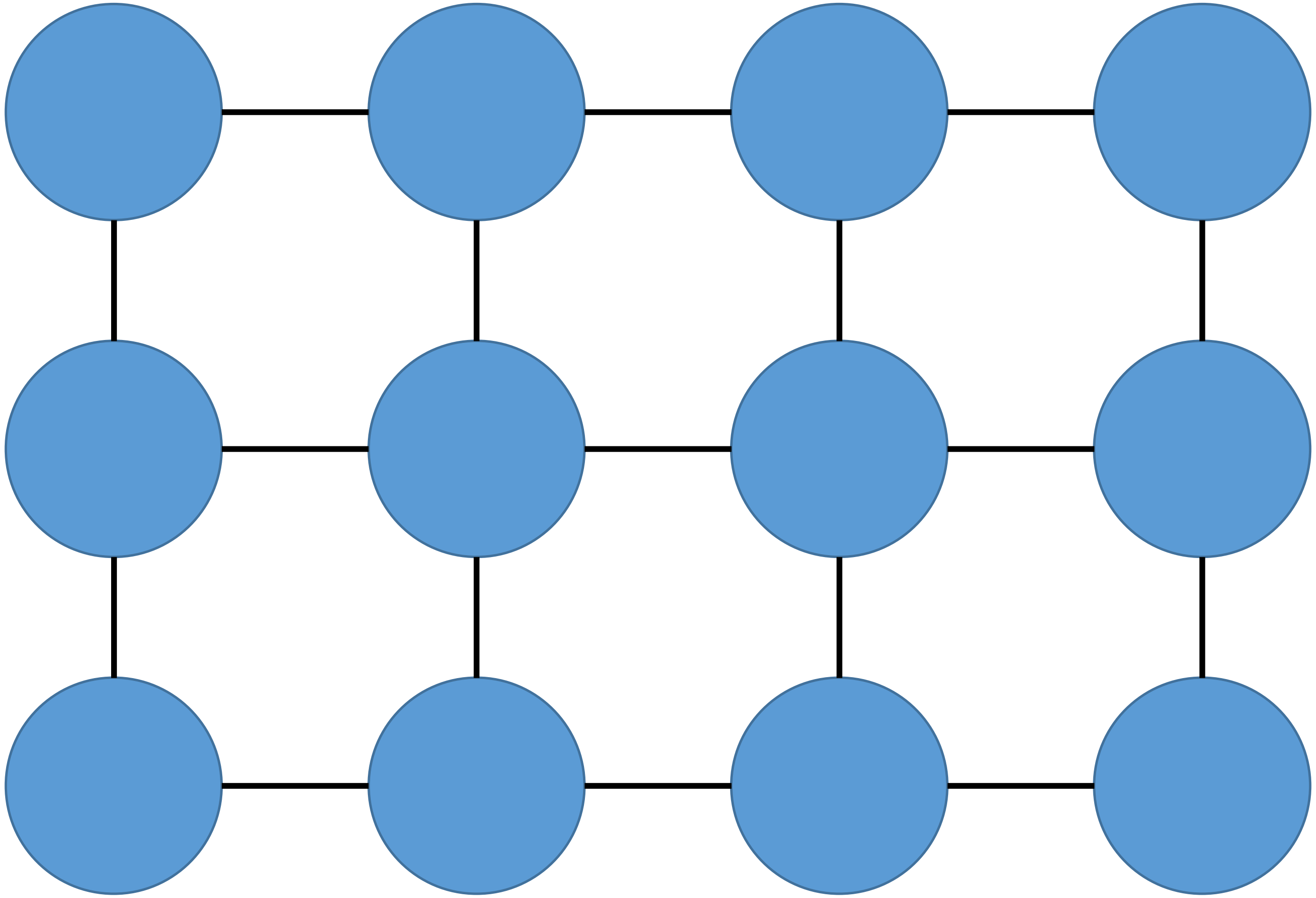
retargeting

segmentation

Image as a graph

Fundamental theme of today's lecture

Images can be viewed as graphs

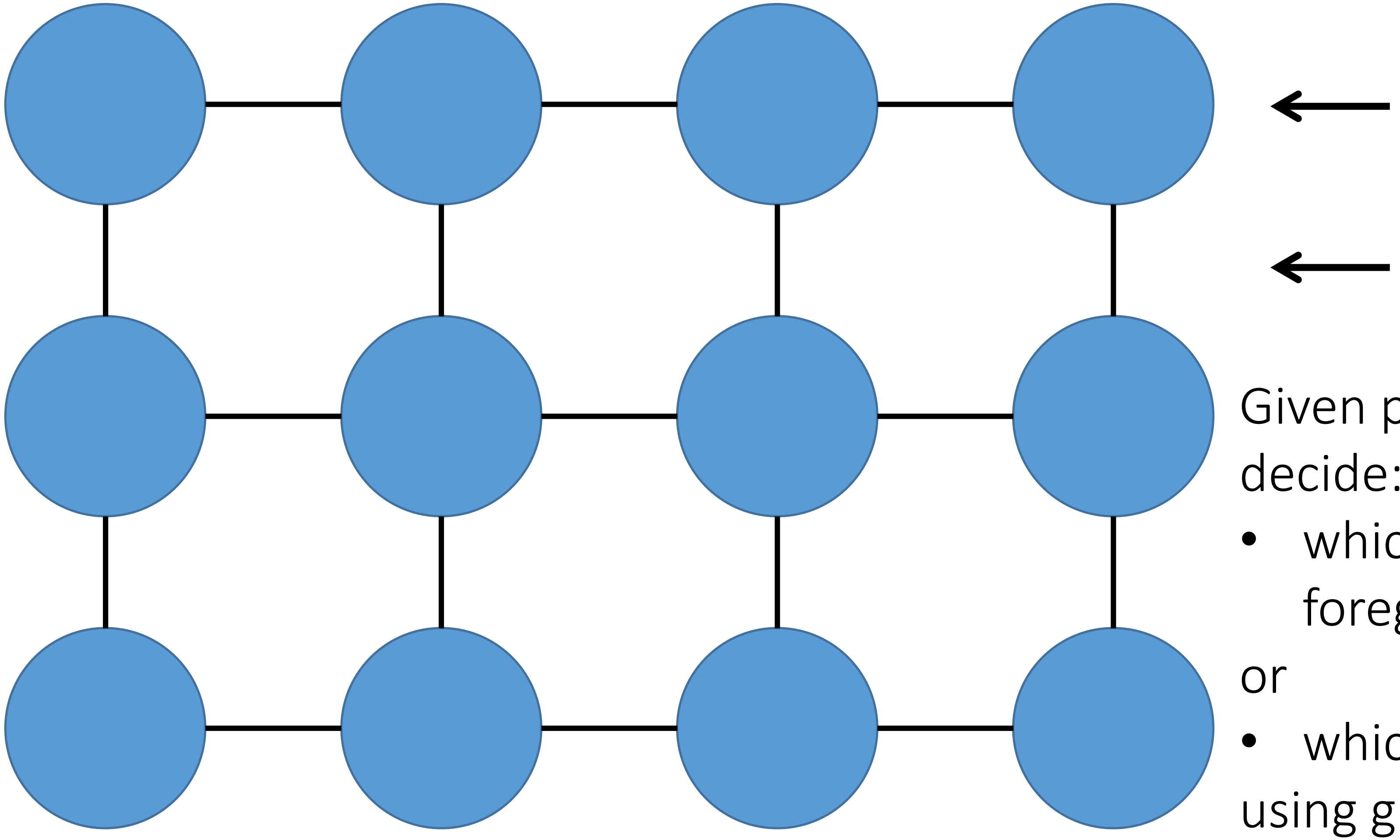


Nodes: pixels

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

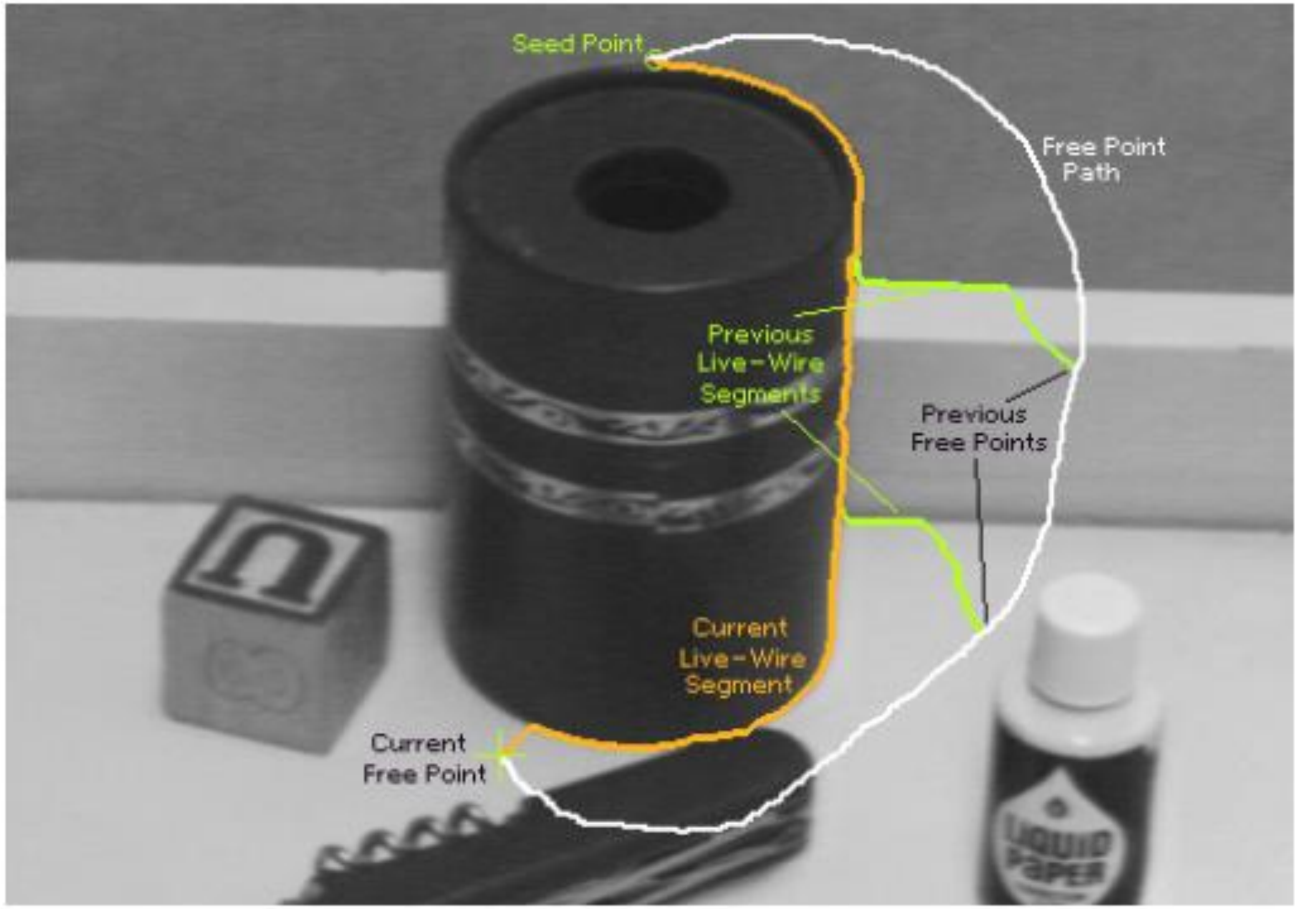
Intelligent scissors

Problem statement:

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

Challenges:

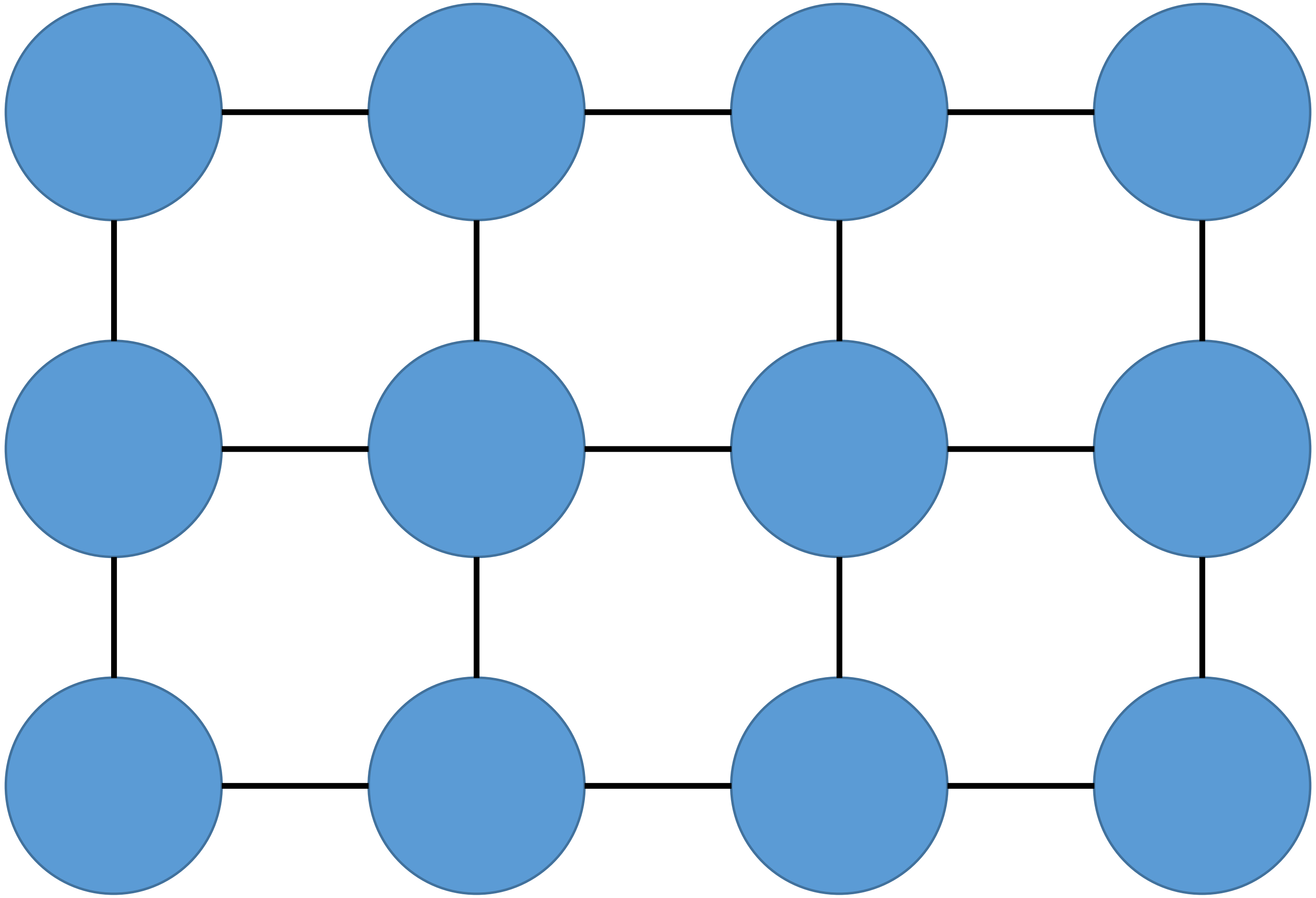
- Make this real-time for interaction
- Define what makes a good boundary



