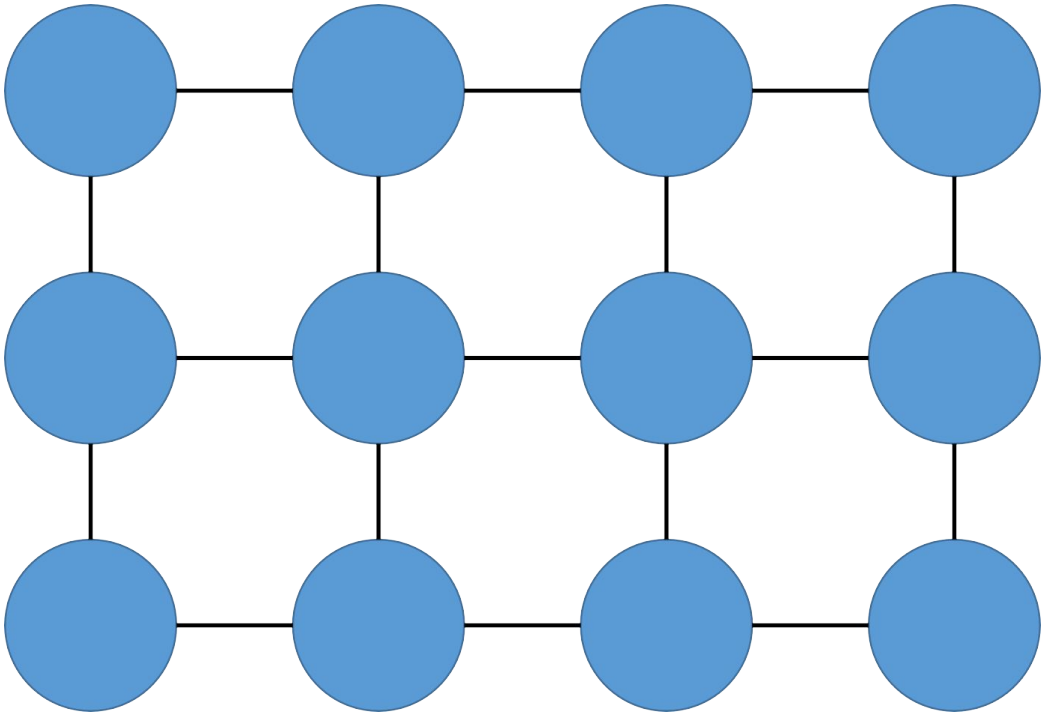
Graph Cuts

Fundamental theme of today's tutorial

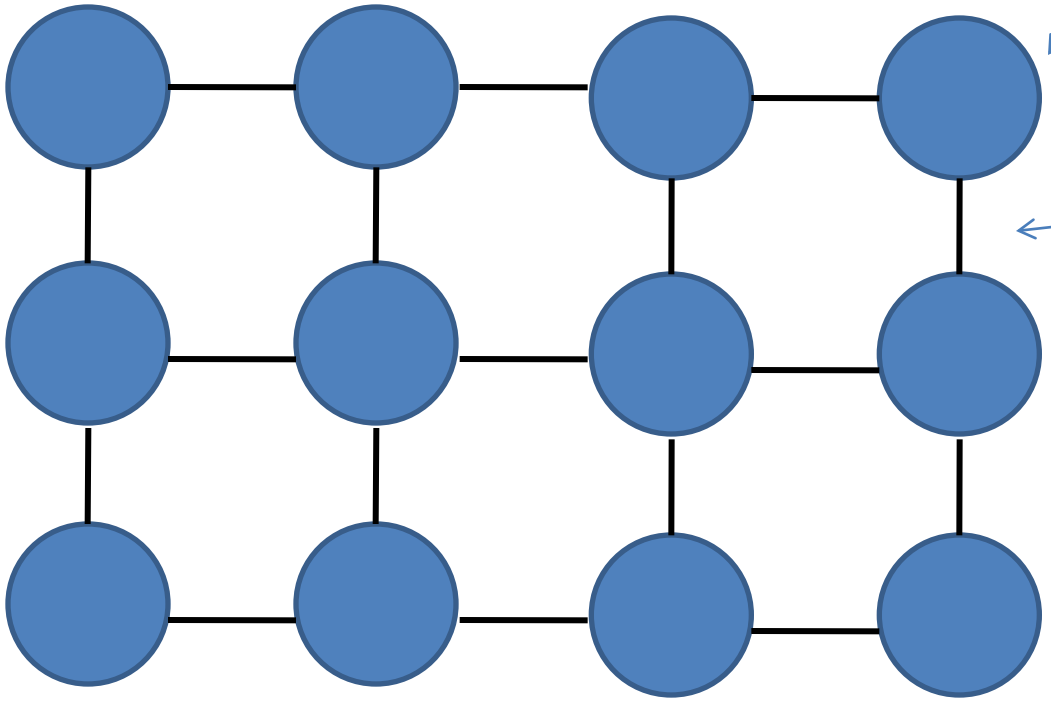
Images can be viewed as graphs



Nodes: pixels

Edges: Constraints between neighboring pixels

Markov Random Fields



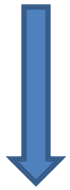
Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

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

Writing Likelihood as an "Energy"

$$P(\mathbf{y};\theta,data) = \frac{1}{Z}\prod_{i=1..N} p_1(y_i;\theta,data)\prod_{i,j\in edges} p_2(y_i,y_j;\theta,data)$$



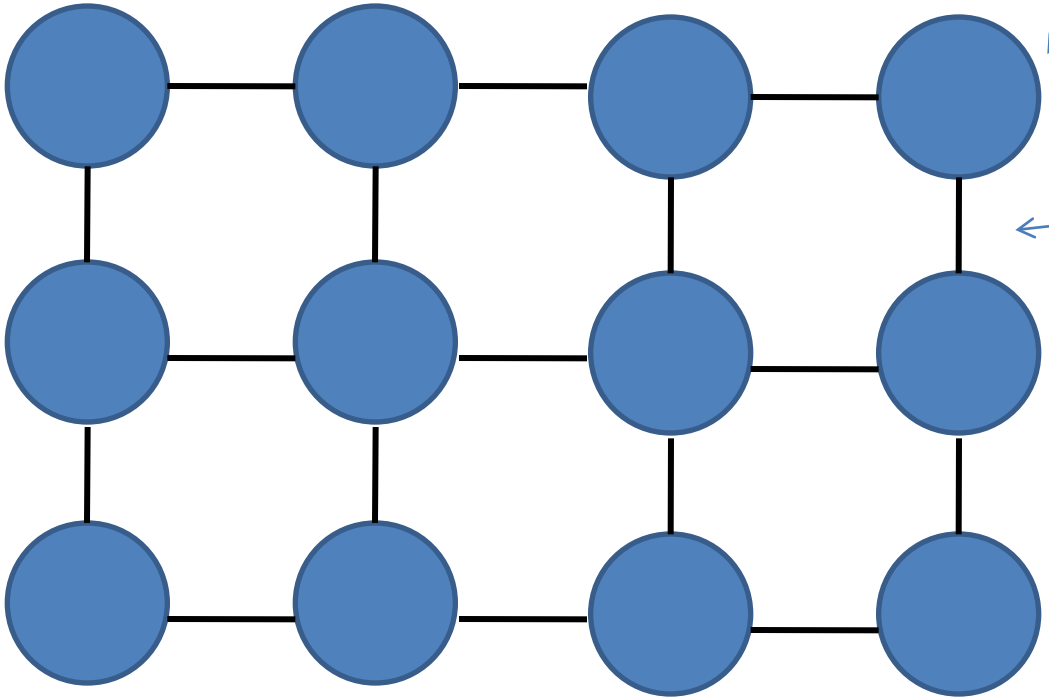
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

"Cost" of assignment y_i

"Cost" of pairwise assignment y_i , y_j

Markov Random Fields

- Example: “label smoothing” grid



Unary potential

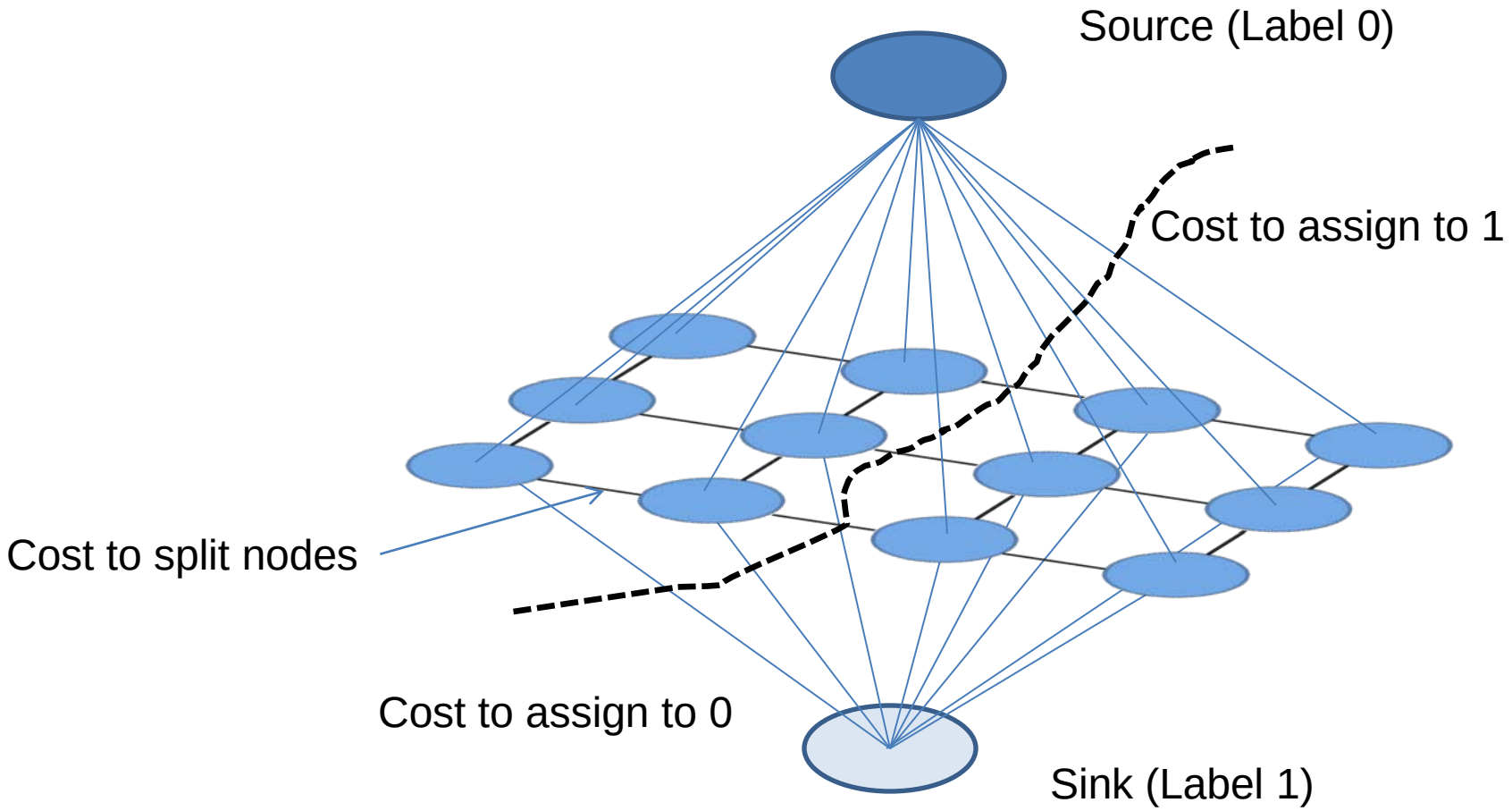
0: -logP(y_i = 0 ; data)
1: -logP(y_i = 1 ; data)