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

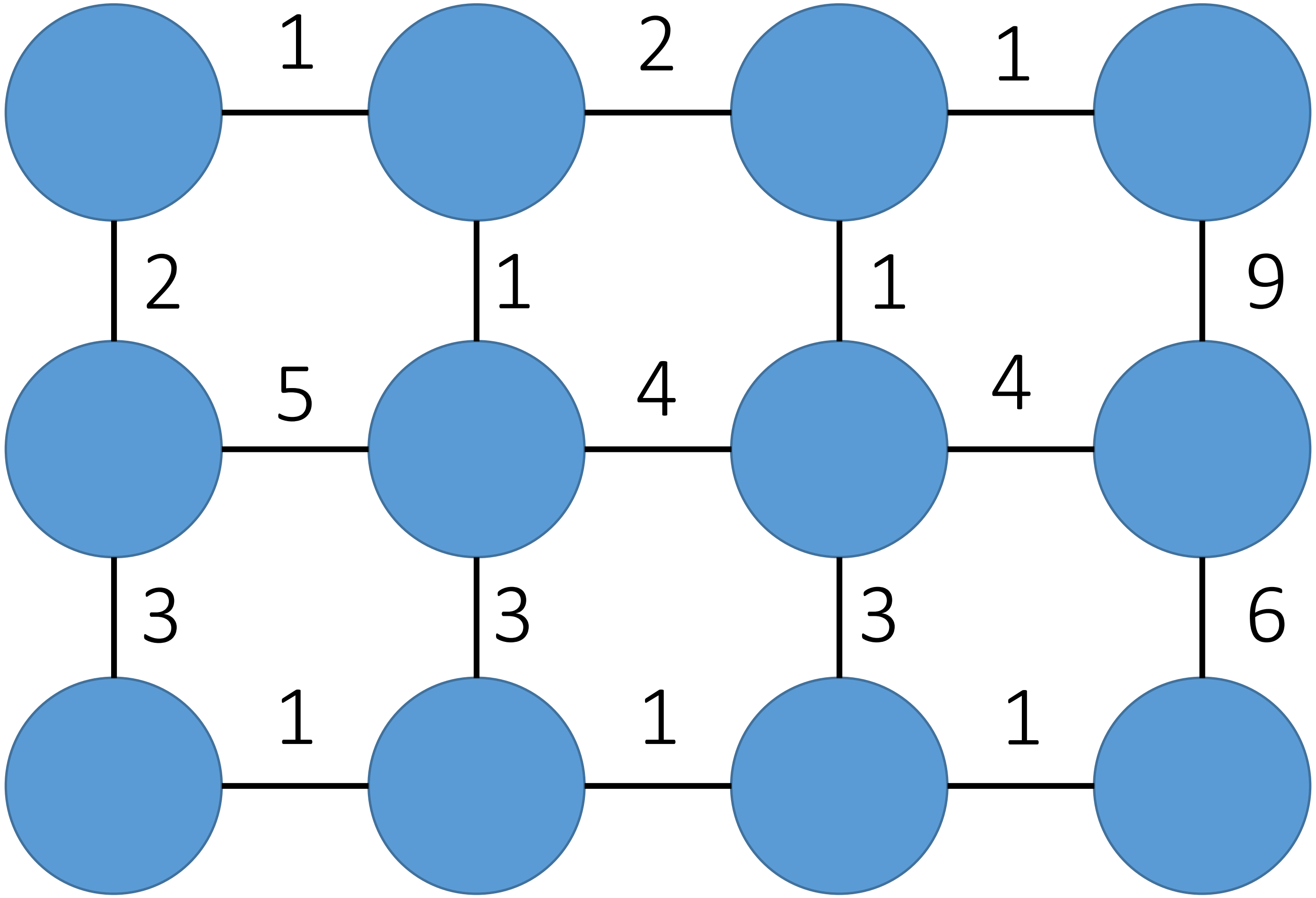


Nodes: pixels

Edges: Constraints between neighboring pixels

Graph-view of this problem

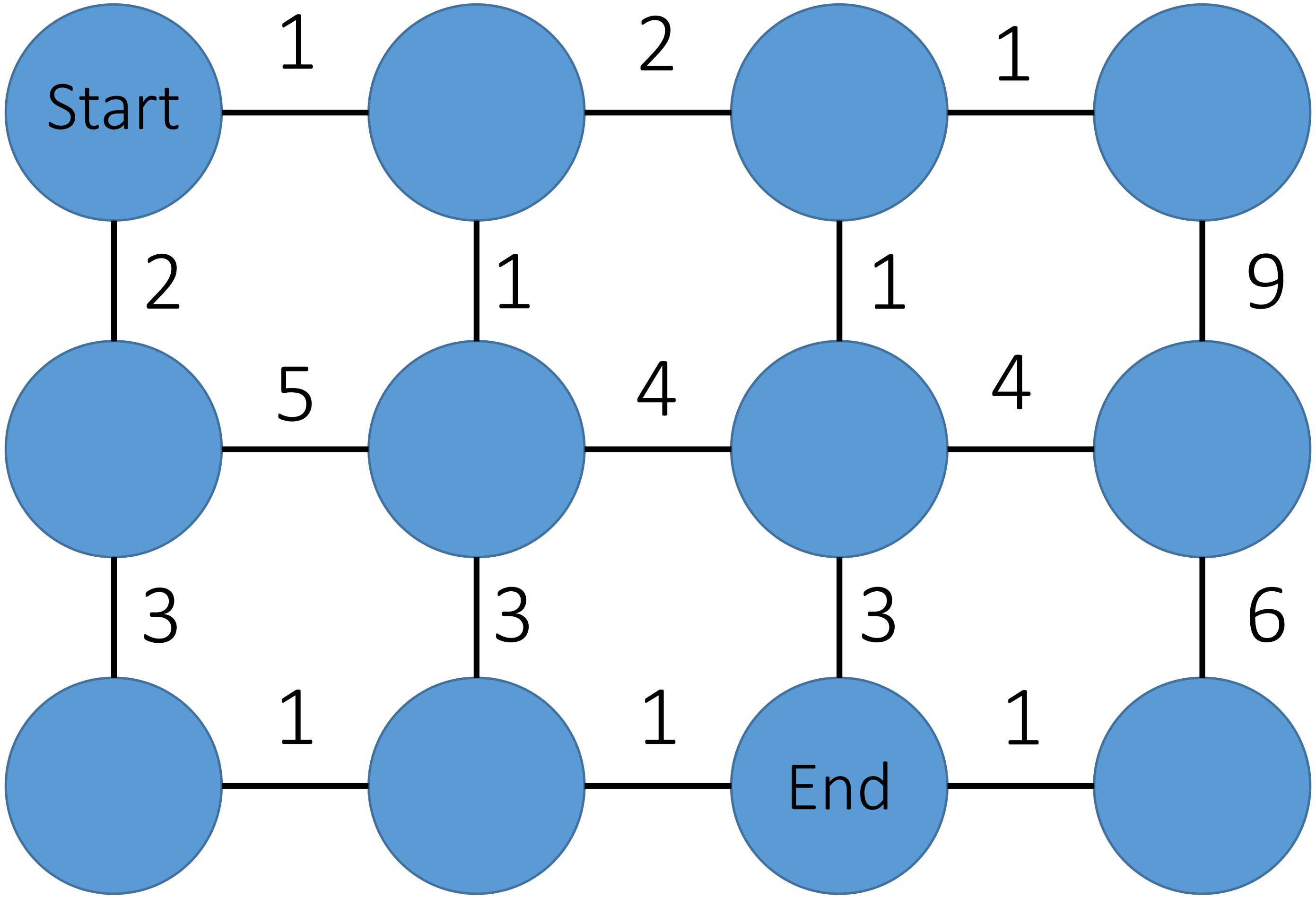
Graph-view of intelligent scissors:



1. Assign weights (costs) to edges

Graph-view of this problem

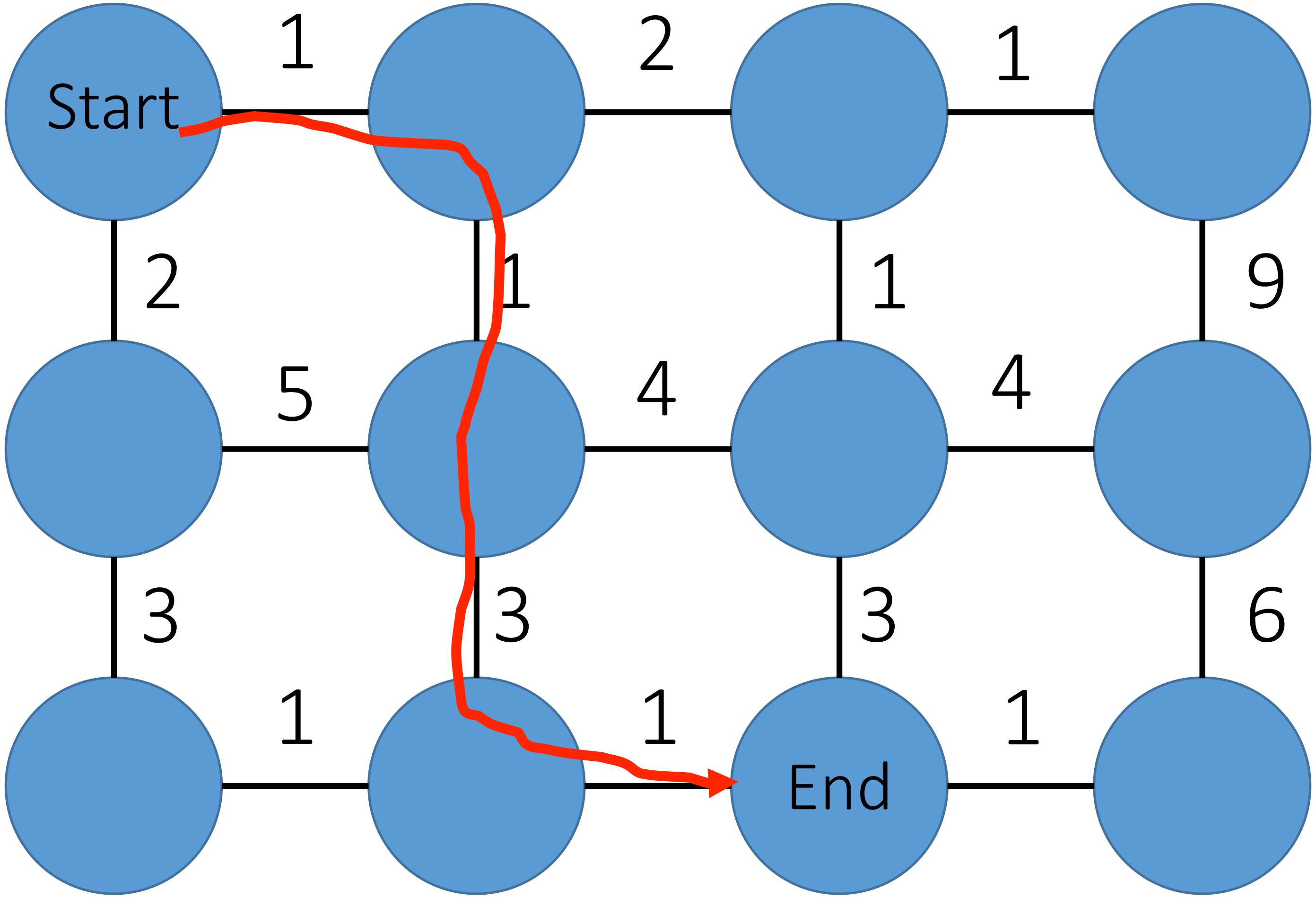
Graph-view of intelligent scissors:



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

Graph-view of this problem

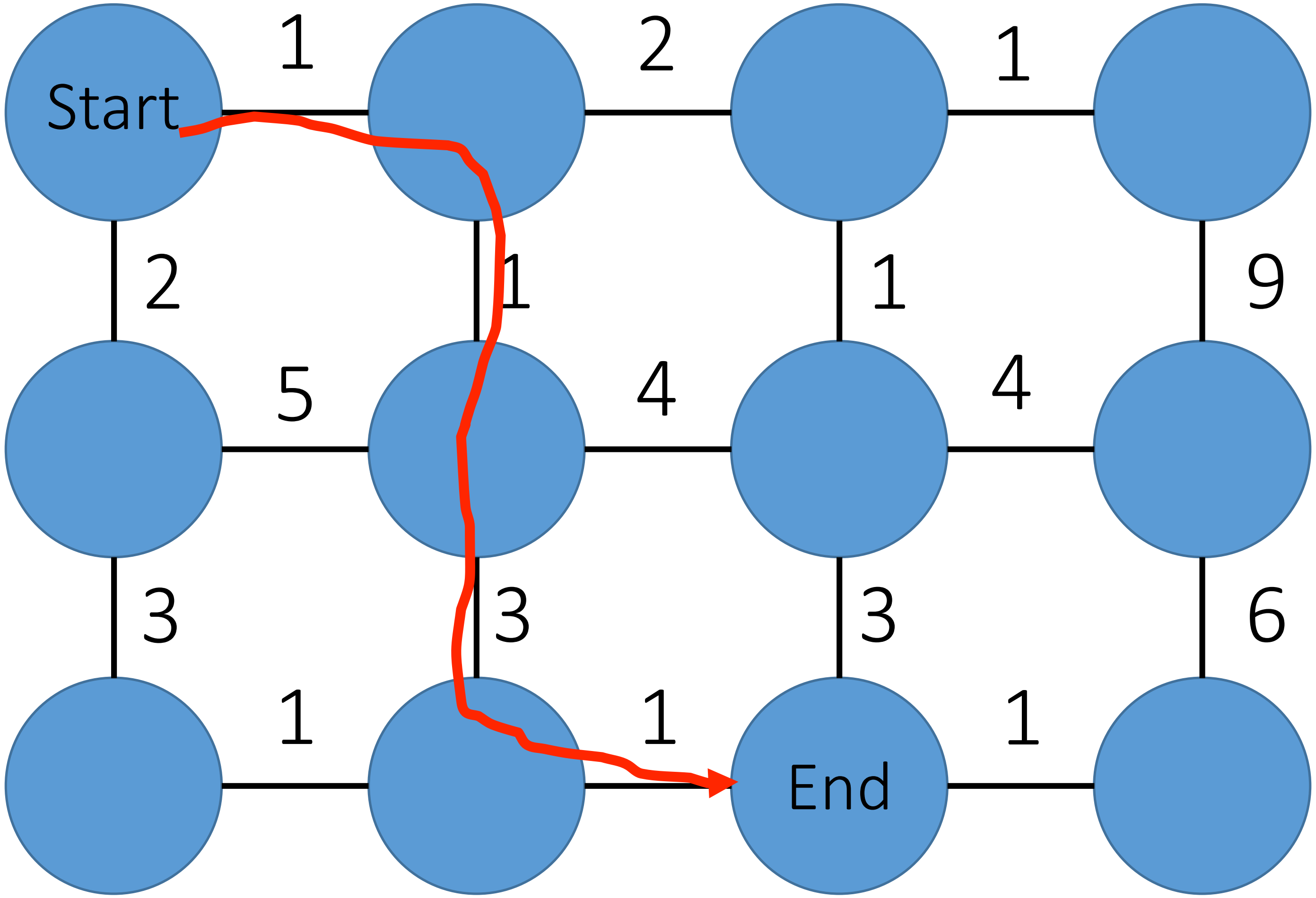
Graph-view of intelligent scissors:



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:

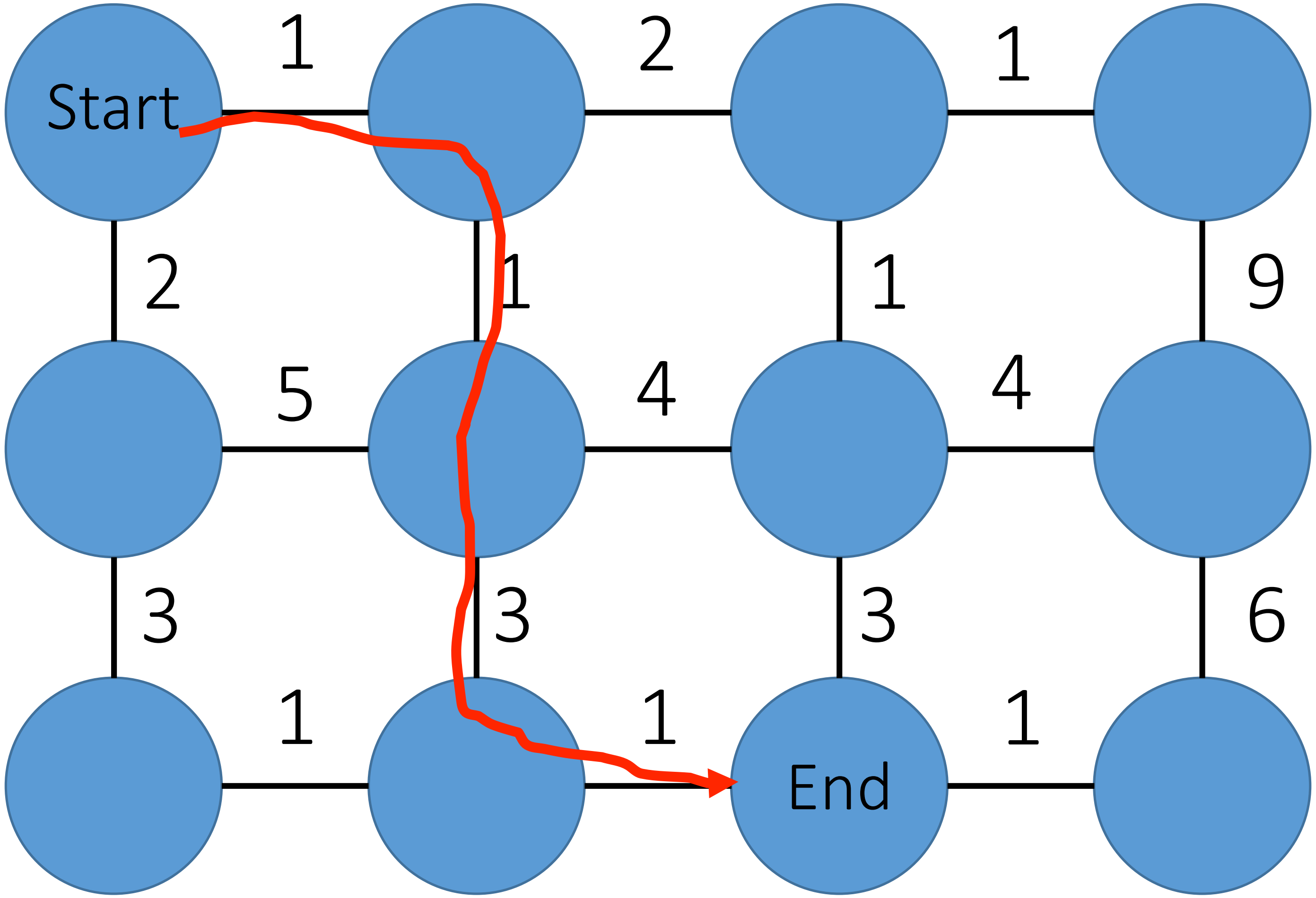


1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

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?

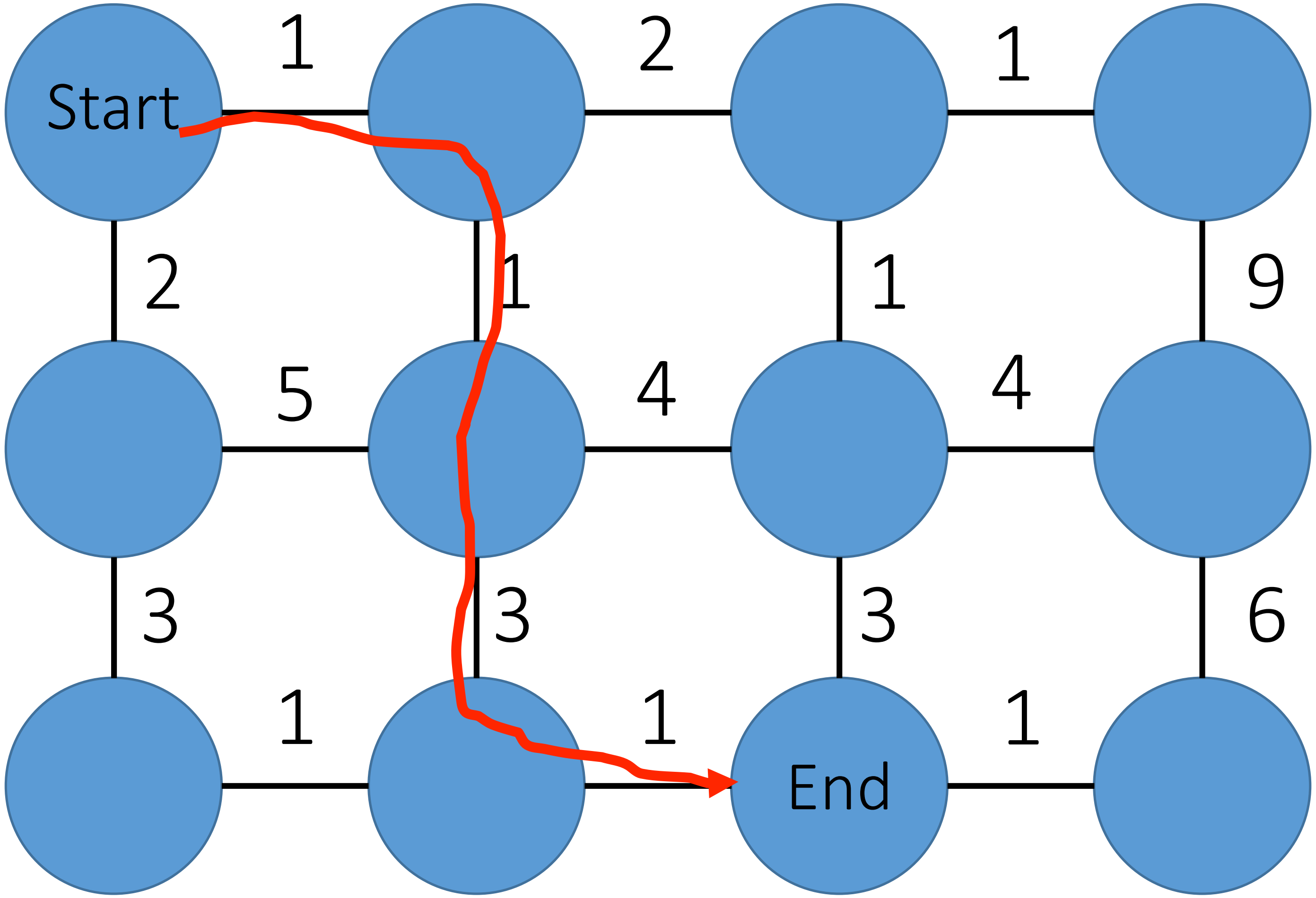
- Dijkstra's algorithm (dynamic programming)

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 cost2(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) + cost2(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

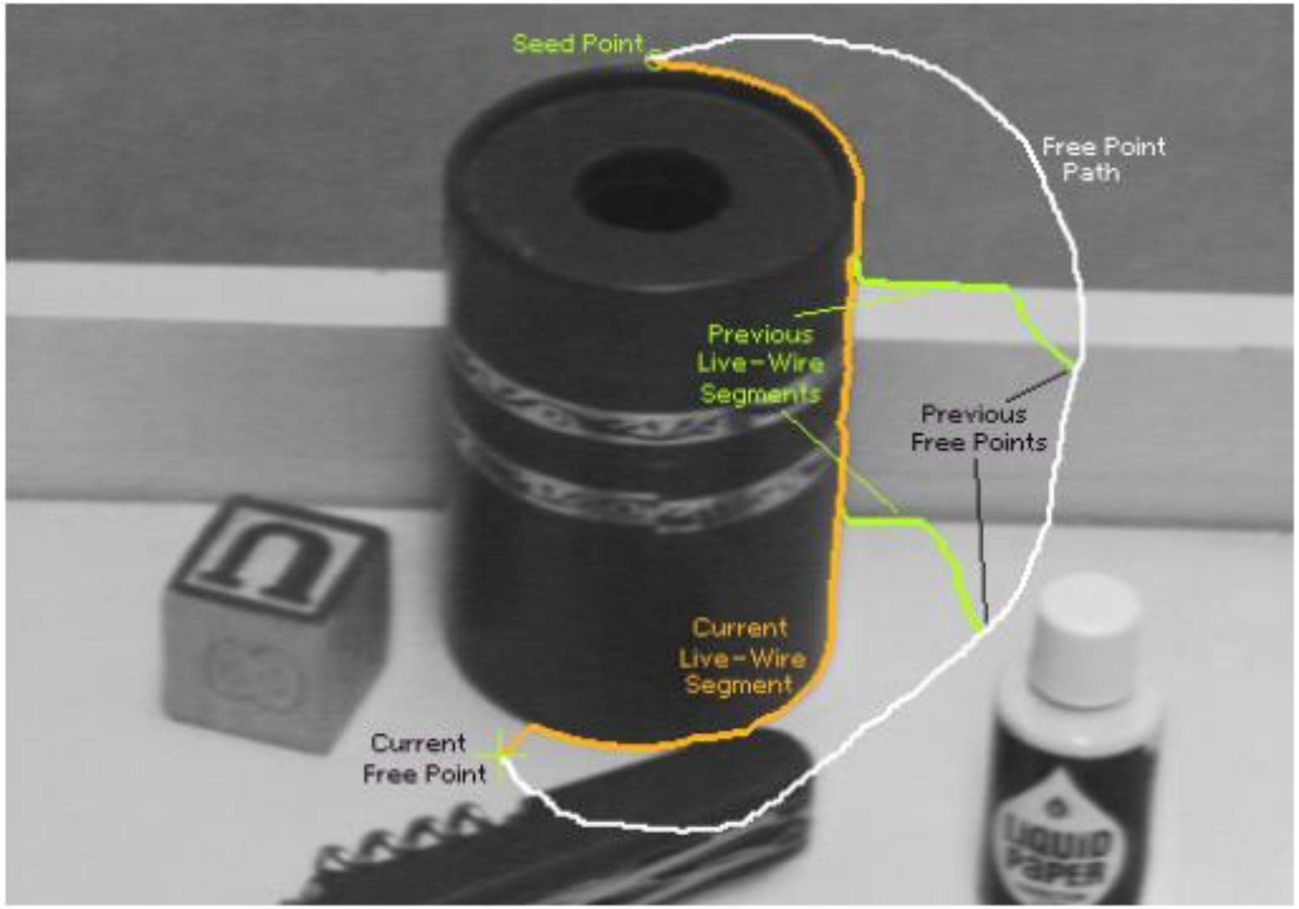
Pixel-wise cost

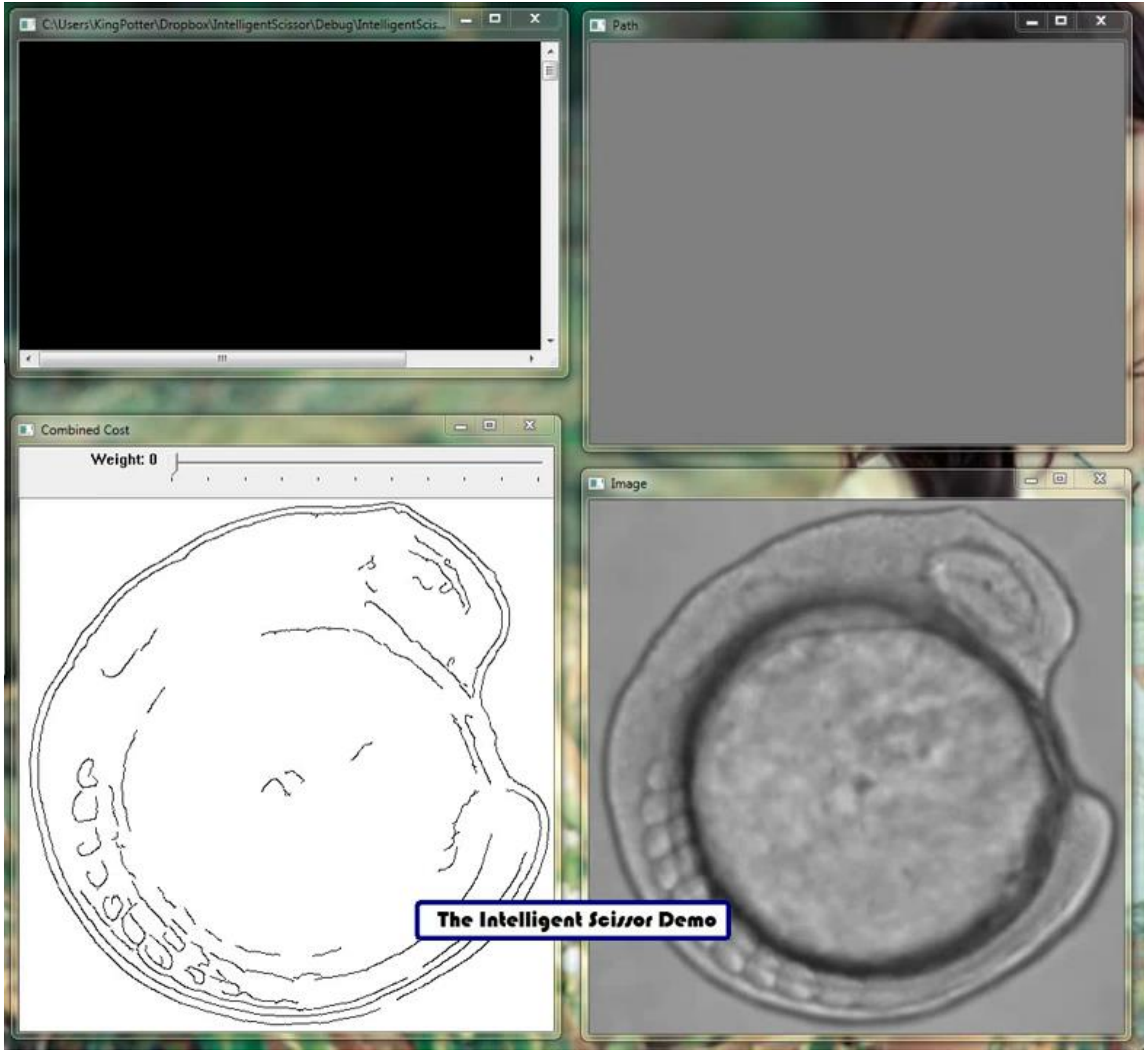
Edge image

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

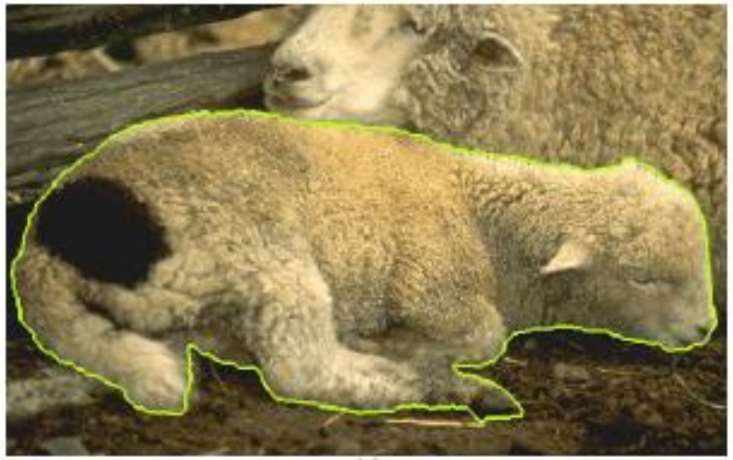



C:\Users\KingPotter\Dropbox\IntelligentScissor\Debug\IntelligentScis...
Path
Combined Cost
Weight: 0
Image
The Intelligent Scissor Demo

Examples

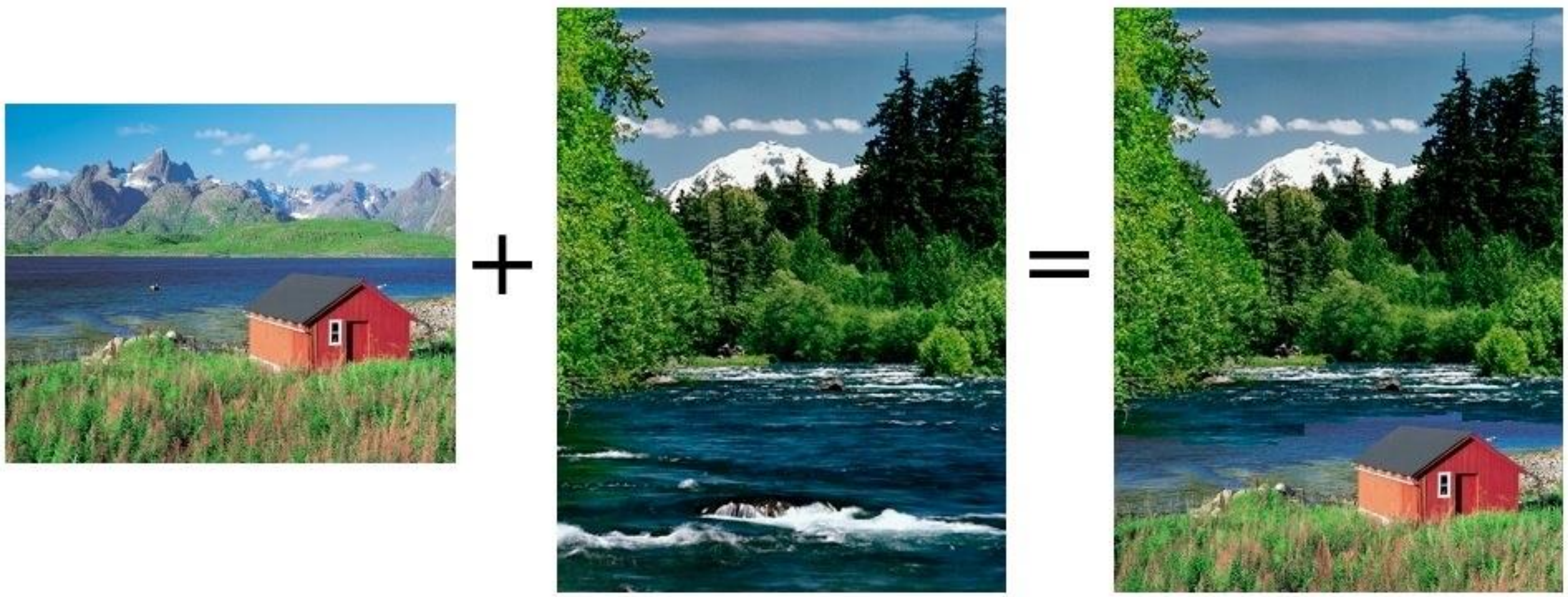

(a)


(b)


(c)