Pairwise Potential

	0	1
0	0	K
1	K	0

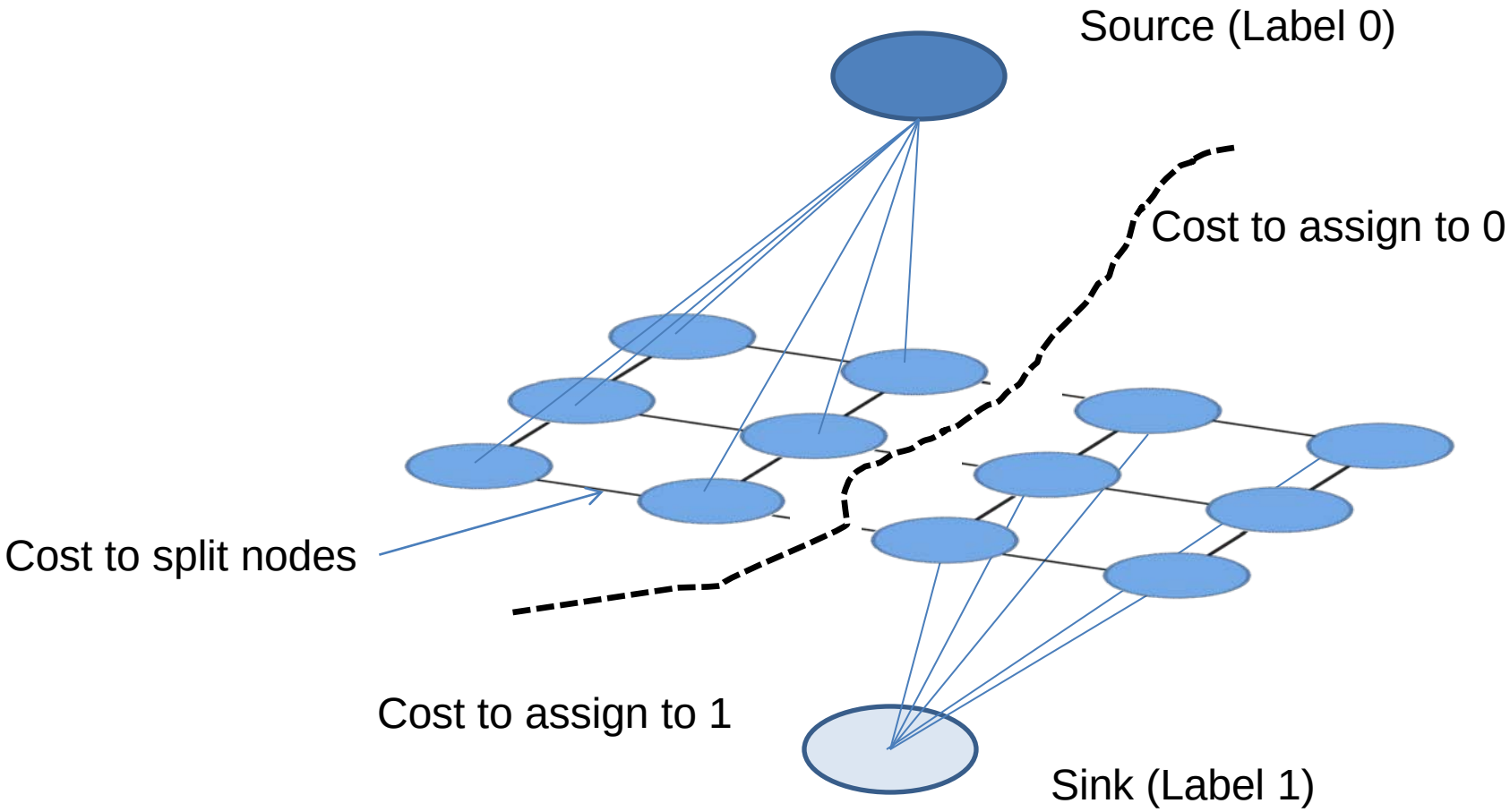
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Solving MRFs with 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 with graph cuts



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

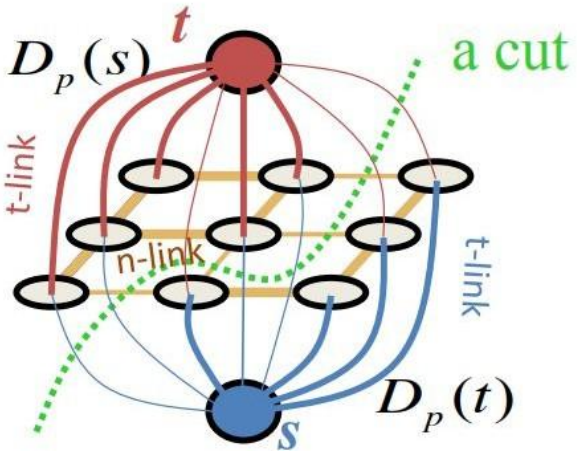
Generalized Potts Model

$$E(f) = \sum_{p} D_p(f_p) + \sum_{\{p,q\}} u_{\{p,q\}} \cdot \delta(f_p \neq f_q)$$

t-links

n-links

$f_p \in \{s,t\}$



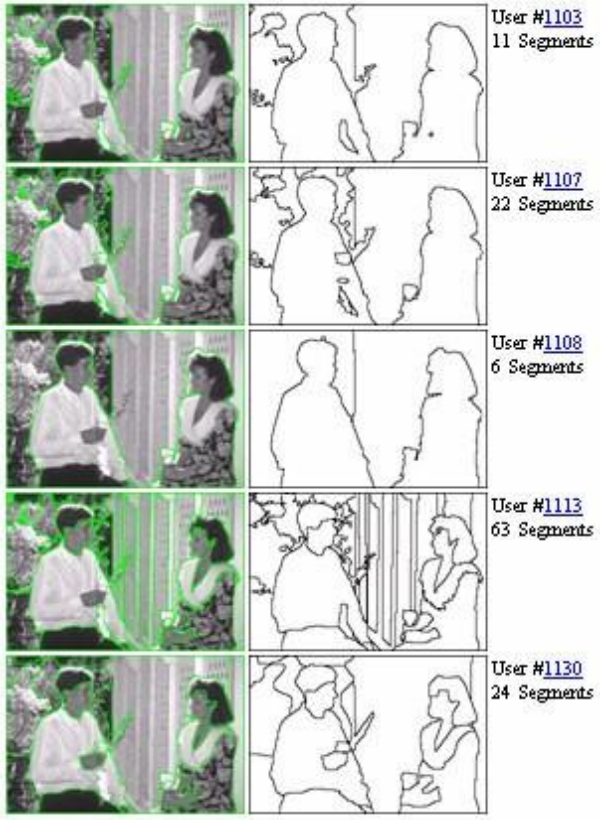
A s-t cut corresponds to one configuration of binary labeling

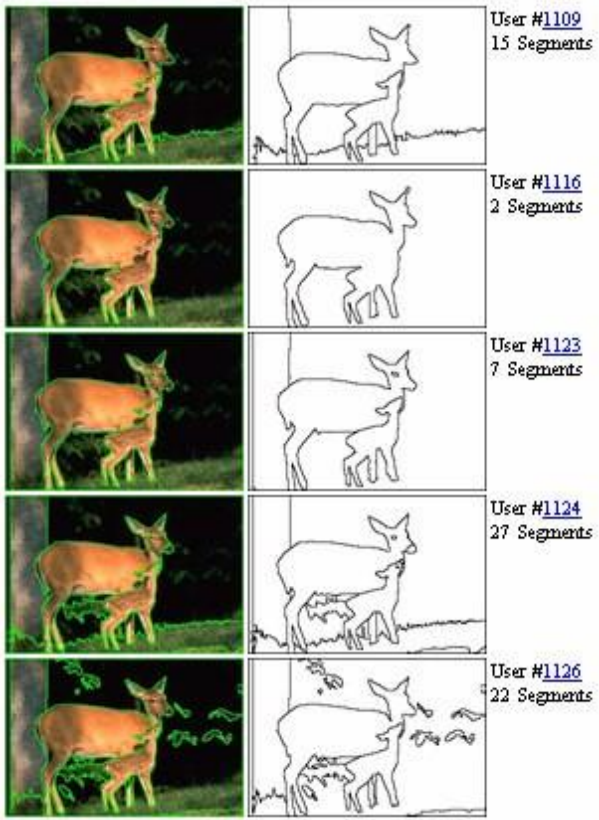
$$C(S,T) = E(f)$$

Segmentation

What is a “good” segmentation??

No objective definition of segmentation!

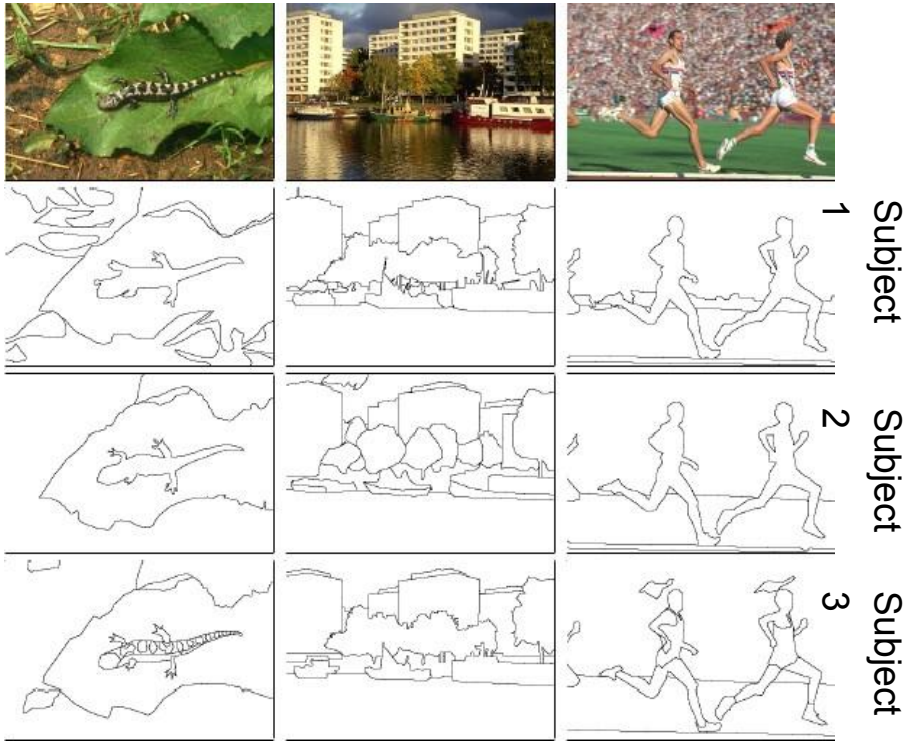




http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/color/317080.html

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

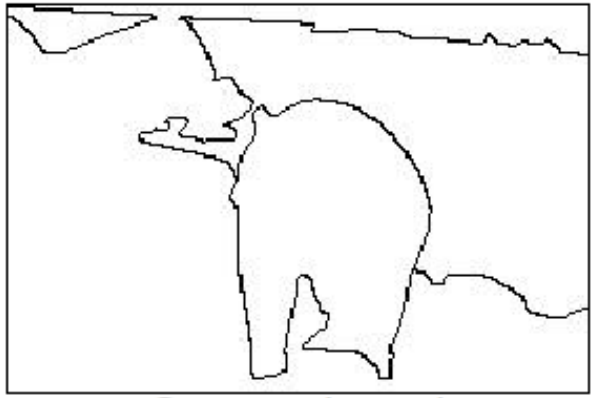
No objective definition of segmentation!



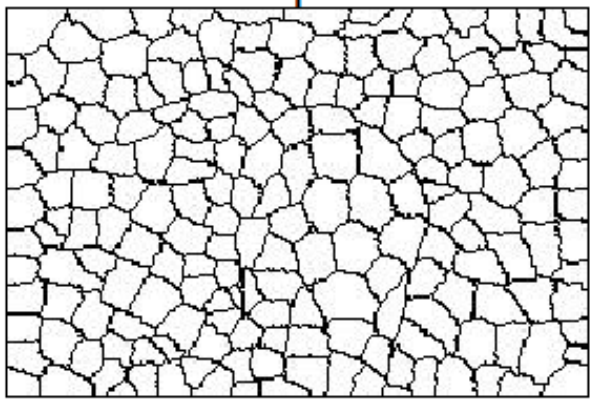
- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Second idea: Superpixels

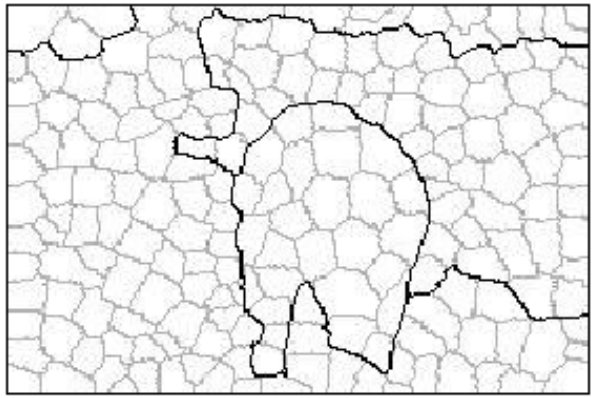
Input



Ground truth



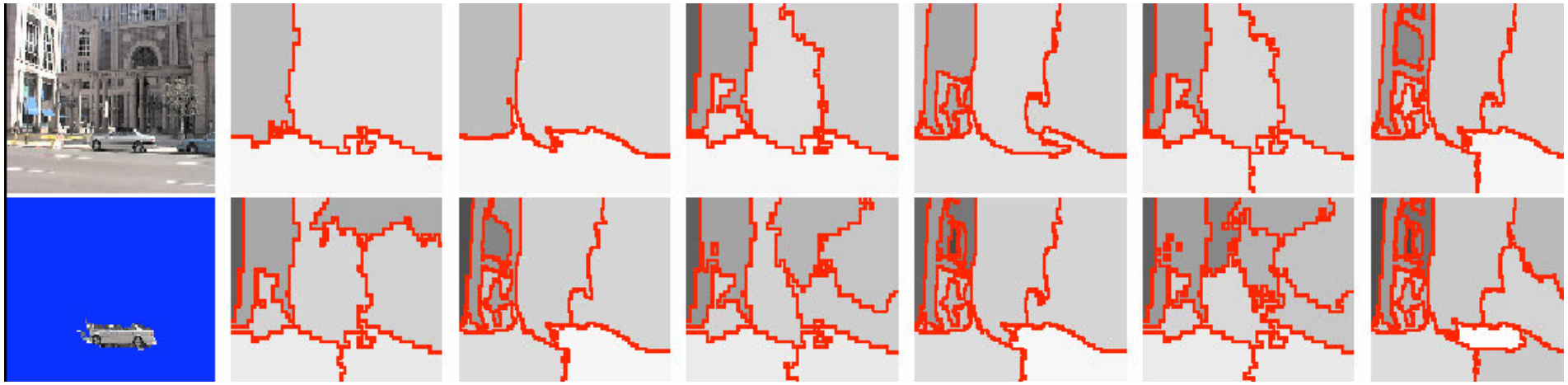
Superpixels



Overlay

- 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

Third idea:
Multiple segmentations

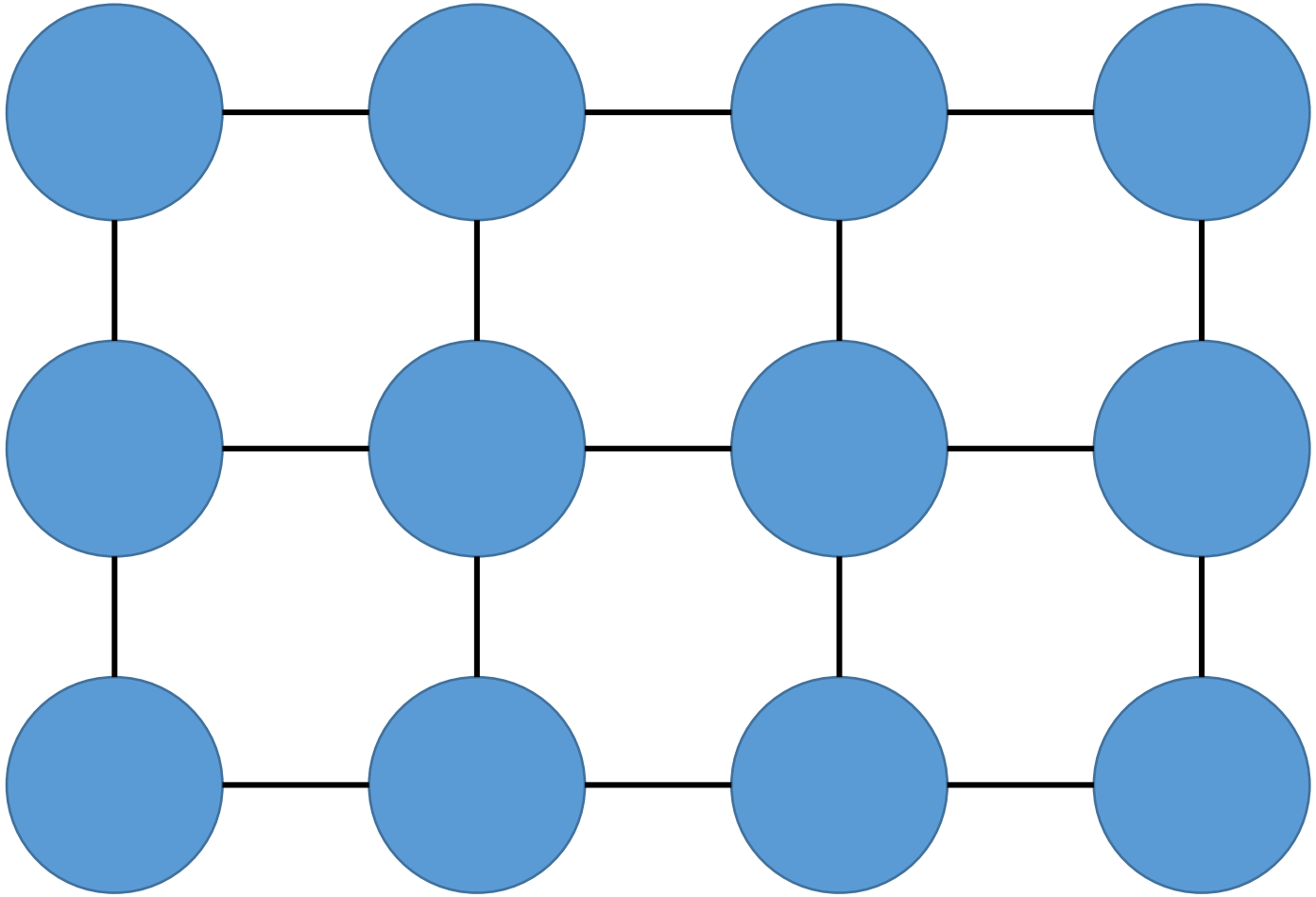


- Generate many segmentations of the same image
- Even though many regions are “wrong”, some consensus should emerge

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

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

GrabCut

Only user input is the box!

grab cut paste

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

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

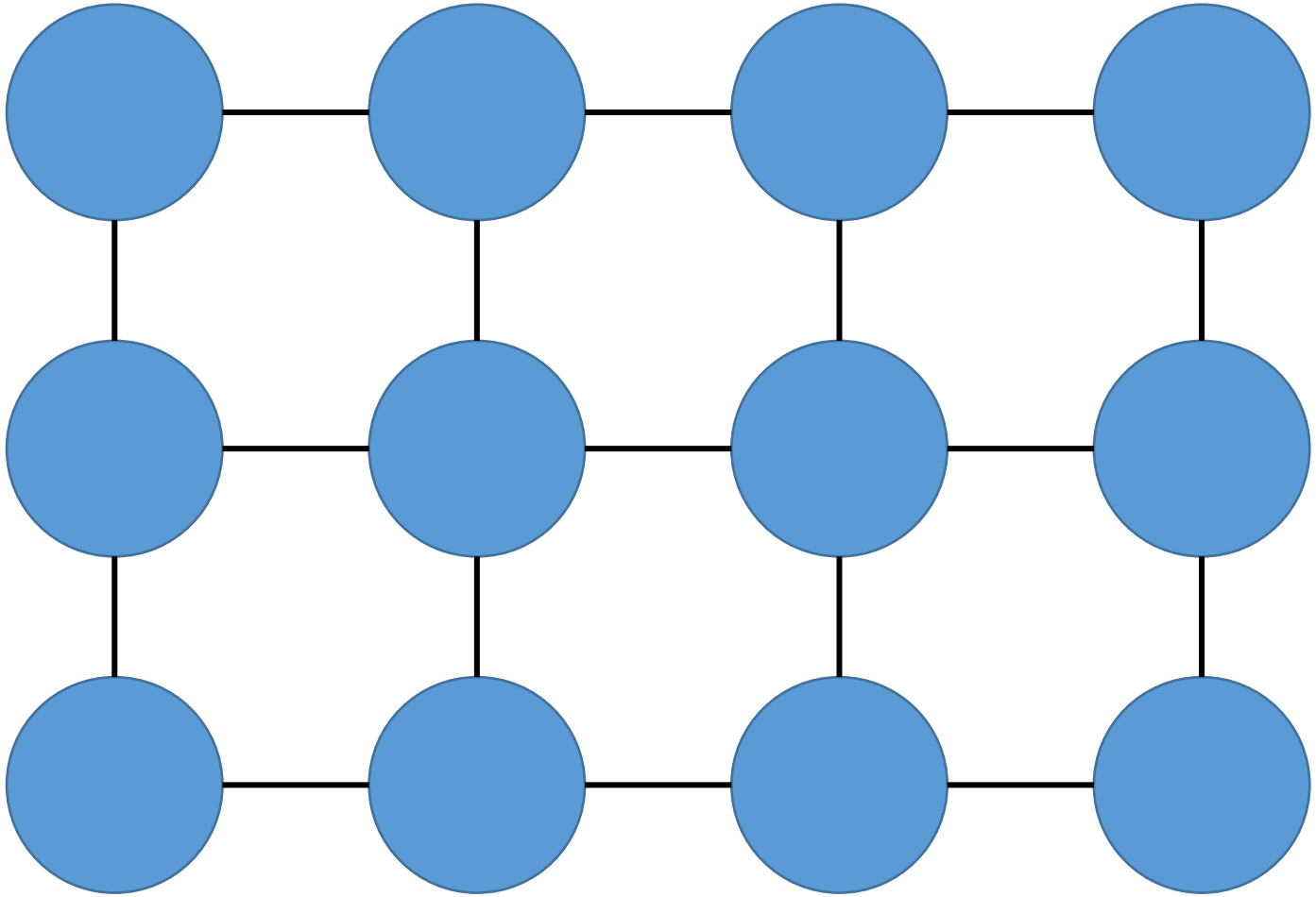
1. Segmentation using graph cuts

2. Foreground-background modeling using unsupervised clustering

Markov Random Field (MRF)

Assign foreground/background labels based on:

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

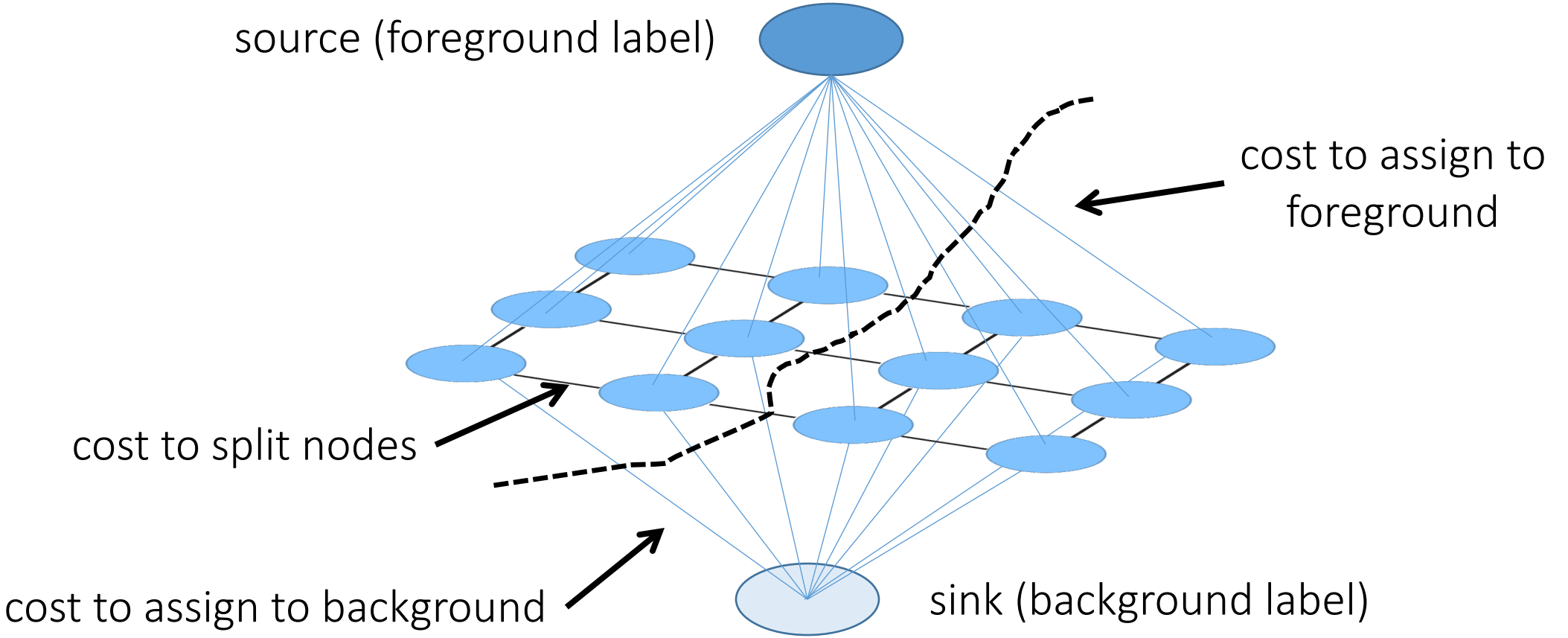


Given its intensity value, how likely is a pixel to be foreground or background?

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

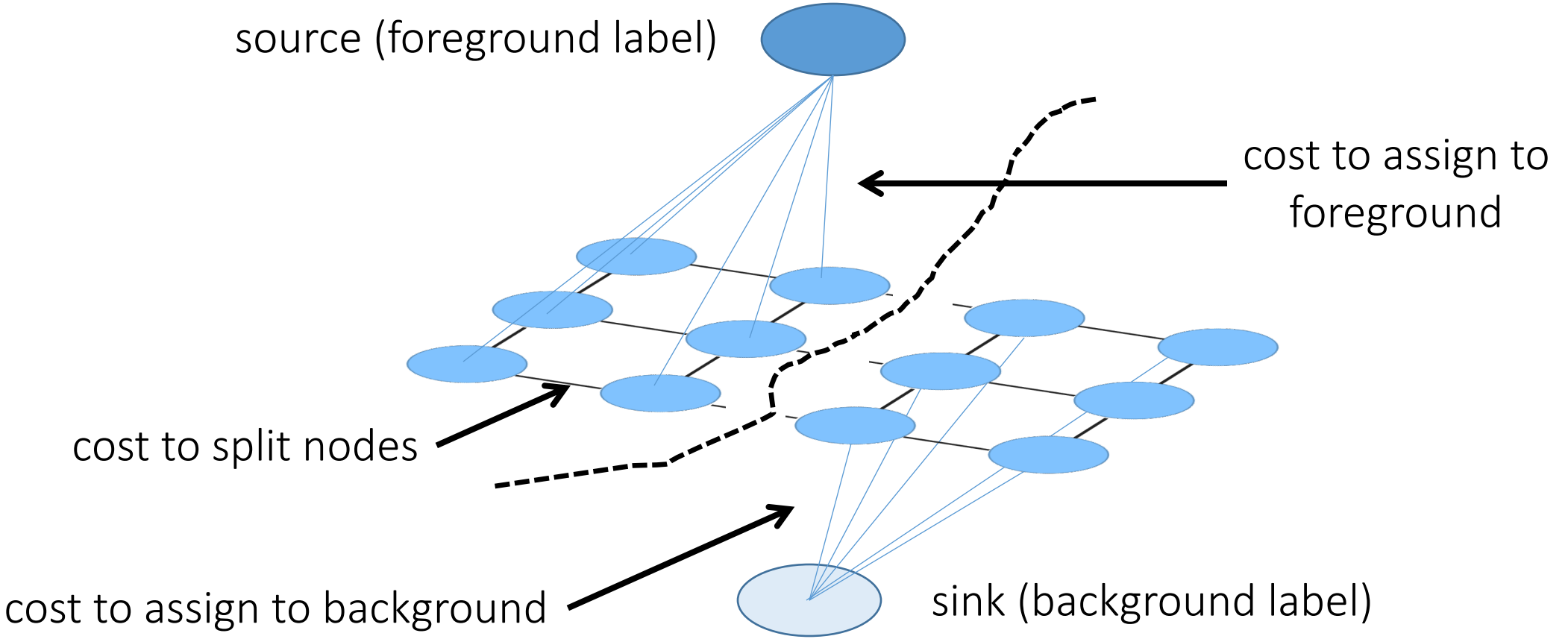
What kind of cost functions would you use for GrabCut?

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)



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

GrabCut is a mixture of two components

1. Segmentation using graph cuts

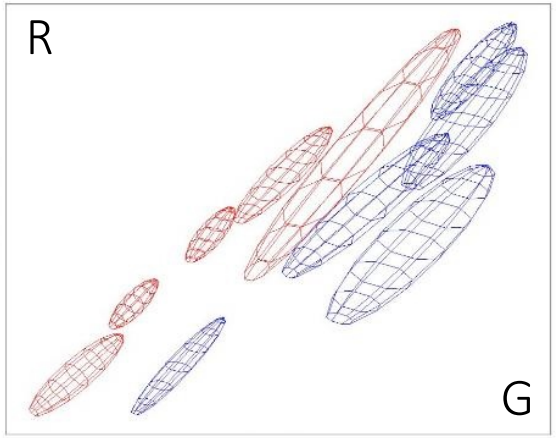
2. Foreground-background modeling using unsupervised clustering

Foreground-background modeling

Given foreground/background labels

build a color model for both

foreground

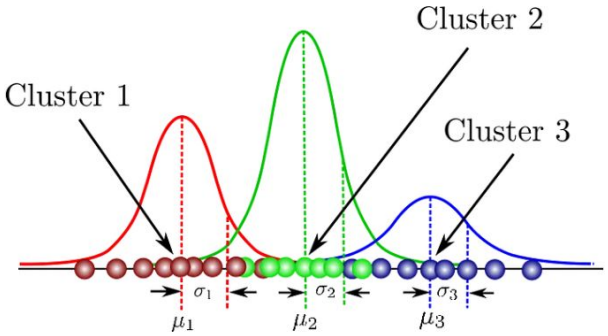


background

Gaussian Mixture Model

A *Gaussian Mixture* is a function that is comprised of several Gaussians, each identified by $k \in \{1,\ldots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

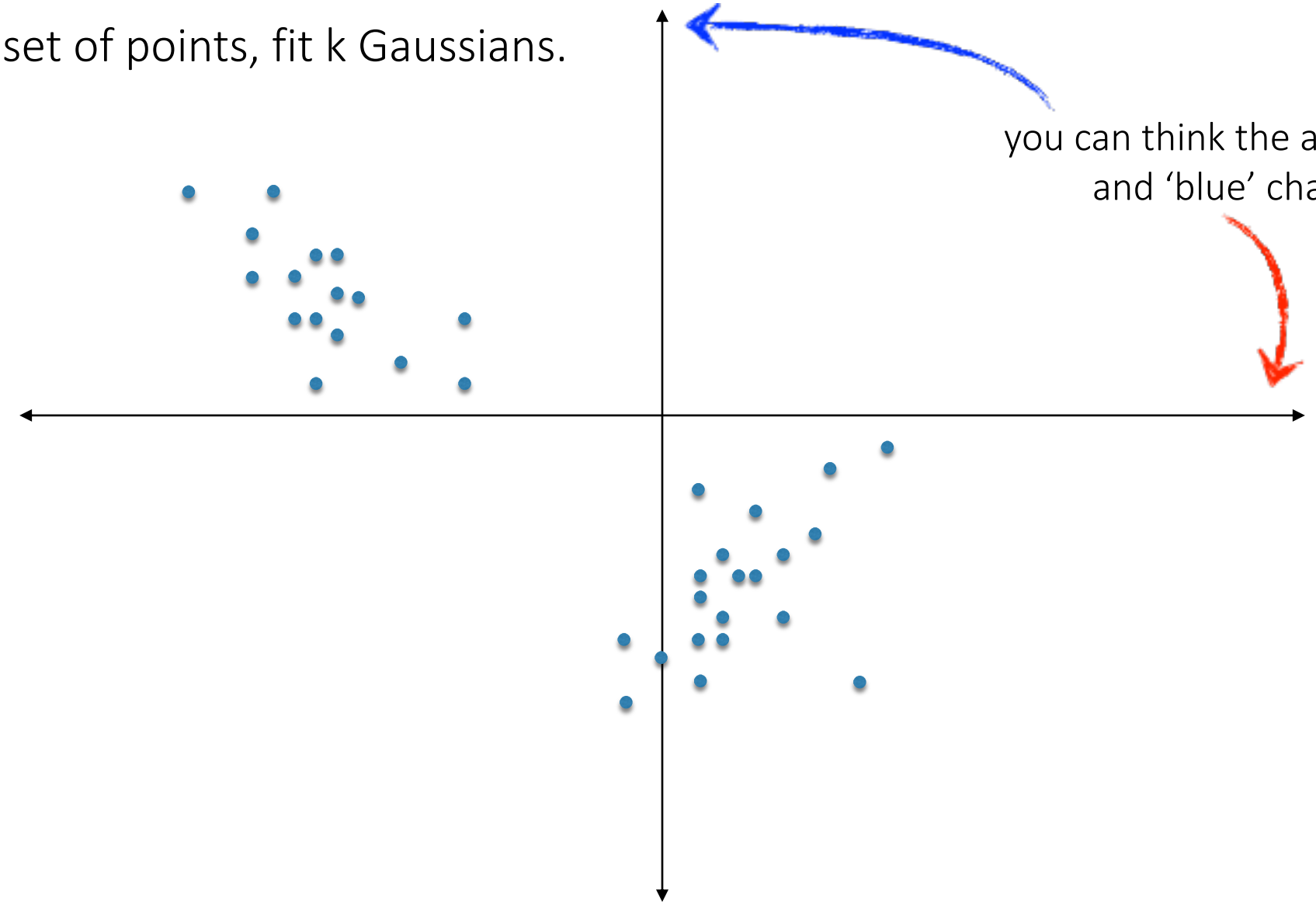
- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.