Seam collaging

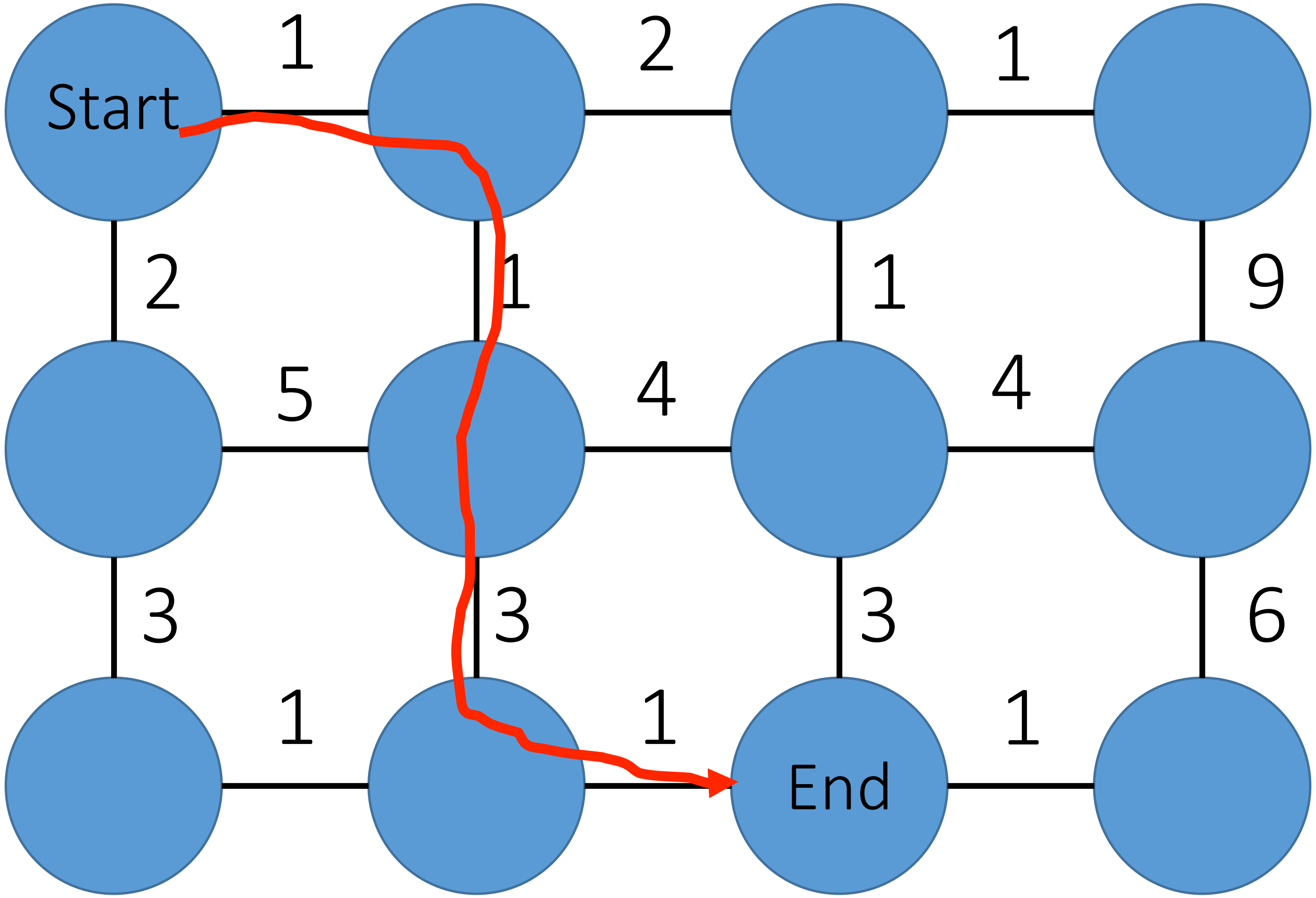
Another use for image seam selection



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

Graph-view of this problem

Graph-view of image collaging:

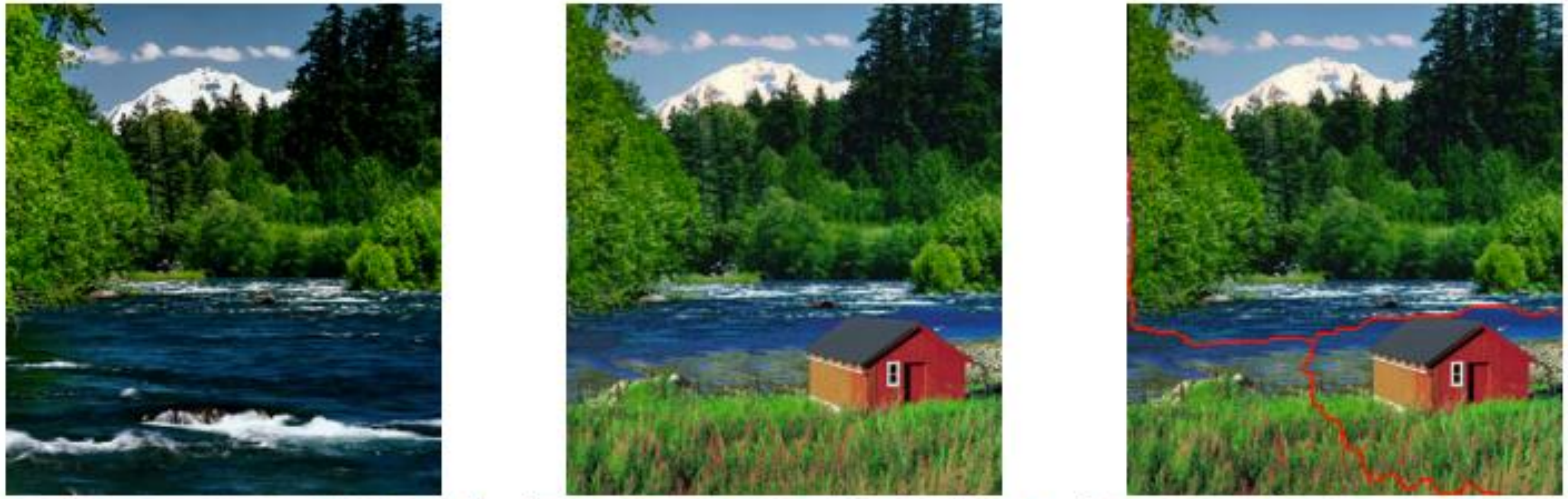


1. Assign weights (costs) to edges
2. Select the seed nodes
3. Find shortest path between them

What edge weights would you use for collaging?

Selecting edge weights for seam collaging

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

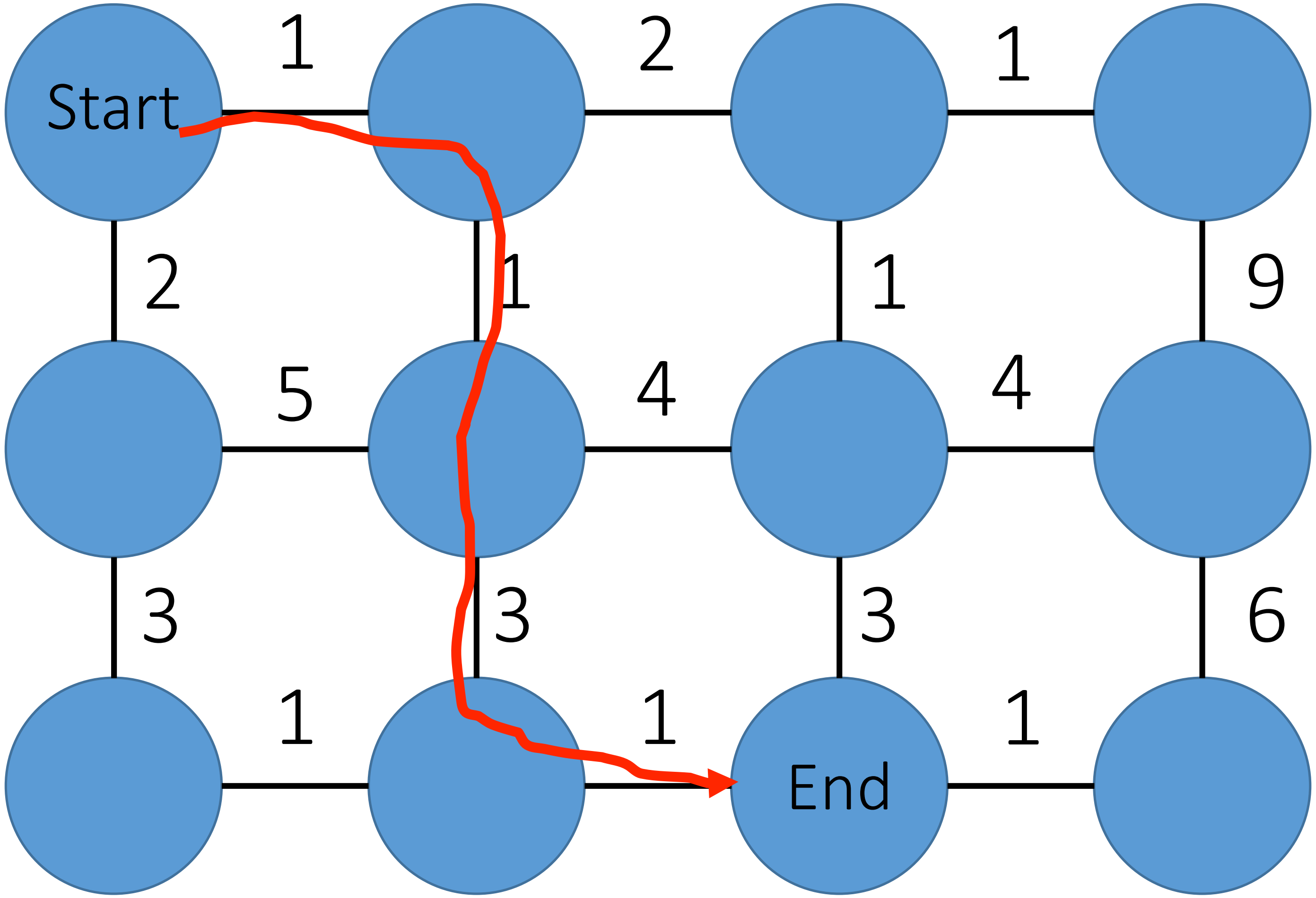

Seam carving

Another use for image seam selection

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

Graph-view of this problem

Graph-view of seam carving:



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

What edge weights would you use for seam carving?

Shai Avidan
Mitsubishi Electric Research Lab
Ariel Shamir
The interdisciplinary Center & MERL

Question about blending (last lecture)

When blending multiple images of the same scene, moving objects become ghosts!

What can we do instead of blending?

Question about blending (last lecture)

When blending multiple images of the same scene, moving objects become ghosts!

Instead of blending the images, cut them and stitch them together!

Question about blending (last lecture)

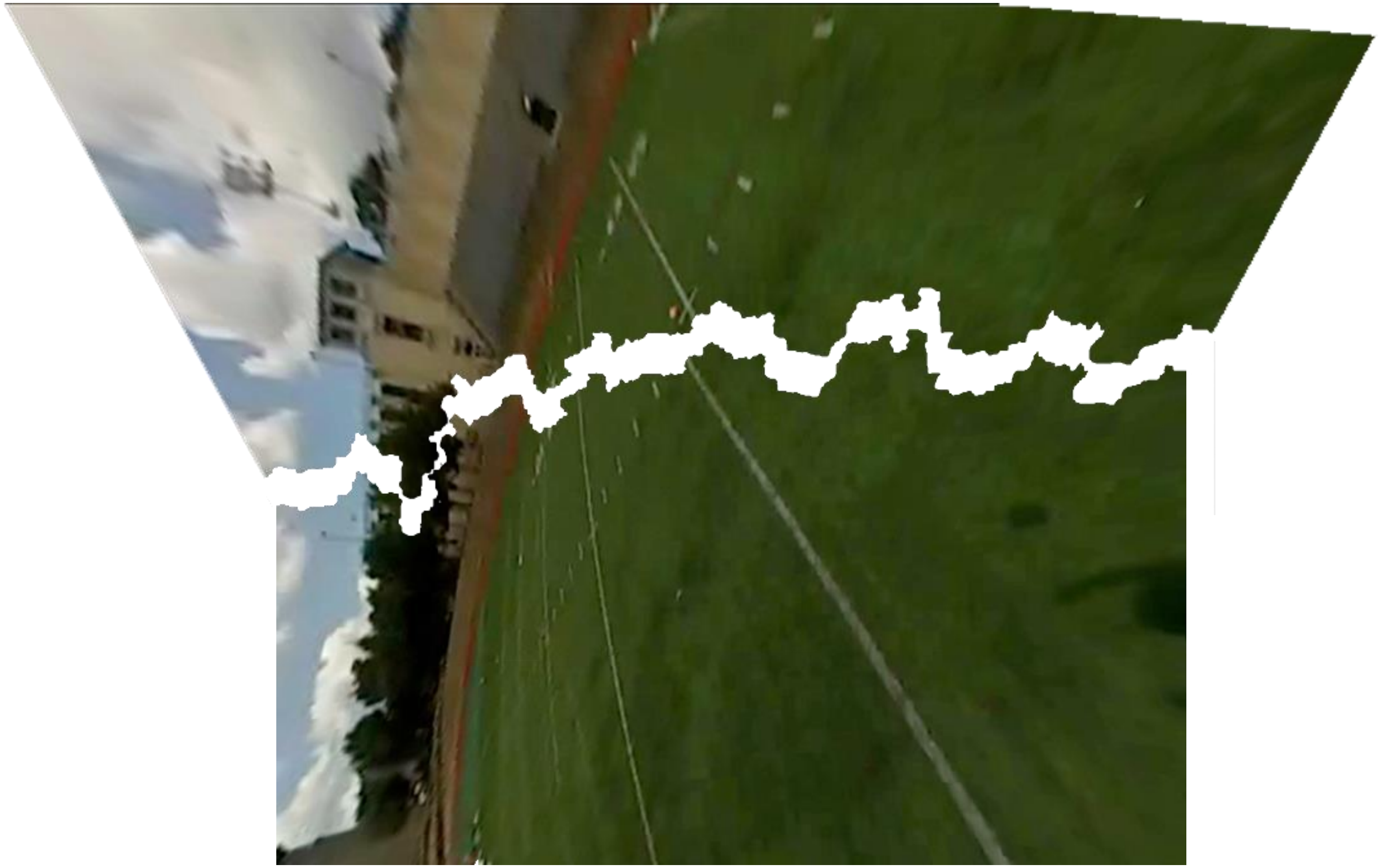
When blending multiple images of the same scene, moving objects become ghosts!

What can we do instead of blending?

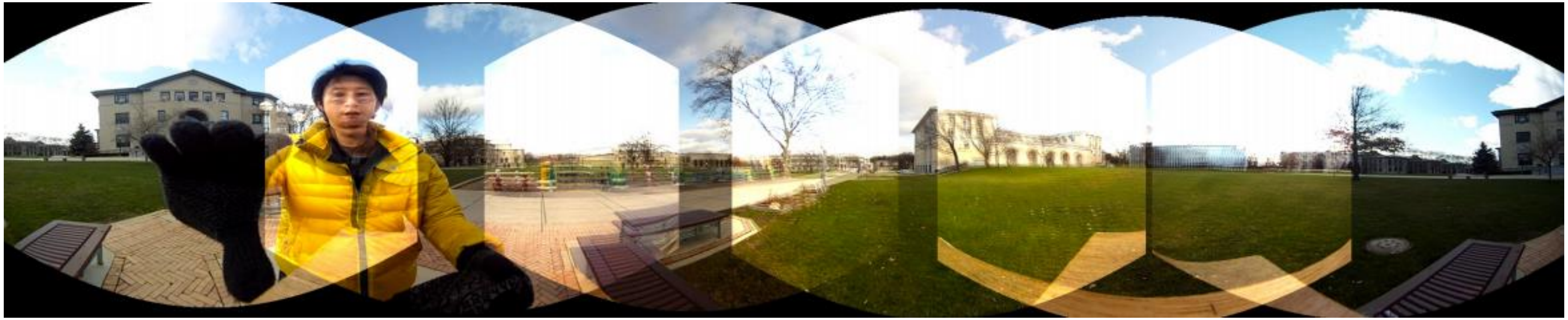
Seam stitching

Another use for image seam selection:

- instead of blending the images, cut them and stitch them together



Seam stitching

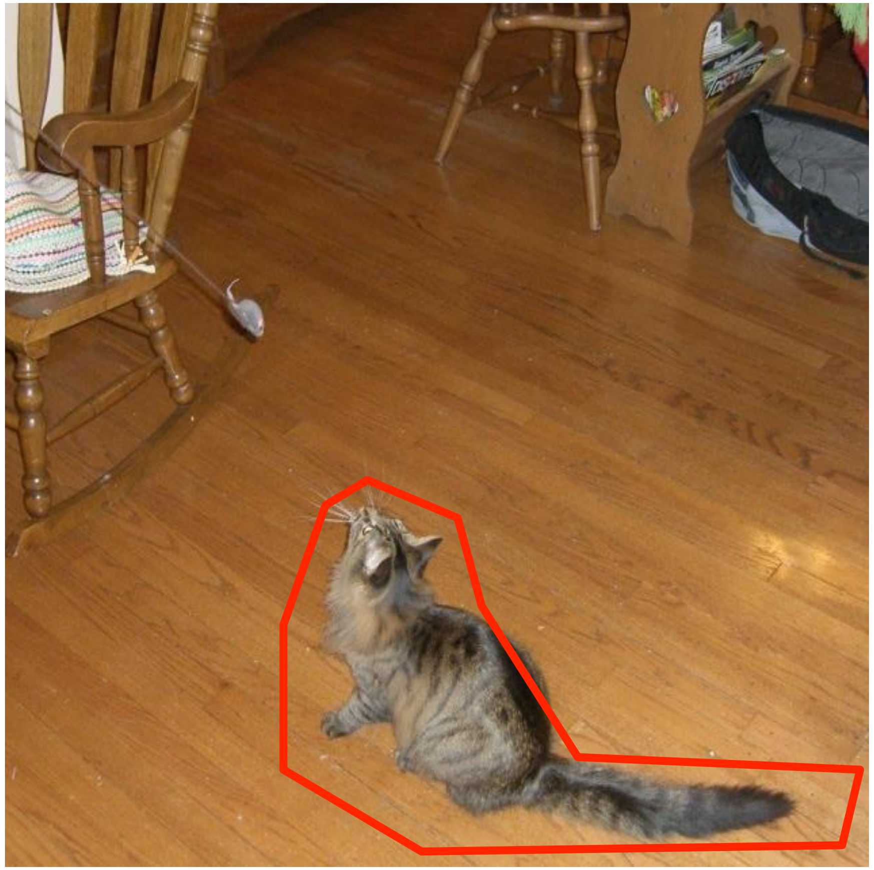


alpha blending

AutoStitch

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

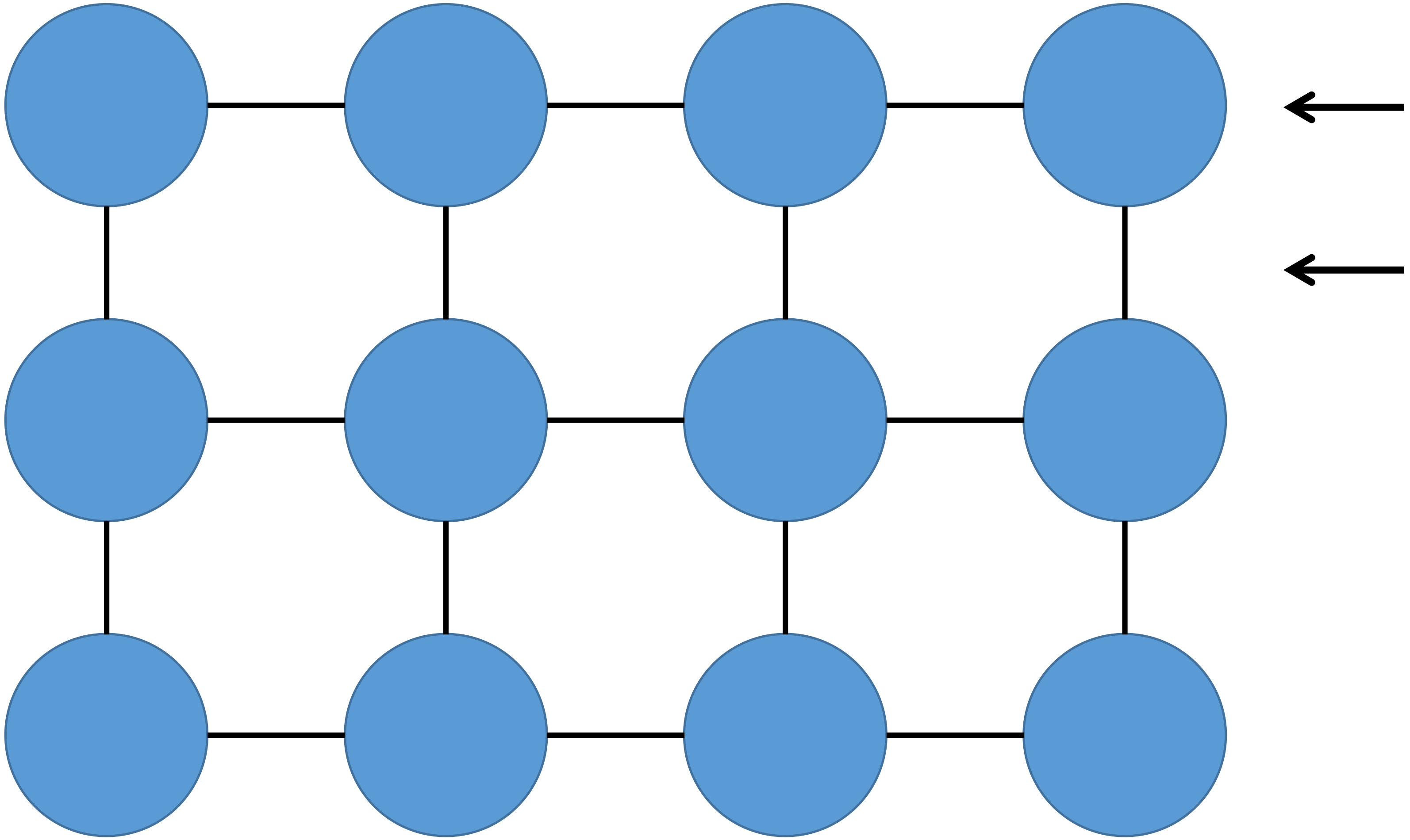
2. Foreground-background modeling using unsupervised clustering

GrabCut is a mixture of two components

1. Segmentation using graph cuts
2. Foreground-background modeling using unsupervised clustering

Segmentation using graph cuts

Remember: Graph-based view of images



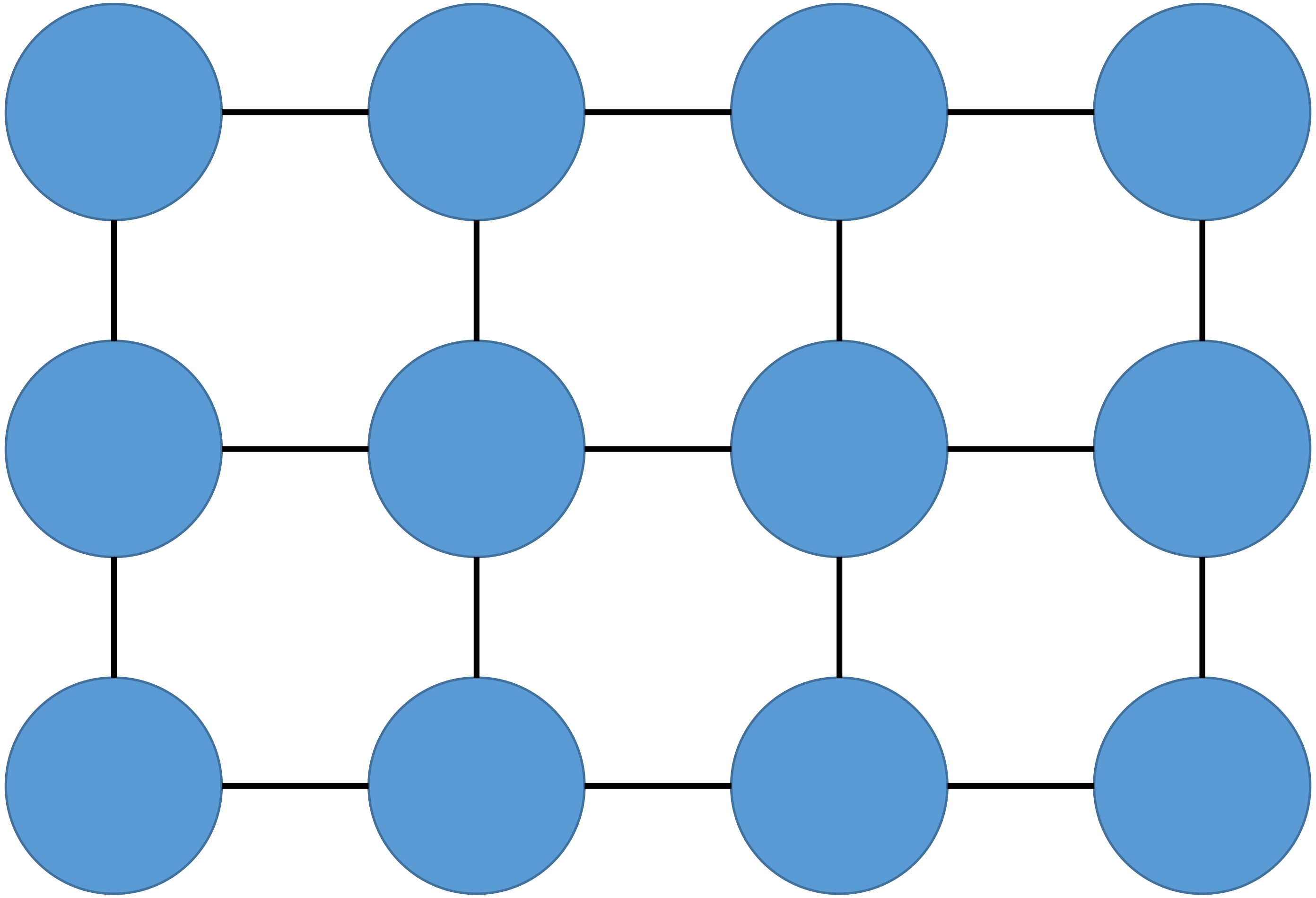
Nodes: pixels

Edges: Constraints between neighboring pixels

Markov Random Field (MRF)

Assign foreground/background labels based on:

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

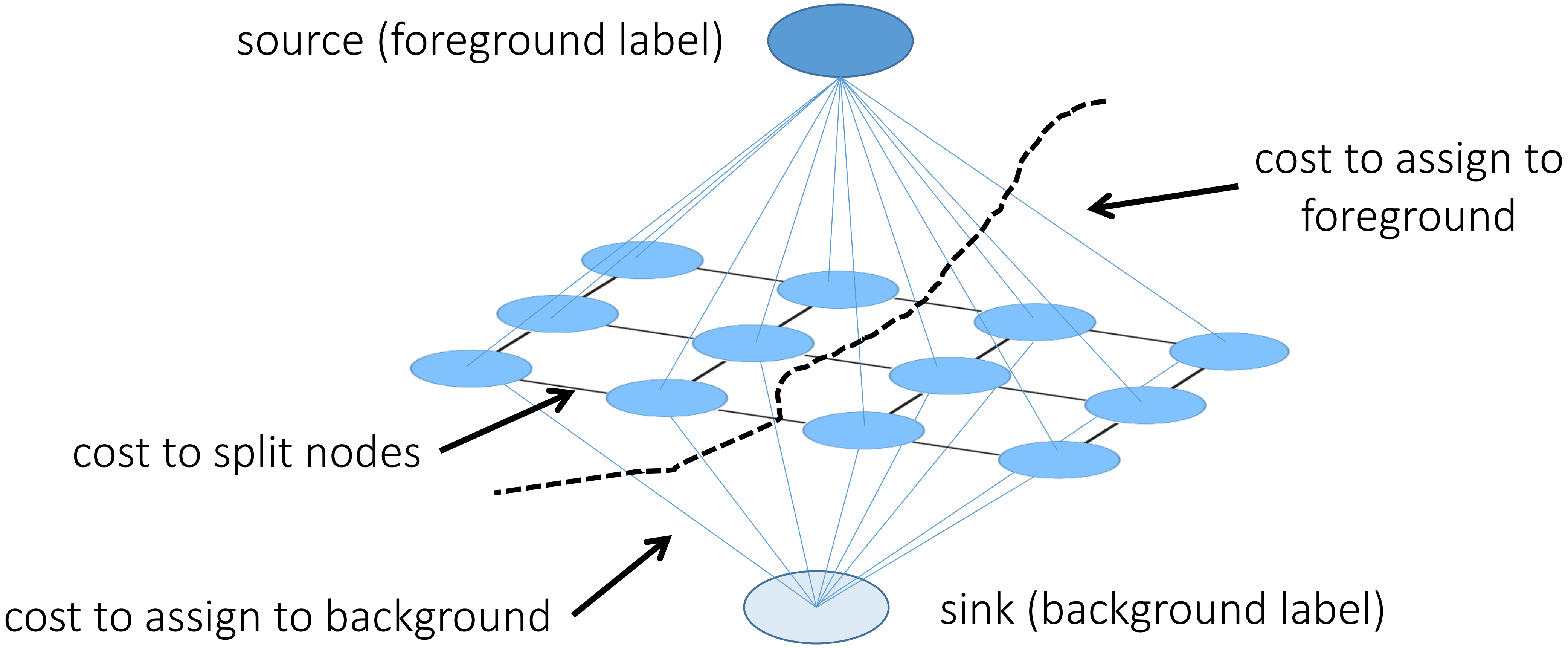


Given its intensity value, how likely is a pixel to be foreground or background?

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

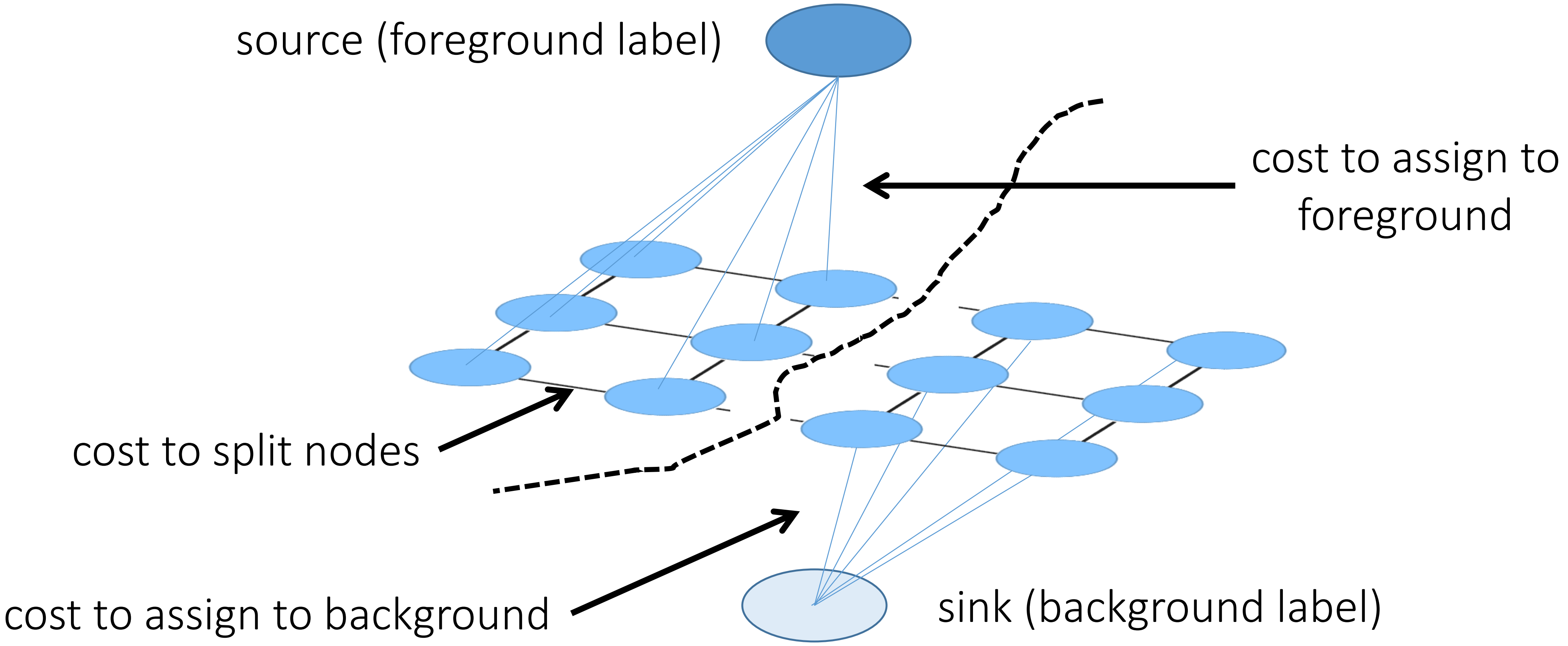
What kind of cost functions would you use for GrabCut?