$$\sum_{k=1}^{K} \pi_k = 1$$

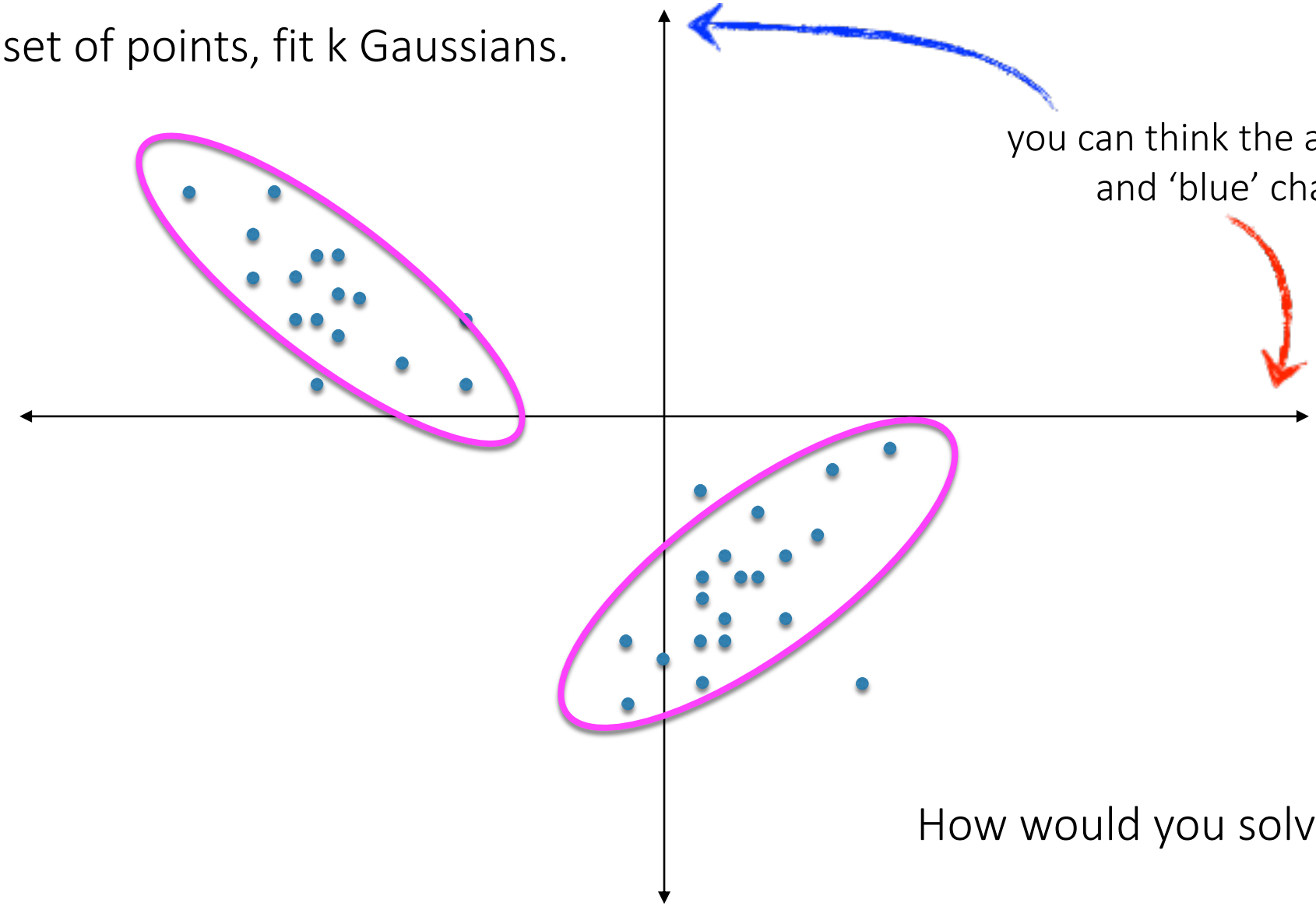
Learning color models

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

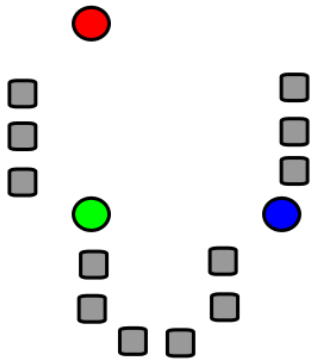
How would you solve this problem?

Intuition: “hard” clustering using K-means

Given k:

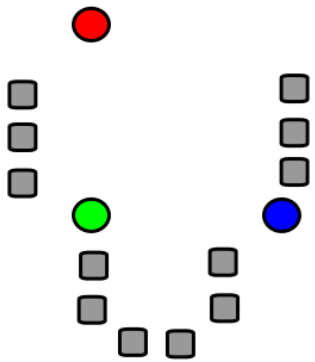
1. Select initial centroids at random.
2. Assign each object to the cluster with the nearest centroid.
3. Compute each centroid as the mean of the objects assigned to it.
4. Repeat previous 2 steps until no change.

K-means visualization

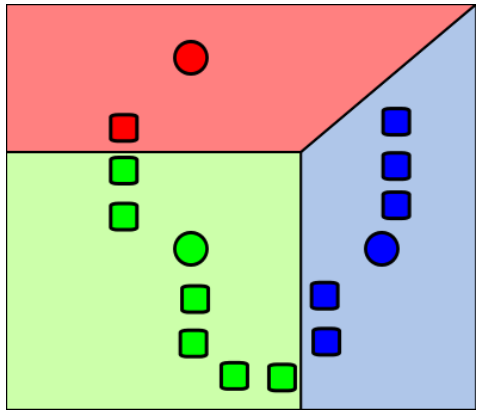


1. Select initial centroids at random

K-means visualization

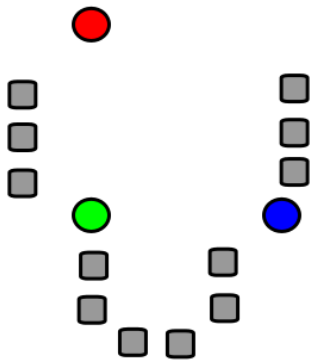


1. Select initial centroids at random

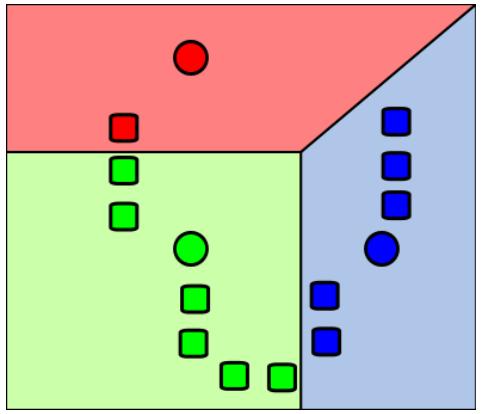


2. Assign each object to the cluster with the nearest centroid.

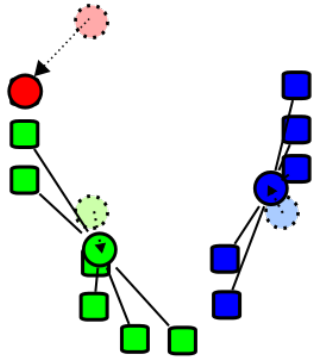
K-means visualization



1. Select initial centroids at random

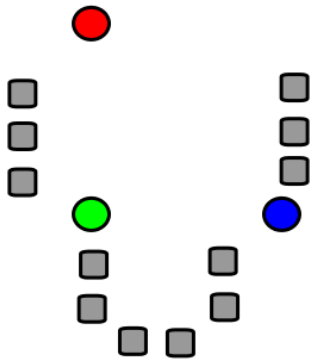


2. Assign each object to the cluster with the nearest centroid.

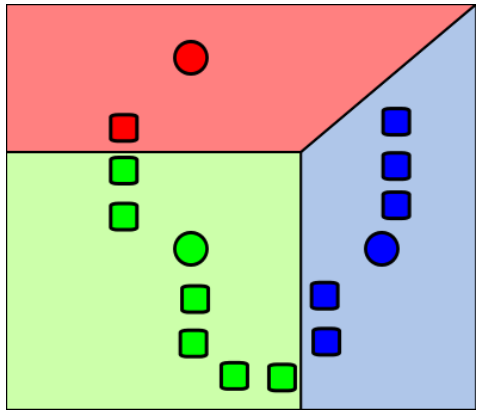


3. Compute each centroid as the mean of the objects assigned to it (go to 2)

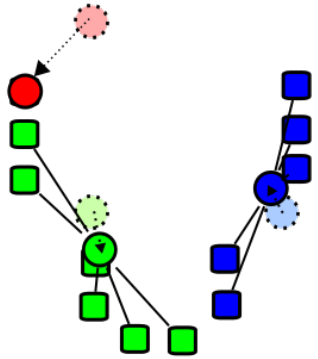
K-means visualization



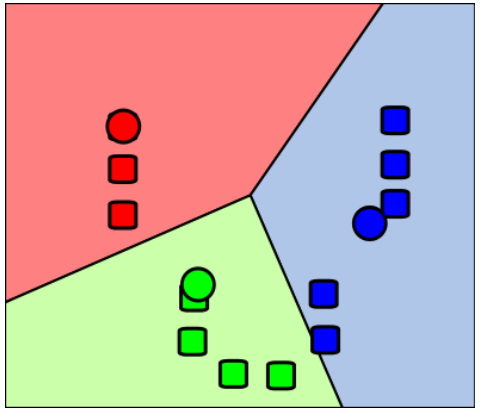
1. Select initial centroids at random



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

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

Given k:

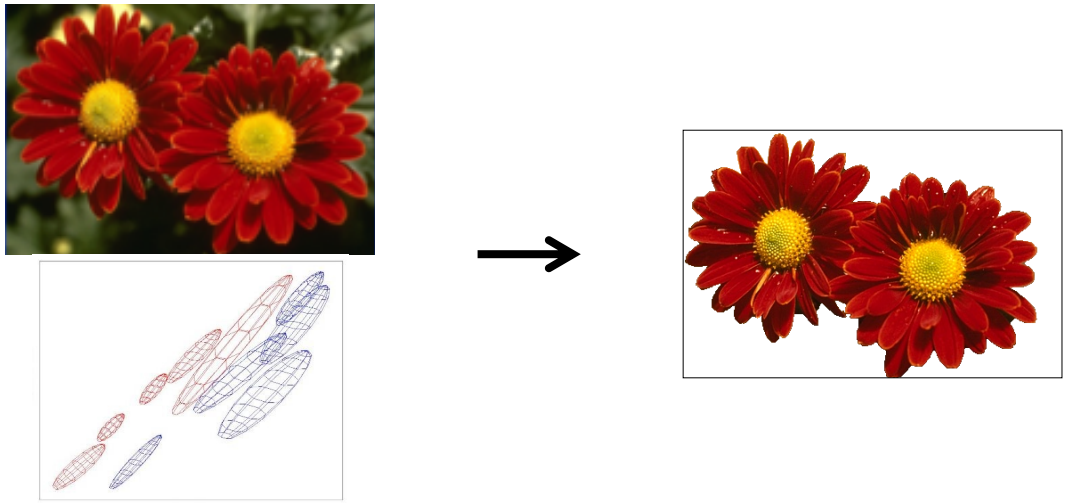
1. Select initial centroids at random.
2. ~~Assign each object to the cluster with the nearest centroid.~~ compute the probability of each object being in a cluster
3. Compute each centroid ^ as the mean and covariance of the objects ~~assigned~~ weighed by the probability of being in that cluster to it.
4. Repeat previous 2 steps until no change.

E-step

M-step

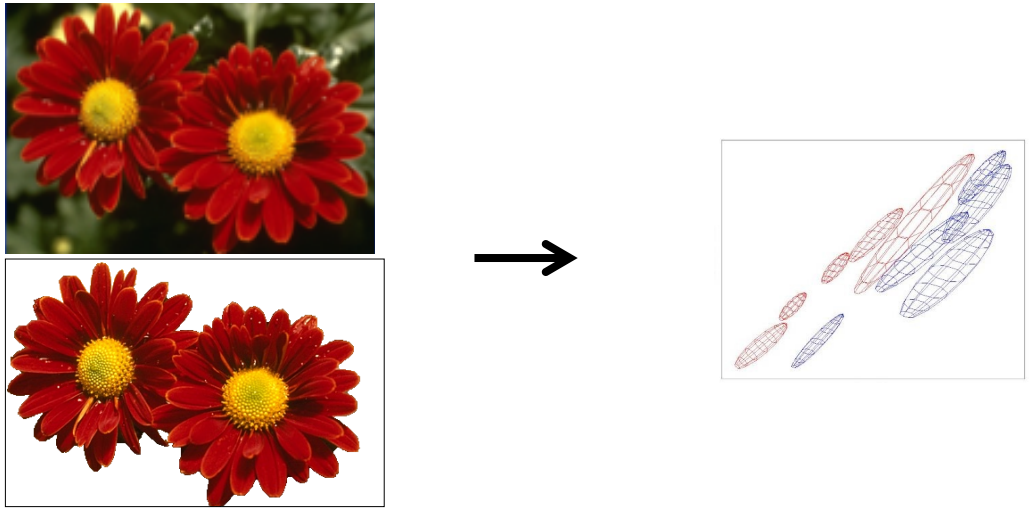
GrabCut is a mixture of two components

1. Segmentation using graph cuts
   - Requires having foreground model



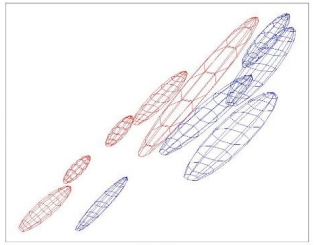
2. Foreground-background modeling using unsupervised clustering
   - Requires having segmentation

What do we do?



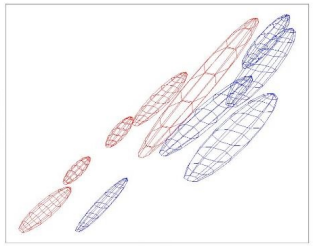
GrabCut: iterate between two steps

1. Segmentation using graph cuts
   - Requires having foreground model

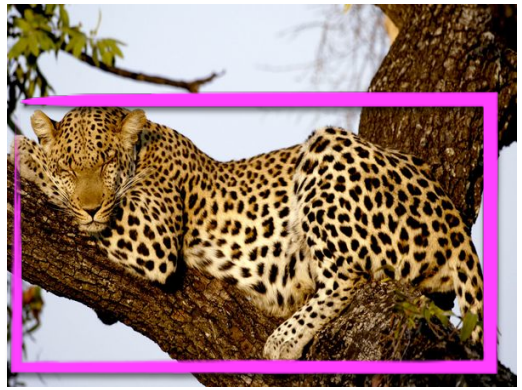


2. Foreground-background modeling using unsupervised clustering
   - Requires having segmentation

What do we do?

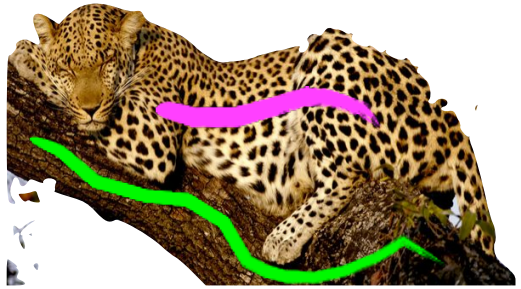


Iteration can be interactive



user specified box

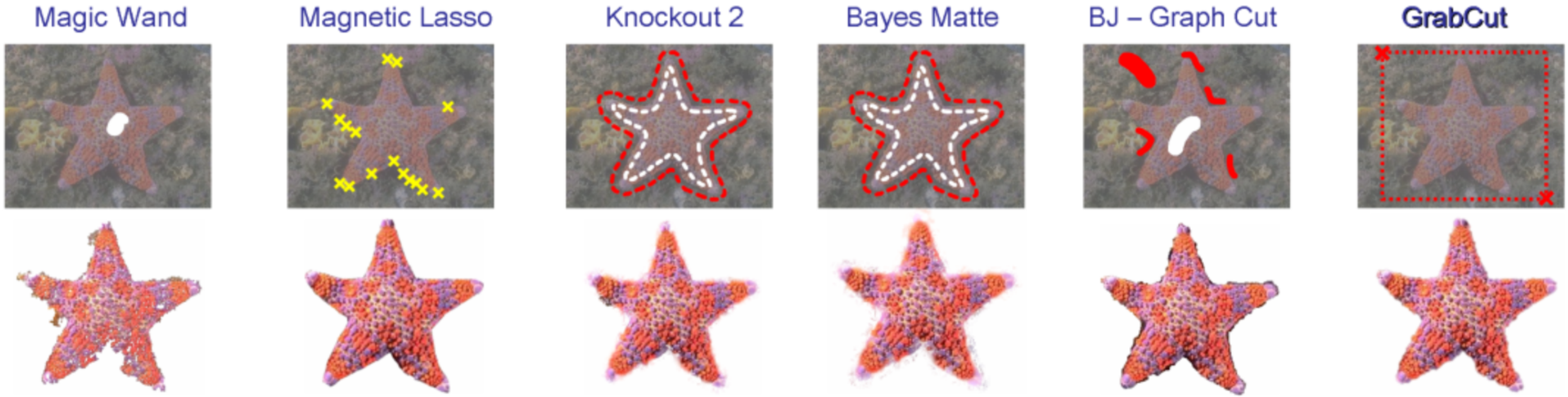
user edit



iterated graph cut

output

Examples

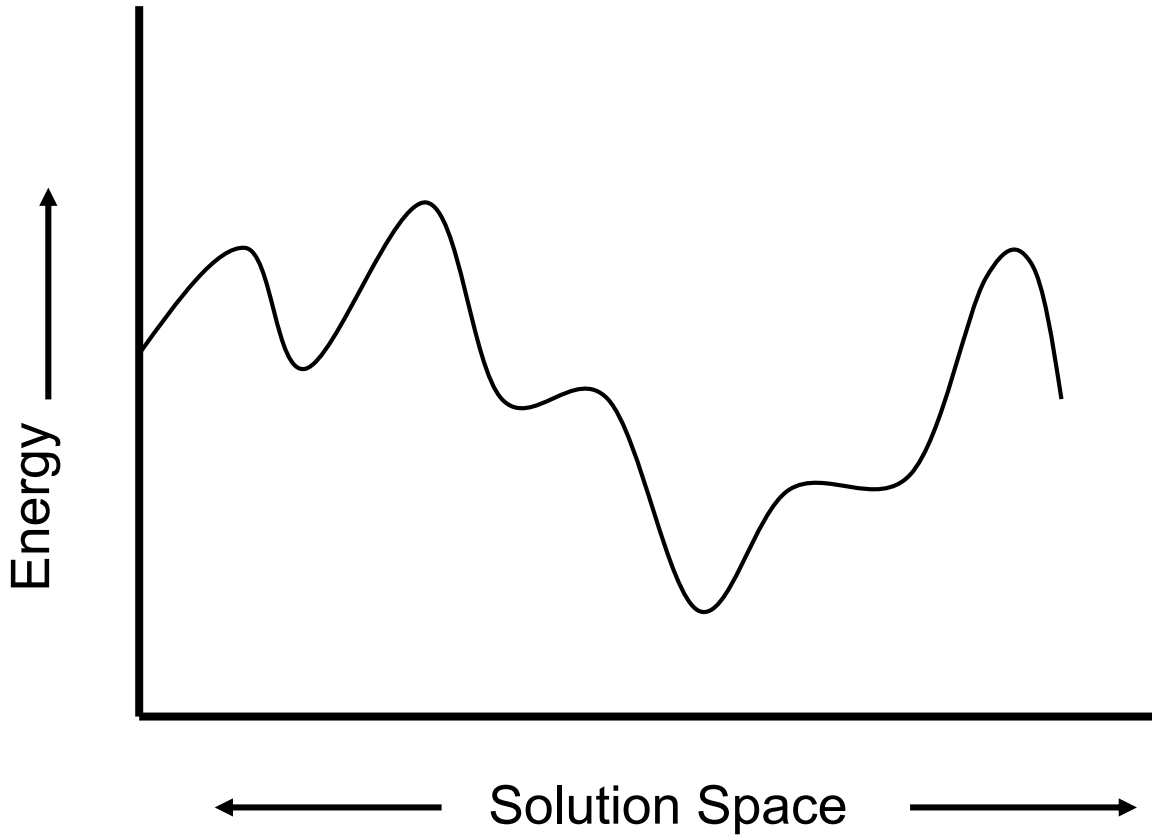

Magic Wand
Magnetic Lasso
Knockout 2
Bayes Matte
BJ – Graph Cut
GrabCut

Multi Label Problems

Approximate Methods:

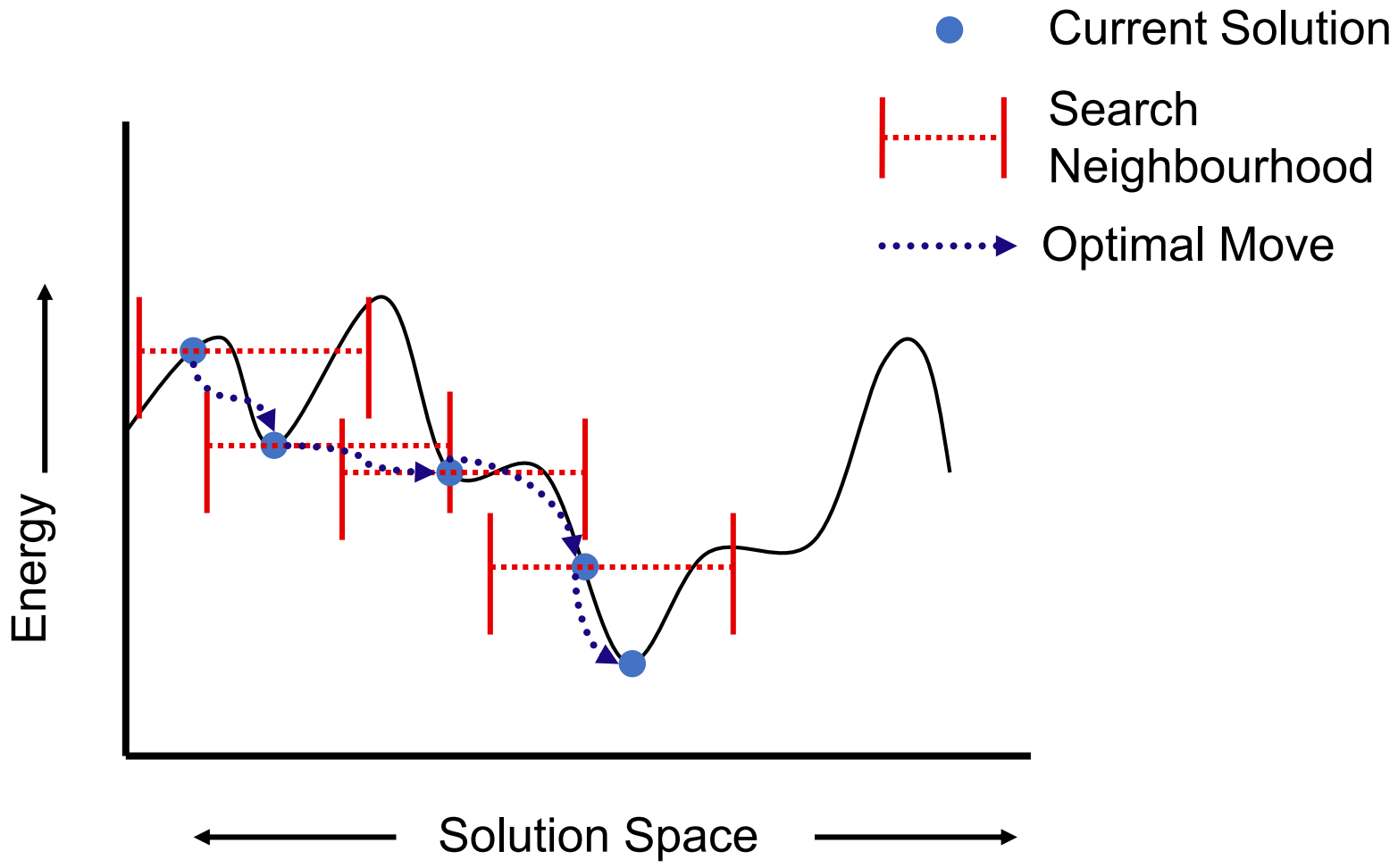
Develop iterative move-making algorithms where each move corresponds to a Boolean problem.