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



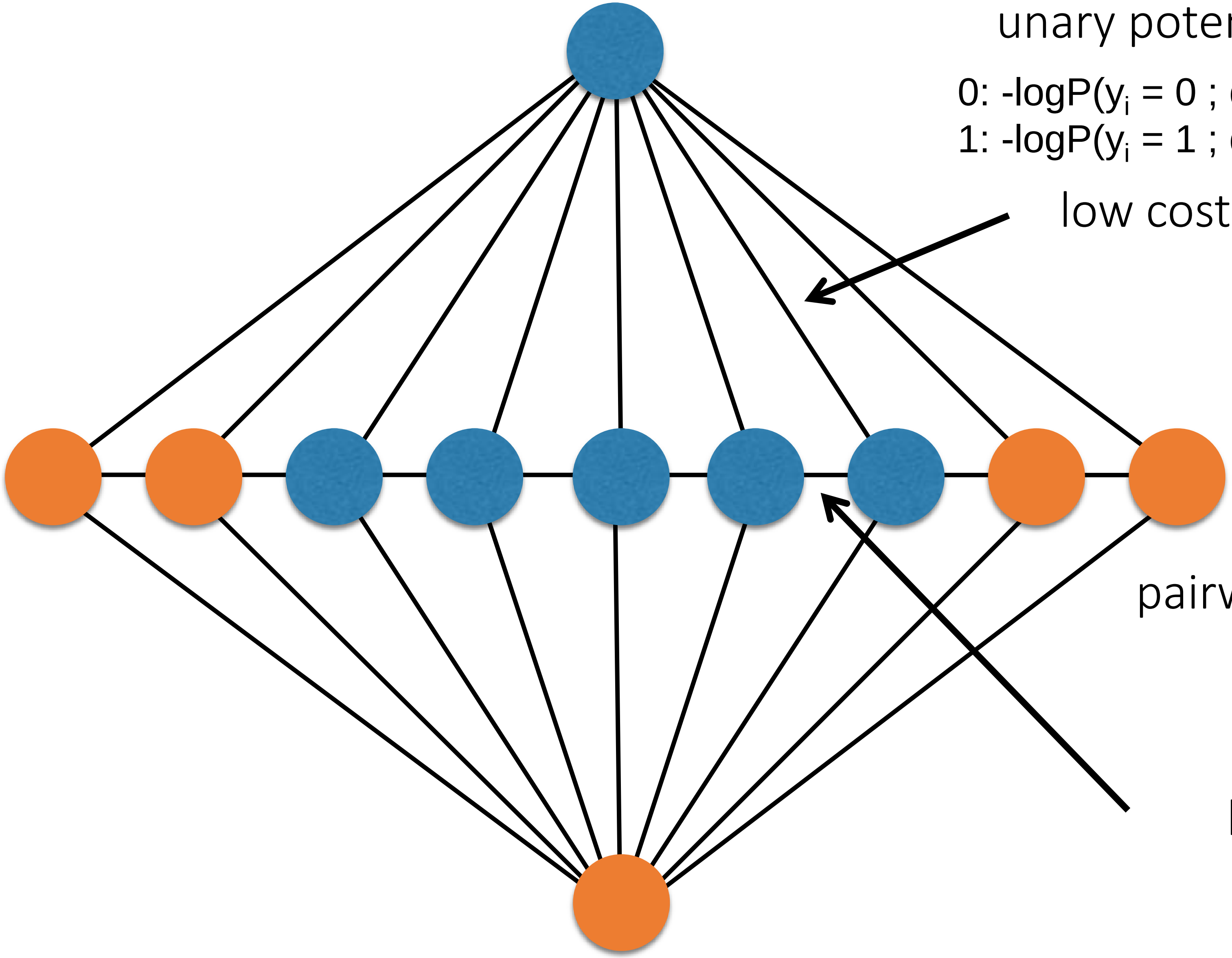
$$Energy\ (\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data)\ \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

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

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

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

GrabCut is a mixture of two components

1. Segmentation using graph cuts

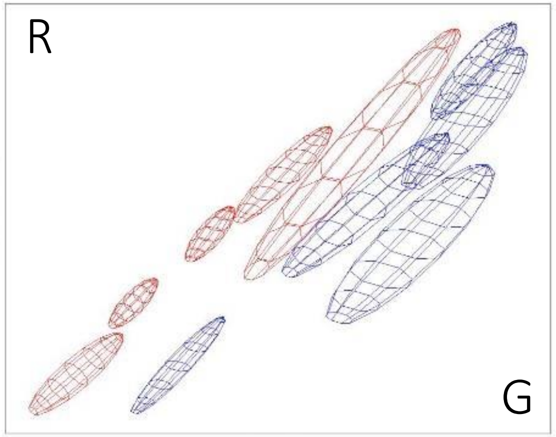
2. Foreground-background modeling using unsupervised clustering

Foreground-background modeling

Given foreground/background labels

build a color model for both

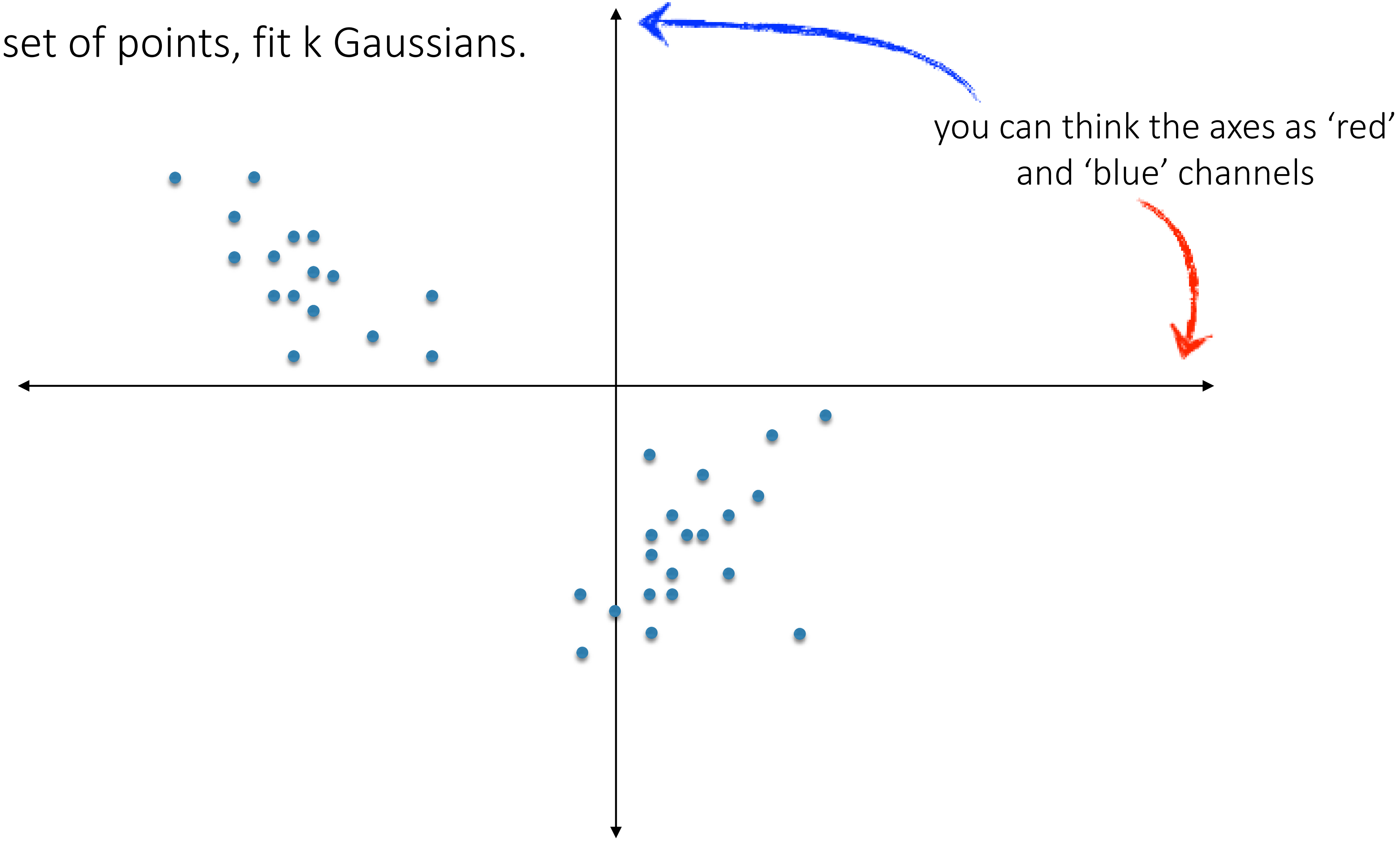
foreground



background

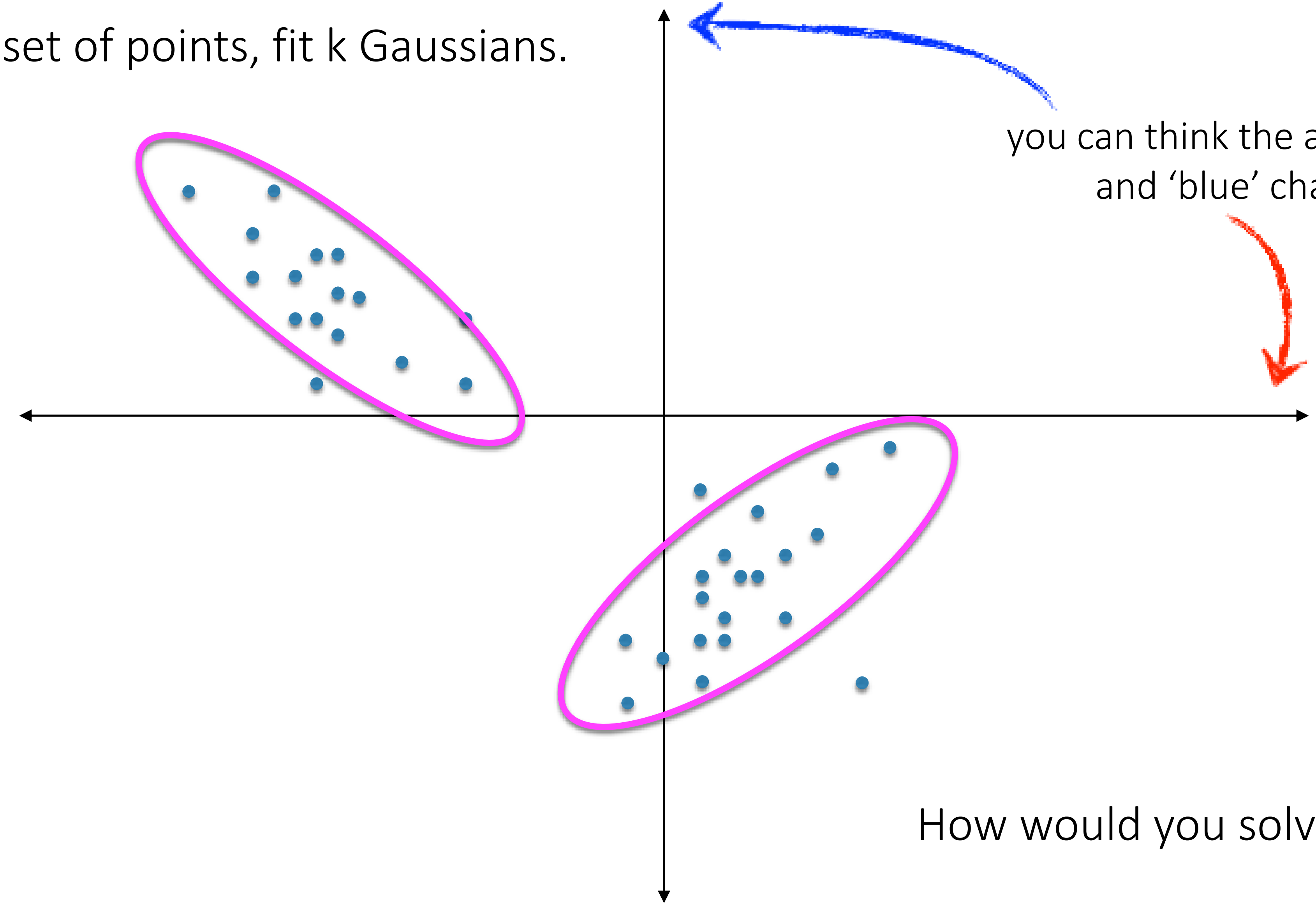
Learning color models

Given a set of points, fit k Gaussians.



Learning color models

Given a set of points, fit k Gaussians.



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

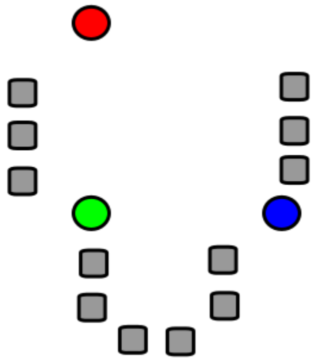
How would you solve this problem?

Intuition: “hard” clustering using K-means

Given k:

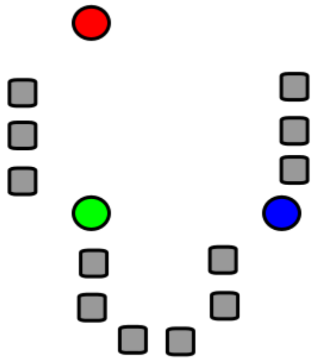
1. Select initial centroids at random.
2. Assign each object to the cluster with the nearest centroid.
3. Compute each centroid as the mean of the objects assigned to it.
4. Repeat previous 2 steps until no change.