Move Making Algorithms



[Image courtesy: Pushmeet Kohli, Phil Torr]

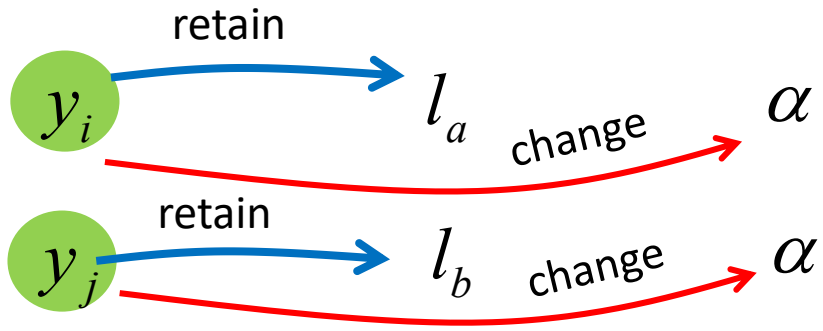
Move Making Algorithms



[Image courtesy: Pushmeet Kohli, Phil Torr]

$\alpha-$ Expansion

- Let y_i and y_j be two adjacent variables whose labels are not α.



In the move space, we compute if the two variables should retain the same labels or move to label α.

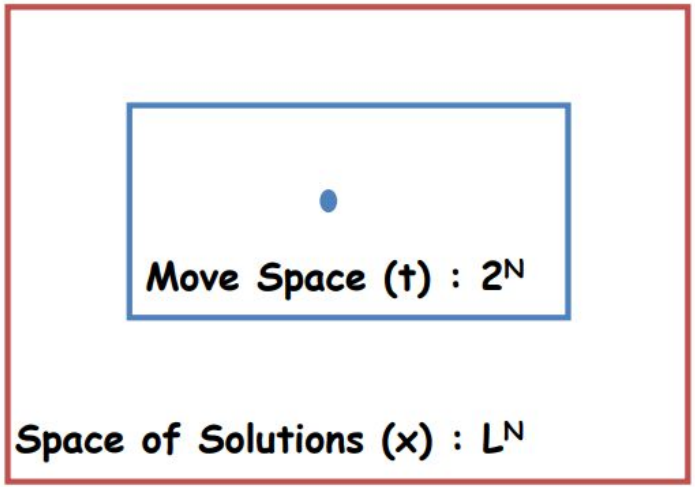
[Boykov et al. 2001]

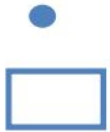
Moves using Graph Cuts

Expansion and Swap move algorithms

[Boykov Veksler and Zabih, PAMI 2001]

- **Makes a series of changes to the solution (moves)**
- **Each move results in a solution with smaller energy**

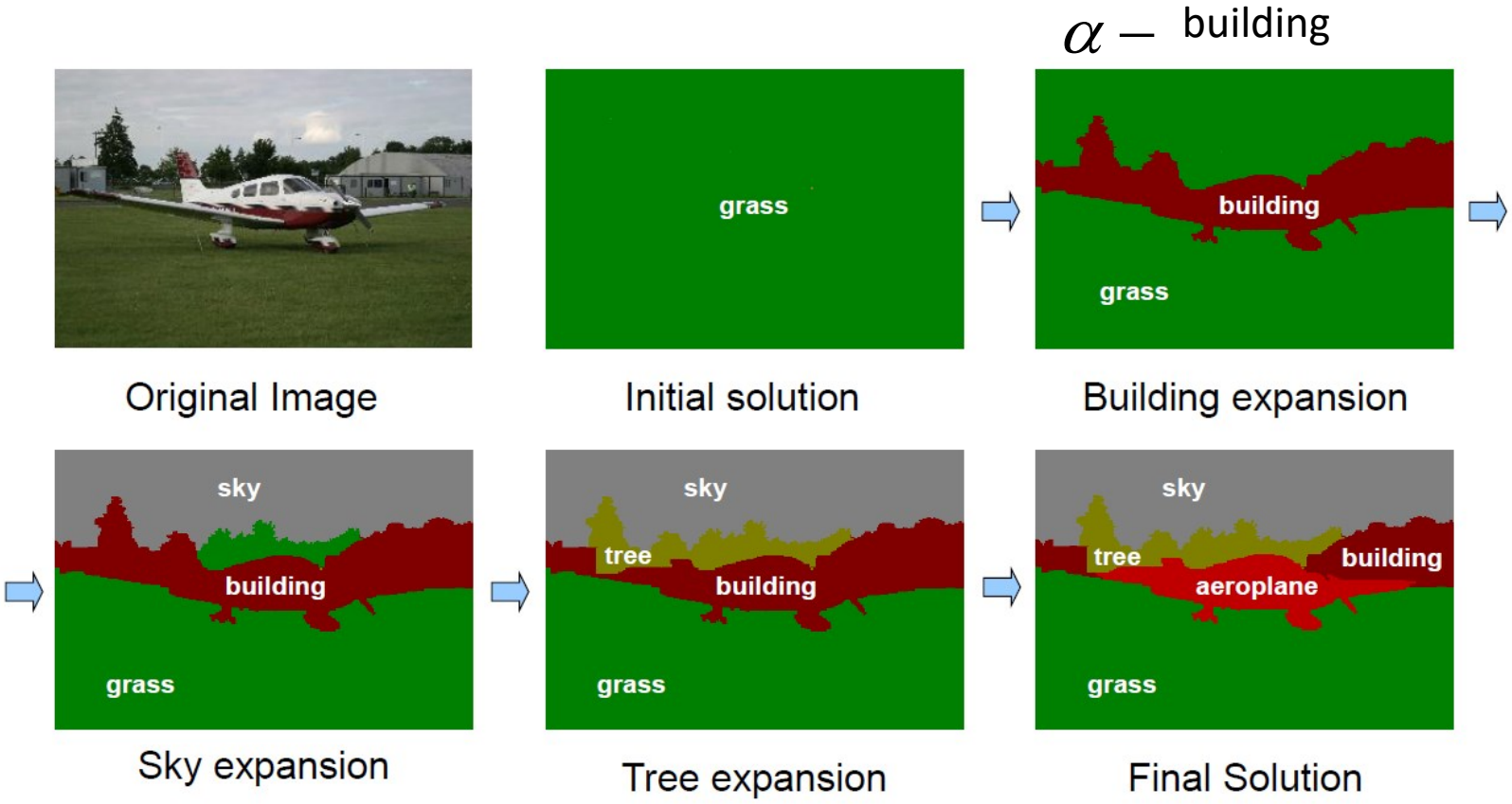




N Number of Variables

L Number of Labels

$\alpha-$ Expansion



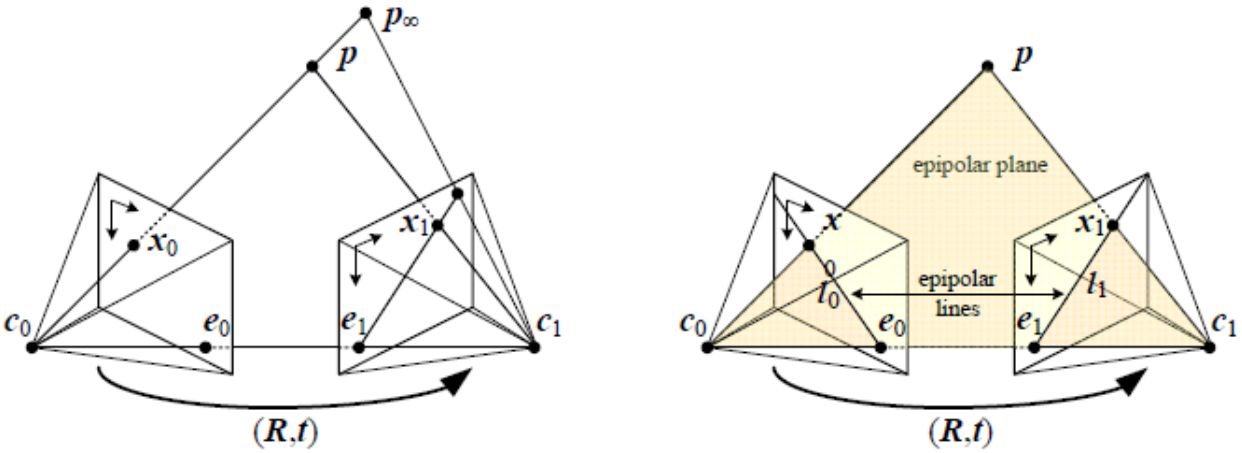
[Boykov et al. 2001]

Stereo Matching

as a Multi Labelling Problem

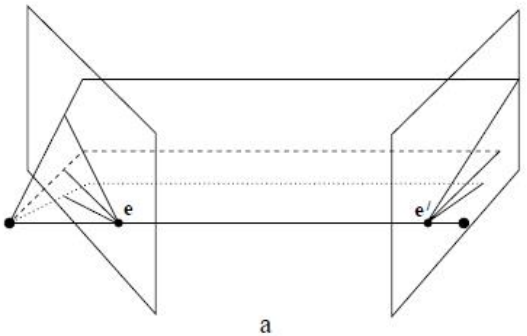
Stereo Matching

- For pixel x_0 in one image, where is the corresponding point x_1 in another image?
  - Stereo: two or more input views
- Based on the epipolar geometry, corresponding points lie on the epipolar lines
  - A matching problem



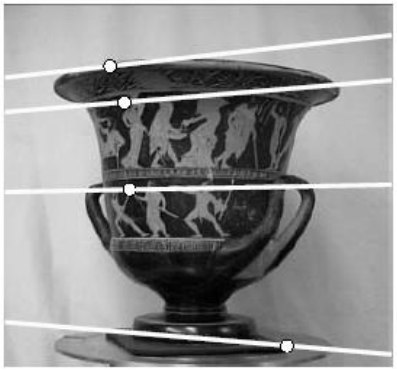
Epipolar Geometry for Converging Cameras