K-means visualization

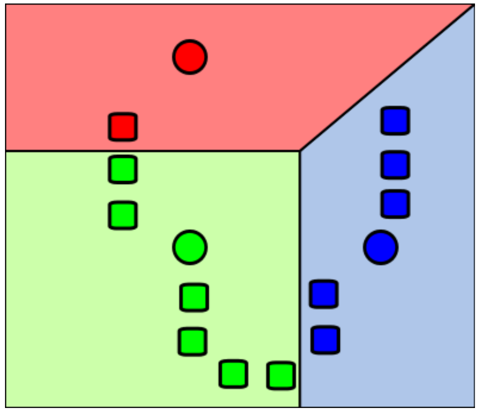


1. Select initial
centroids at random

K-means visualization

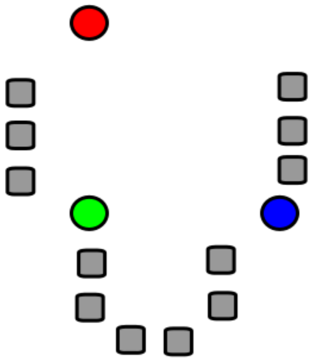


1. Select initial centroids at random

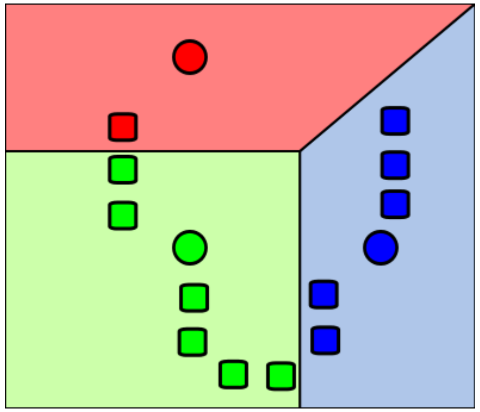


2. Assign each object to the cluster with the nearest centroid.

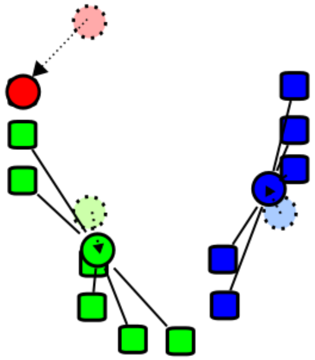
K-means visualization



1. Select initial centroids at random

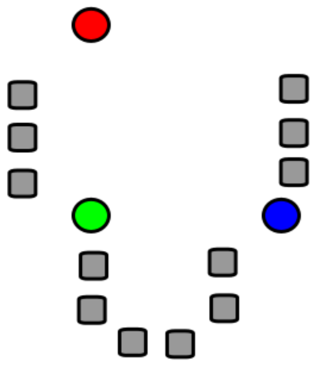


2. Assign each object to the cluster with the nearest centroid.

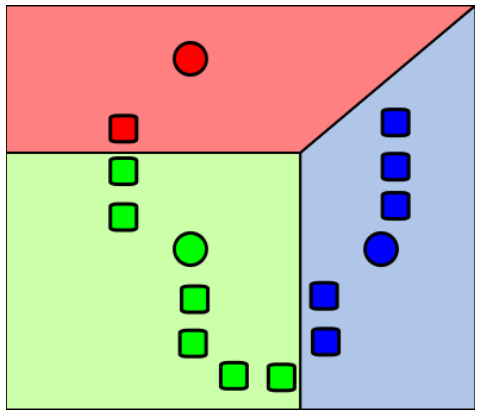


3. Compute each centroid as the mean of the objects assigned to it (go to 2)

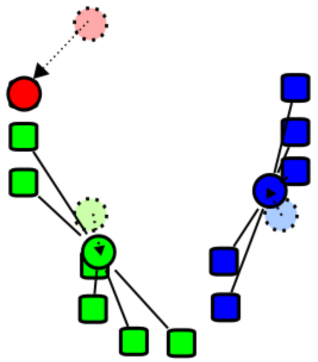
K-means visualization



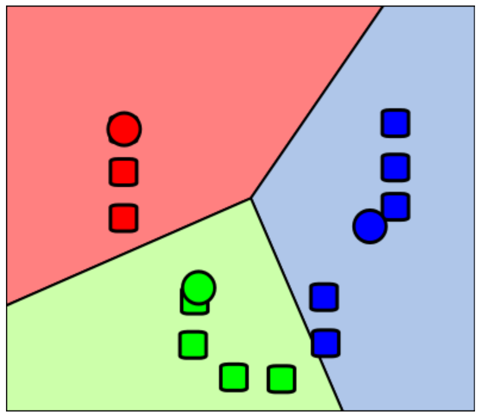
1. Select initial centroids at random



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



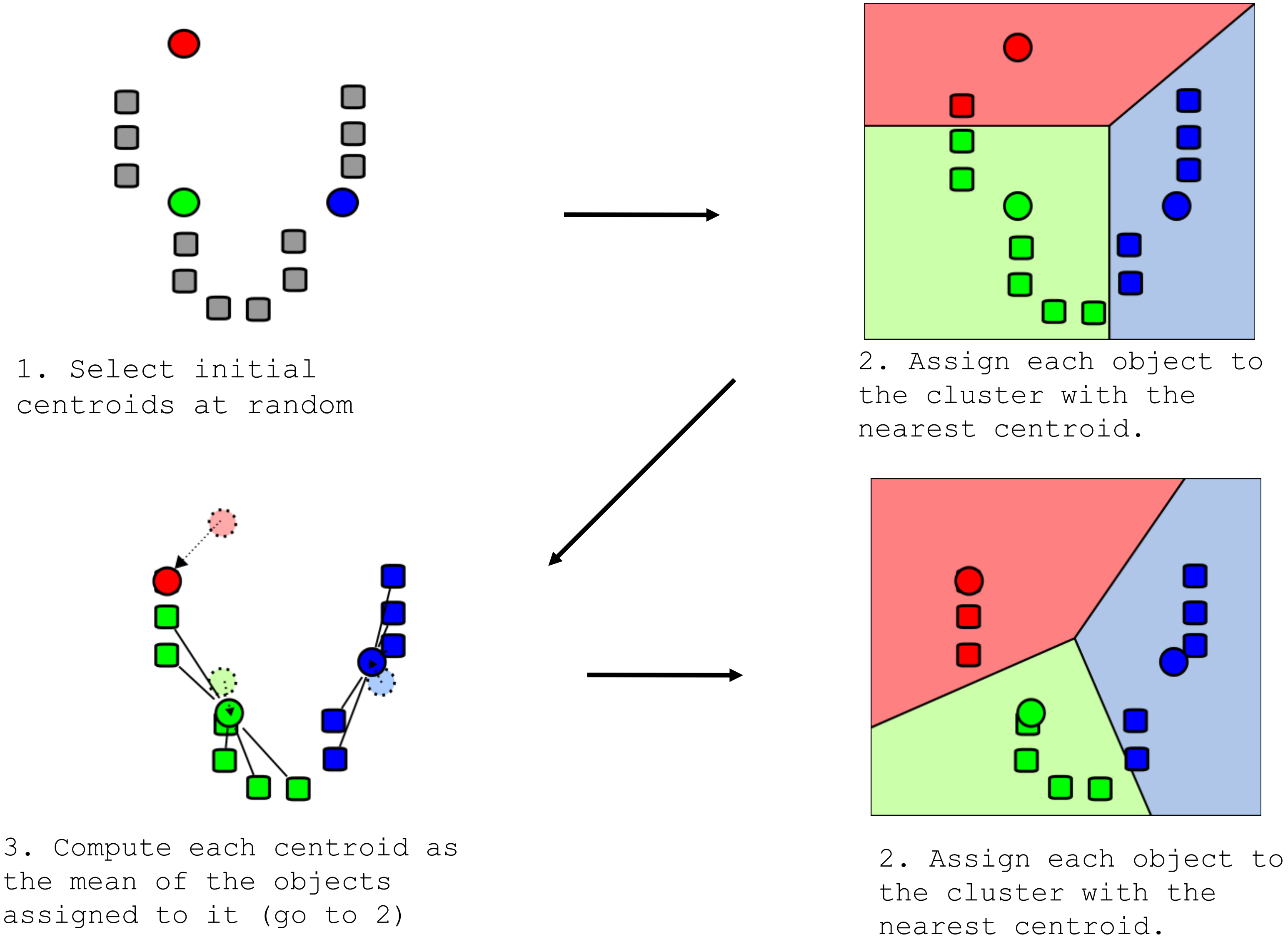
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



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

Repeat previous 2 steps until no change

K-means visualization



Expectation-Maximization: “soft” version of K-means

Given k:

1. Select initial centroids at random.

compute the probability of each object being in a cluster

E-step 2. ~~Assign each object to the cluster with the~~ nearest centroid.

and covariance

M-step 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

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

Compute the expected log-likelihood

M step

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

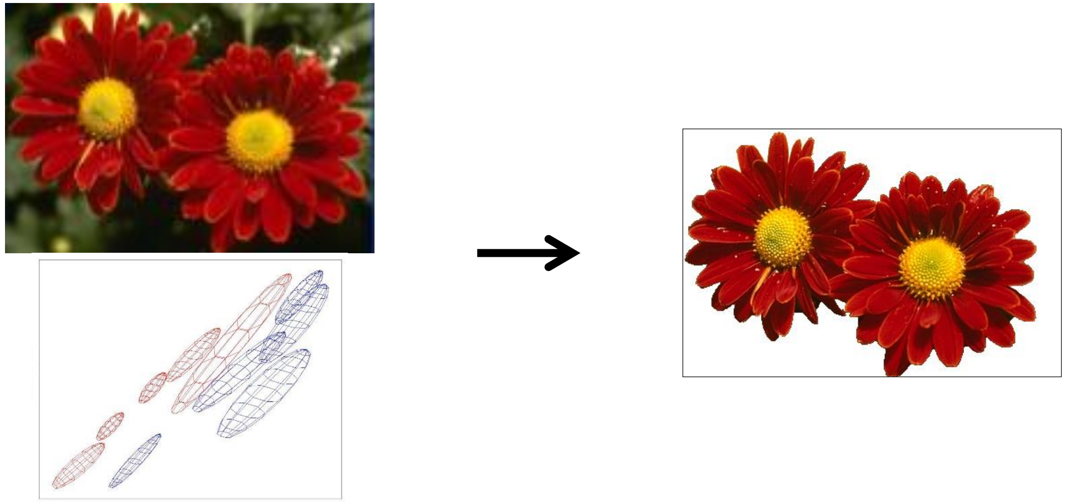
Update parameters based on likelihood

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

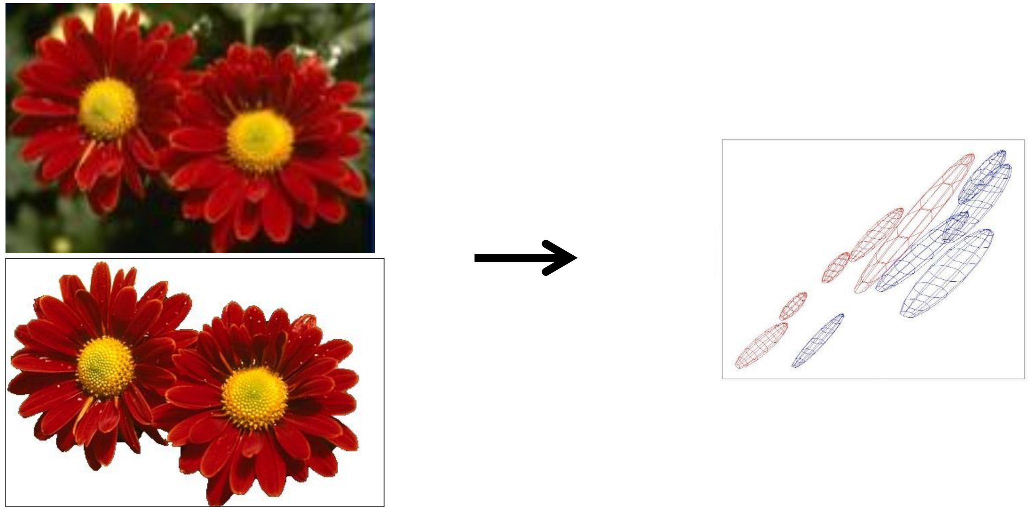
$$p(c(x);\boldsymbol{\theta}) = \prod_{k=1}^{K} \alpha_k \cdot \mathcal{N}(c(x);\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

GrabCut is a mixture of two components

1. Segmentation using graph cuts
   - Requires having foreground model



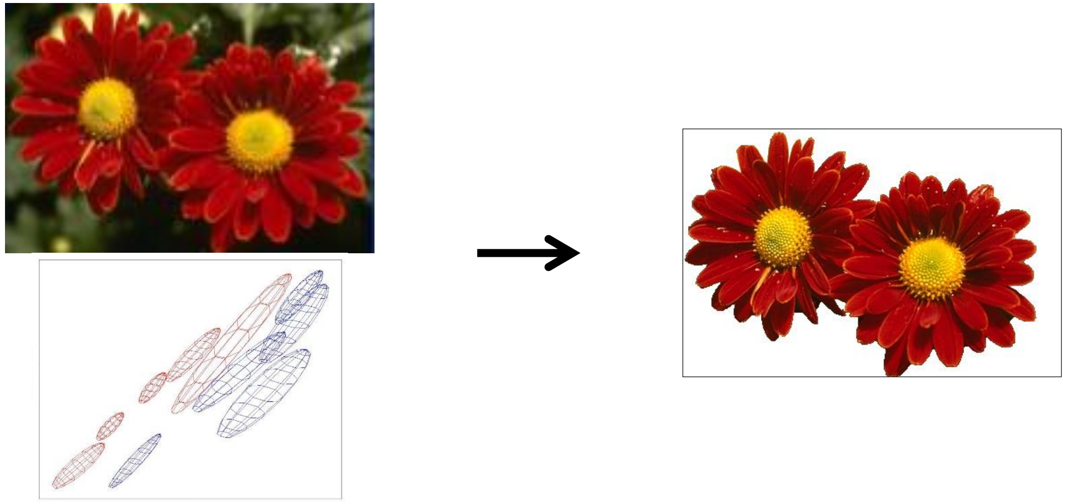
2. Foreground-background modeling using unsupervised clustering
   - Requires having segmentation



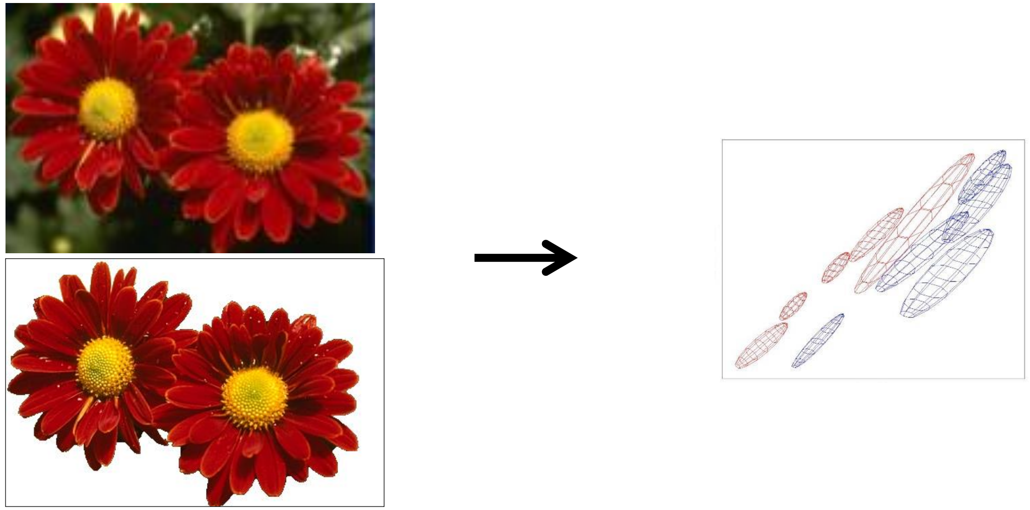
What do we do?

GrabCut: iterate between two steps

1. Segmentation using graph cuts
   - Requires having foreground model

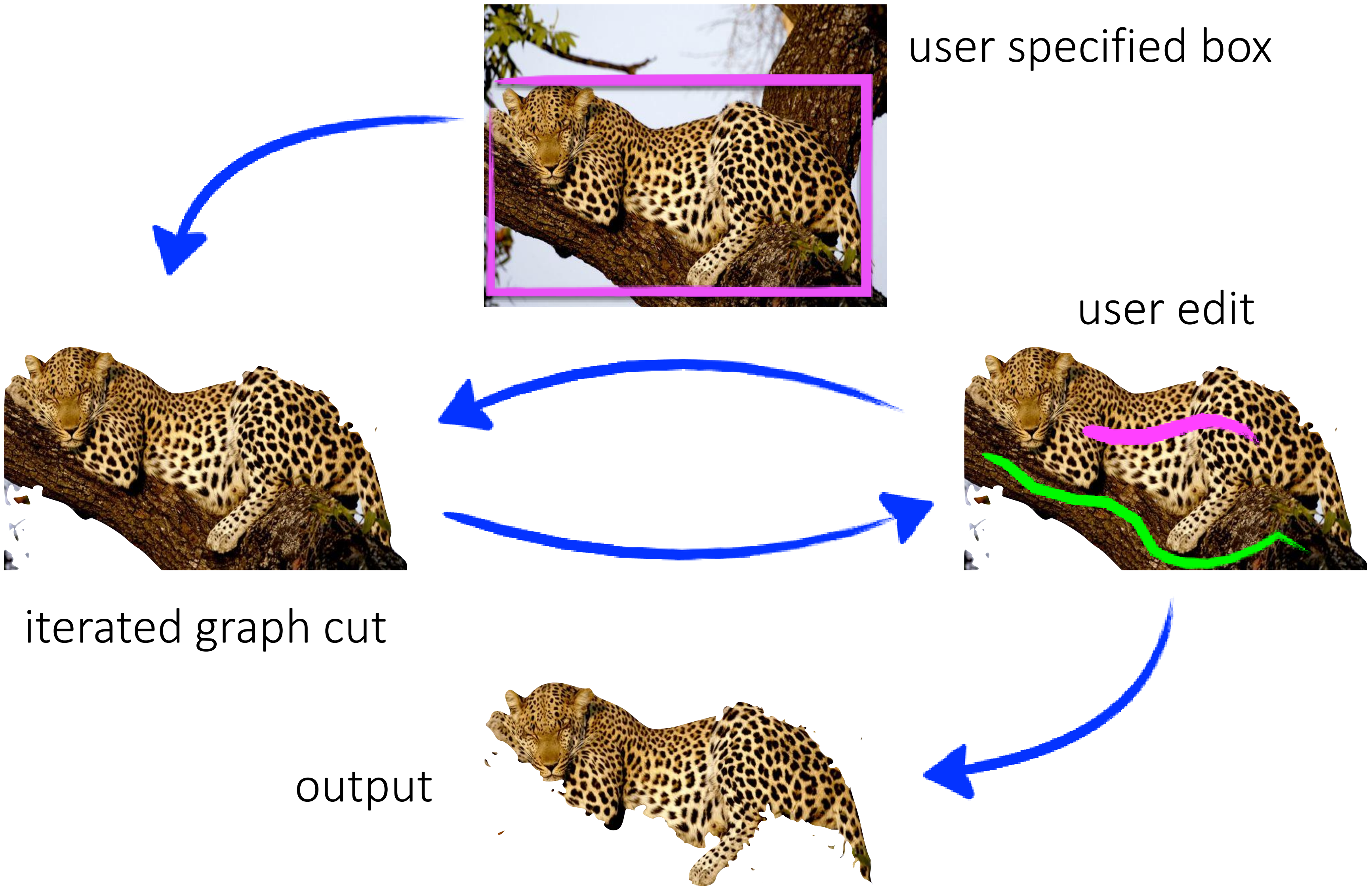


2. Foreground-background modeling using unsupervised clustering
   - Requires having segmentation

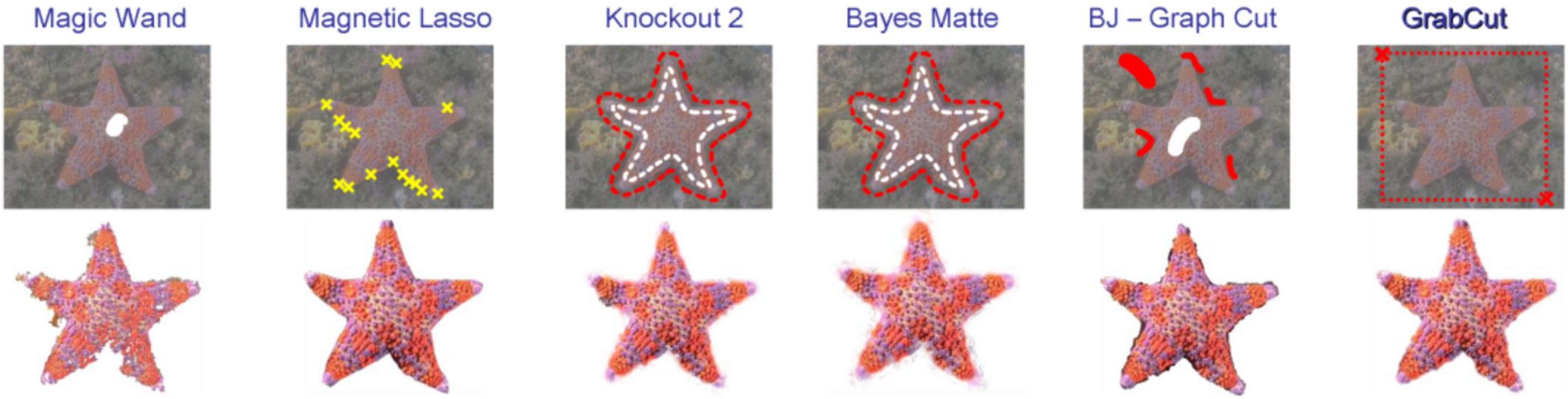


What do we do?

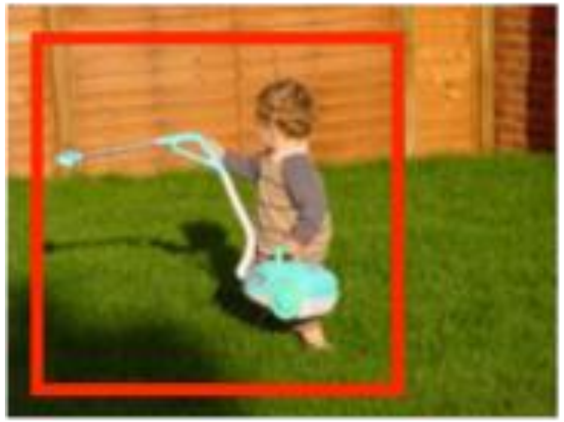
Iteration can be interactive



Examples


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

Examples

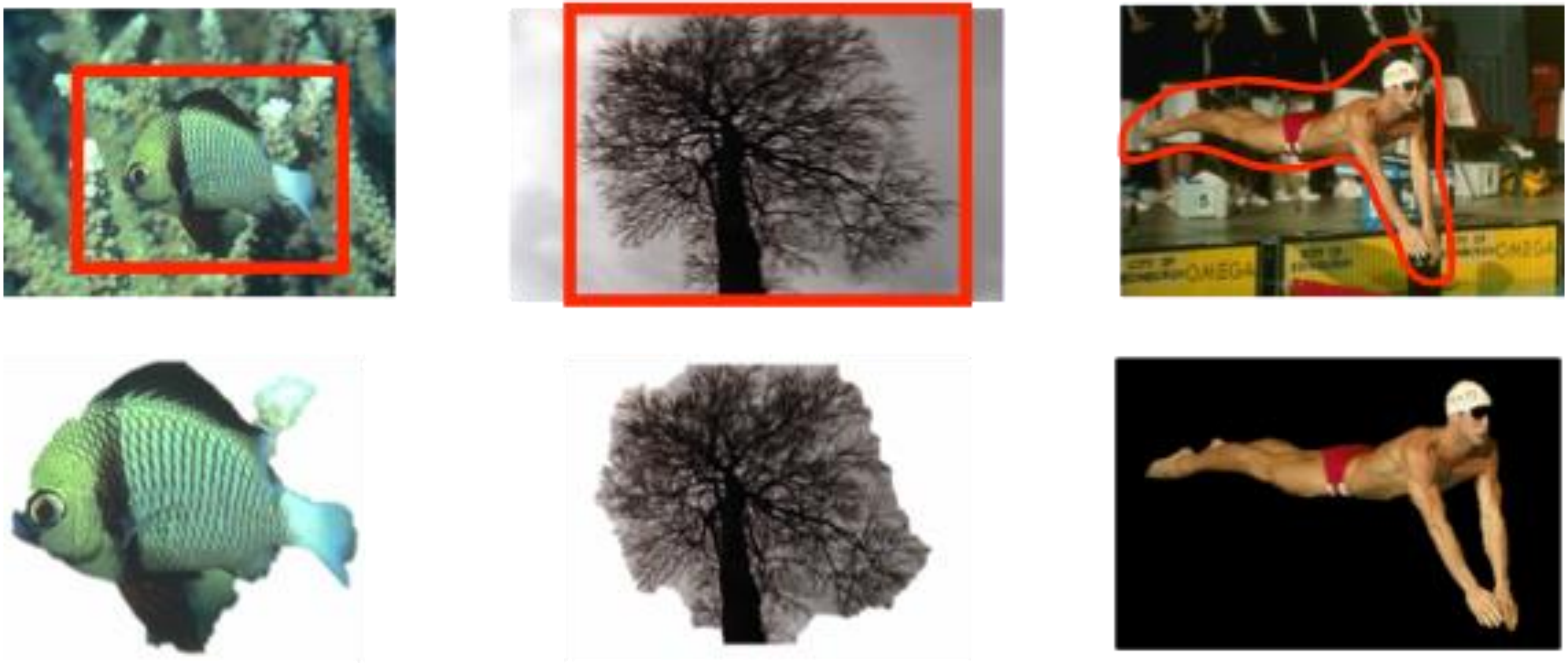





What is easy or hard about these cases for graph cut-based segmentation?

Examples

Examples



Examples

Lazy Snapping
[Li et al. SIGGRAPH 2004]

Graph-cuts are a very general, very useful tool

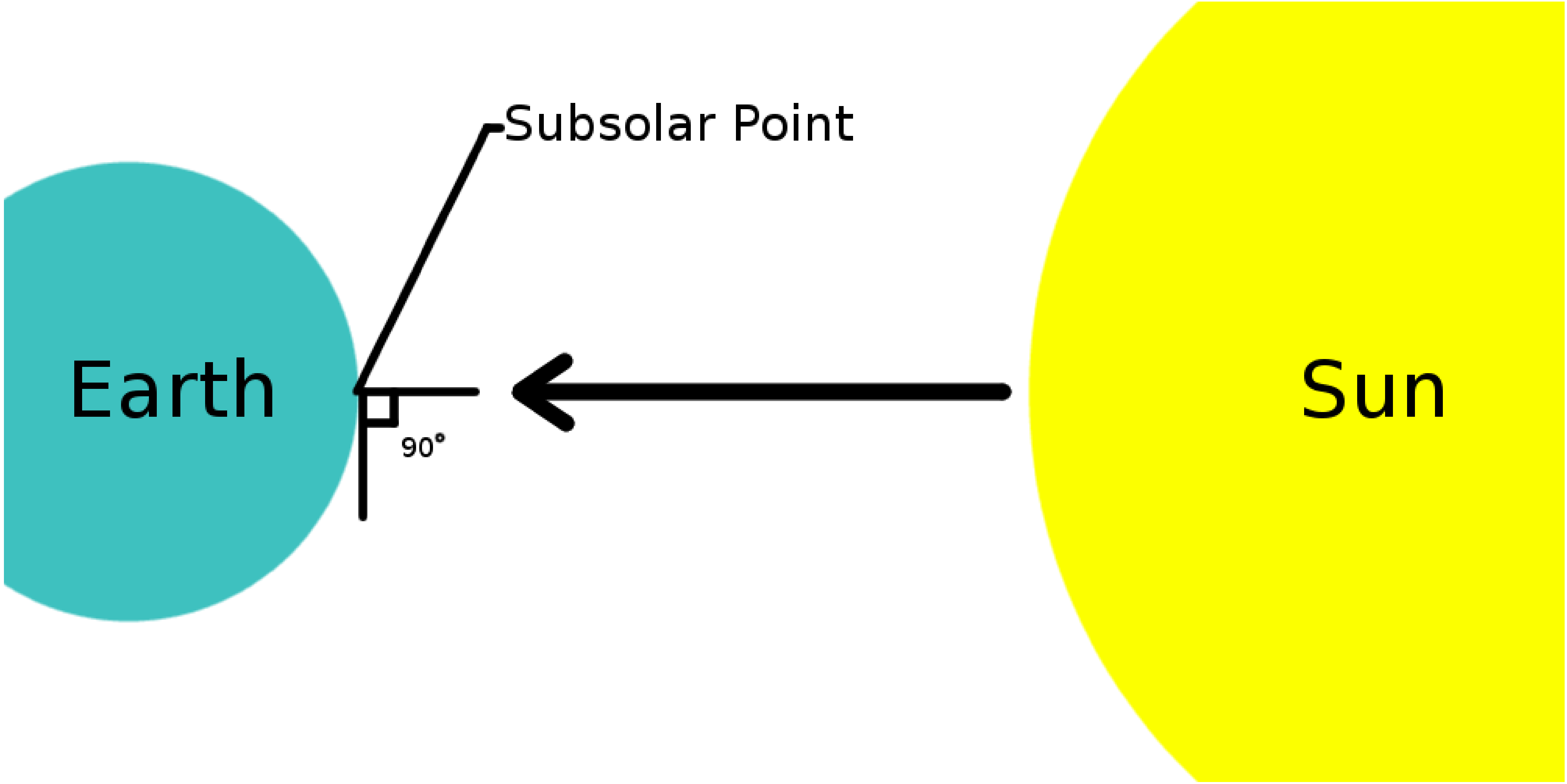
- denoising
- stereo
- texture synthesis
- segmentation
- classification
- recognition
- …

3D model of scene

Some notes about cutting-and-pasting

Real or composite, and why?

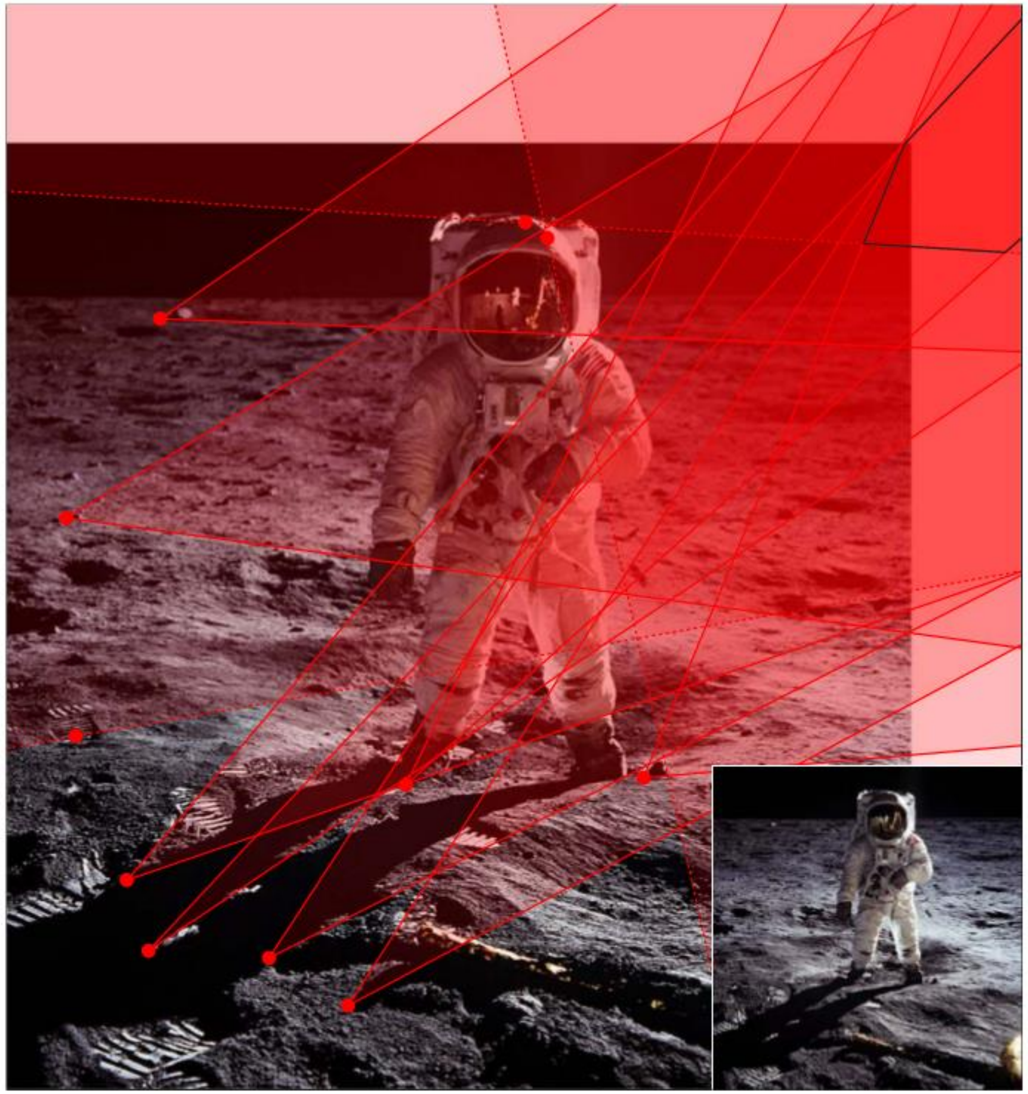
Real: Lahaina noon (or noon at subsolar point)



Real or composite, and why?

Composite: Inconsistent shadows

Composite: Inconsistent shadows



Original image copyright 1969, NASA

Fig. 1. Our algorithm finds that the shadows in this 1969 moon landing photo are physically consistent with a single light source. The solid lines correspond to constraints from cast shadows and dashed lines correspond to constraints from attached shadows. The region outlined in black, which extends beyond the figure boundary, contains the projected light locations that satisfy all of these constraints.

Kee et al., “Exposing Photo Manipulation with Inconsistent Shadows,” ToG 2014

Photorealistic compositing

Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

Photorealistic compositing

Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

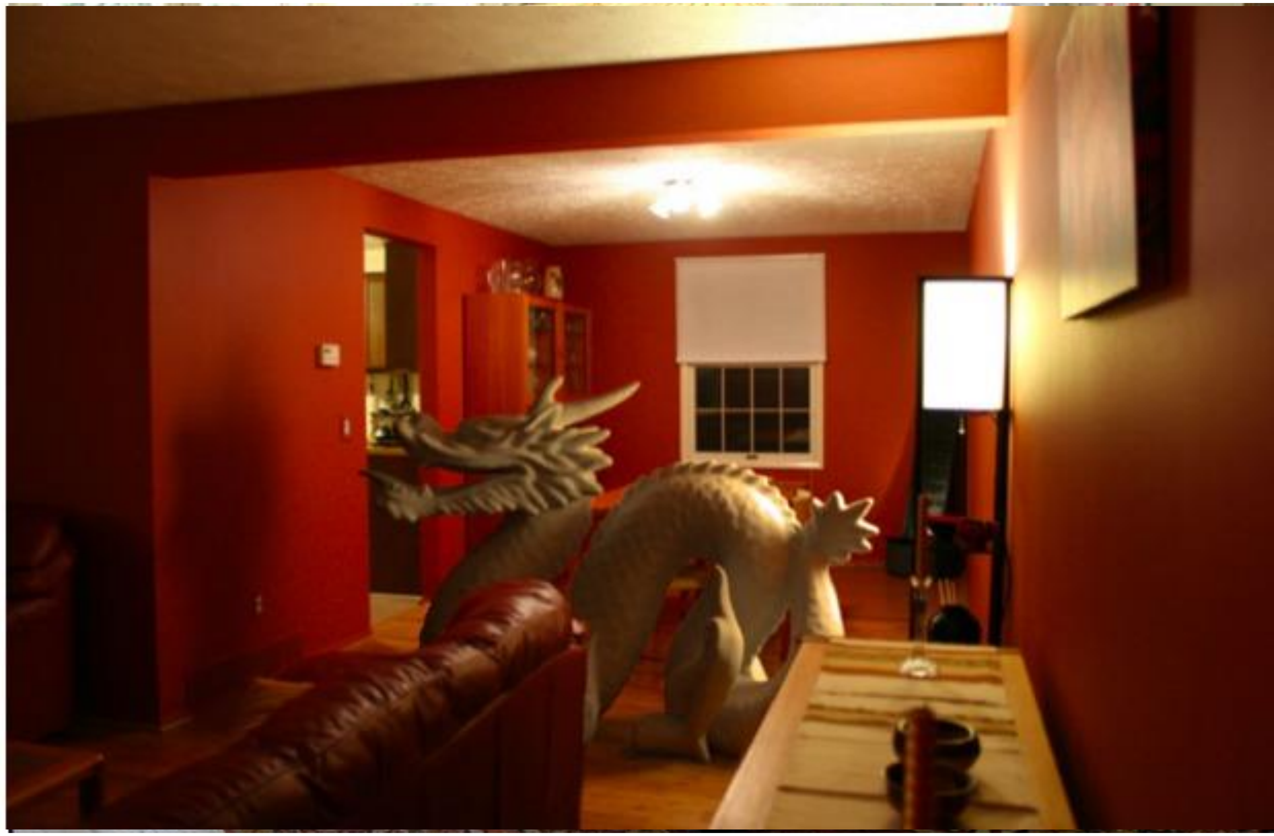
Photorealistic compositing

Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

Photorealistic compositing

Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

Photorealistic compositing



Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

Photorealistic compositing

Karsch et al., “Rendering Synthetic Objects into Legacy Photographs,” SIGGRAPH Asia 2011

Photorealistic compositing

How would you do this?

References

Basic reading:

- Szeliski textbook, Sections 5.1.3, 5.3.1, 9.3.2, 9.3.3, 10.4.3.
- Mortensen and Barrett, “Intelligent scissors for image composition,” SIGGRAPH 1995.
  the intelligent scissors paper.
- Kwatra et al., Graphcut Textures: Image and Video Synthesis using Graph Cuts, SIGGRAPH 2003.
  the seam collaging paper.
- Rother et al., “Interactive Foreground Extraction with Iterated Graph Cuts,” SIGGRAPH 2004.
  the GrabCut paper.
- Avidan and Shamir, “Seam Carving for Content-aware Image Resizing,” SIGGRAPH 2007.
  the seam carving paper.

Additional reading:

- Li et al., “Lazy Snapping,” SIGGRAPH 2004.
  a popular variant of GrabCut.
- Felzenszwalb and Zabih, “Dynamic Programming and Graph Algorithms in Computer Vision,” PAMI 2010.
  a great review of graph-based techniques, including shortest path and graph-cut, in computer vision.
- Kee et al., “Exposing photo manipulation with inconsistent shadows,” ToG 2013.
  the paper demonstrating how image forgeries can be detecting by reasoning about the physical accuracy of shadows in an image.
- Karsch et al., “Rendering synthetic objects into legacy photographs”, SIGGRAPH 2011.
  the paper where the photorealistic compositing examples came from.