- Still difficult
  - Need to trace different epipolar lines for every point



a

b



c

Image Rectification

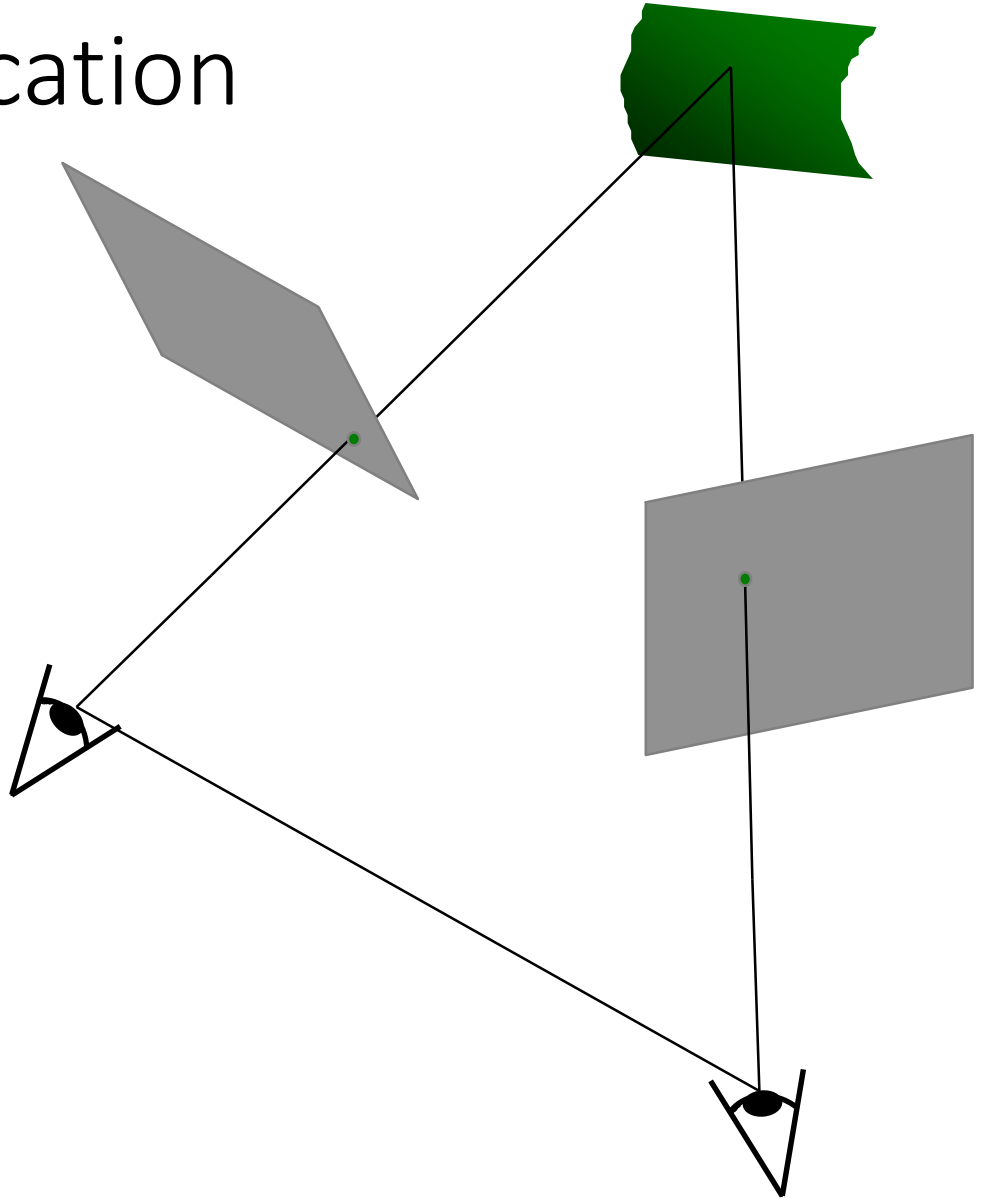


Image Rectification

- Reproject image planes onto a common plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each image

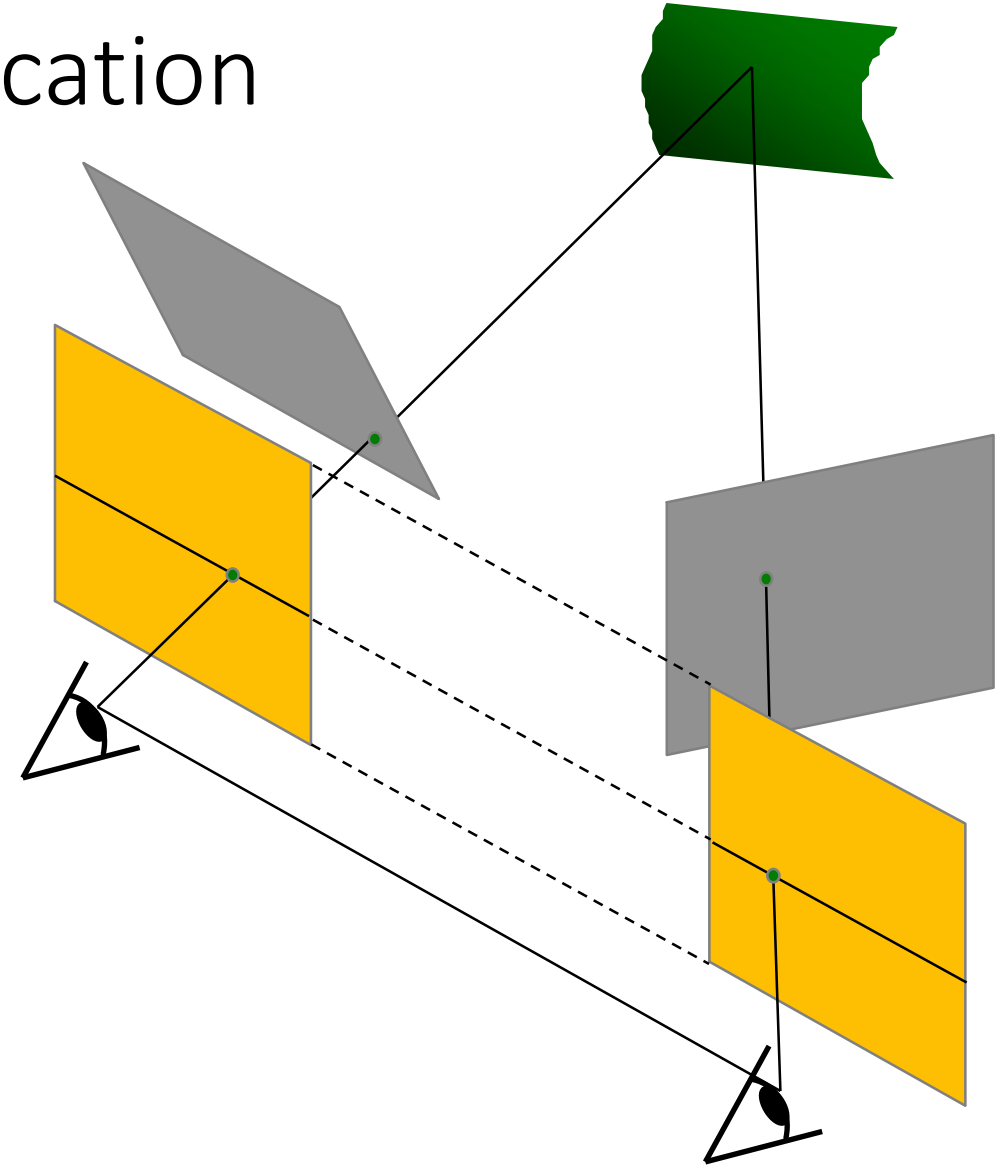


Image Rectification

- [Loop and Zhang 1999]

Original image pair overlaid with several epipolar lines.

Images transformed so that epipolar lines are parallel.

Images rectified so that epipolar lines are horizontal and aligned in vertical.

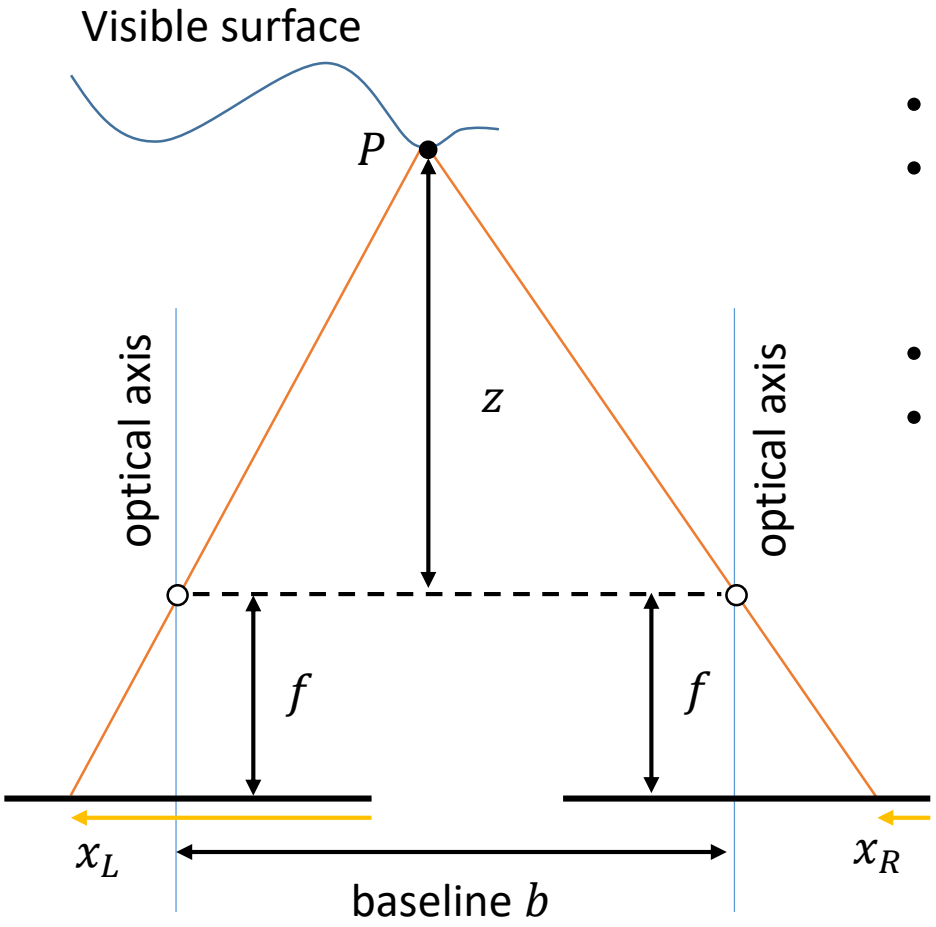
Final rectification that minimizes horizontal distortions. (Shearing)

Loop and Zhang. Computing Rectifying Homographies for Stereo Vision. In CVPR 1999.

Disparity Estimation

- After rectification, stereo matching becomes the disparity estimation problem
- Disparity = horizontal displacement of corresponding points in the two images
  - Disparity of ✕ = $x_L - x_R$

Depth from Disparity

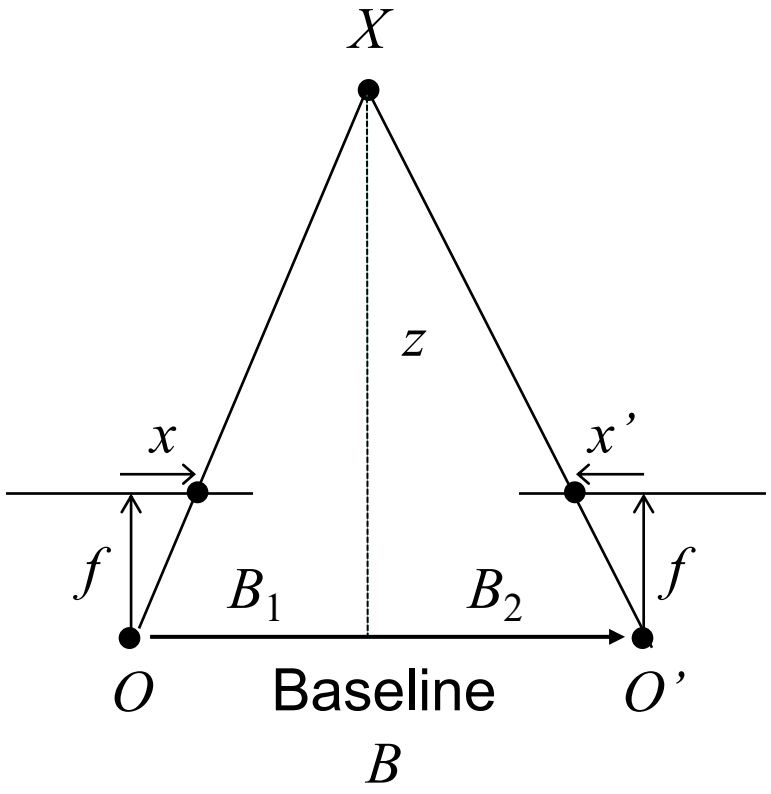


- Disparity $d = x_L - x_R$
- It can be derived that

$$d = \frac{f \cdot b}{z}$$

- Disparity = 0 for distant points
- Larger disparity for closer points

Depth from disparity



$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{-x'}{f} = \frac{B_2}{z}$$

$$\frac{x - x'}{f} = \frac{B_1 + B_2}{z}$$

$$disparity = x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth!

Slide from L. Lazebnik.

Problem formulation

- Given a calibrated binocular stereo pair, fuse it to produce a depth image

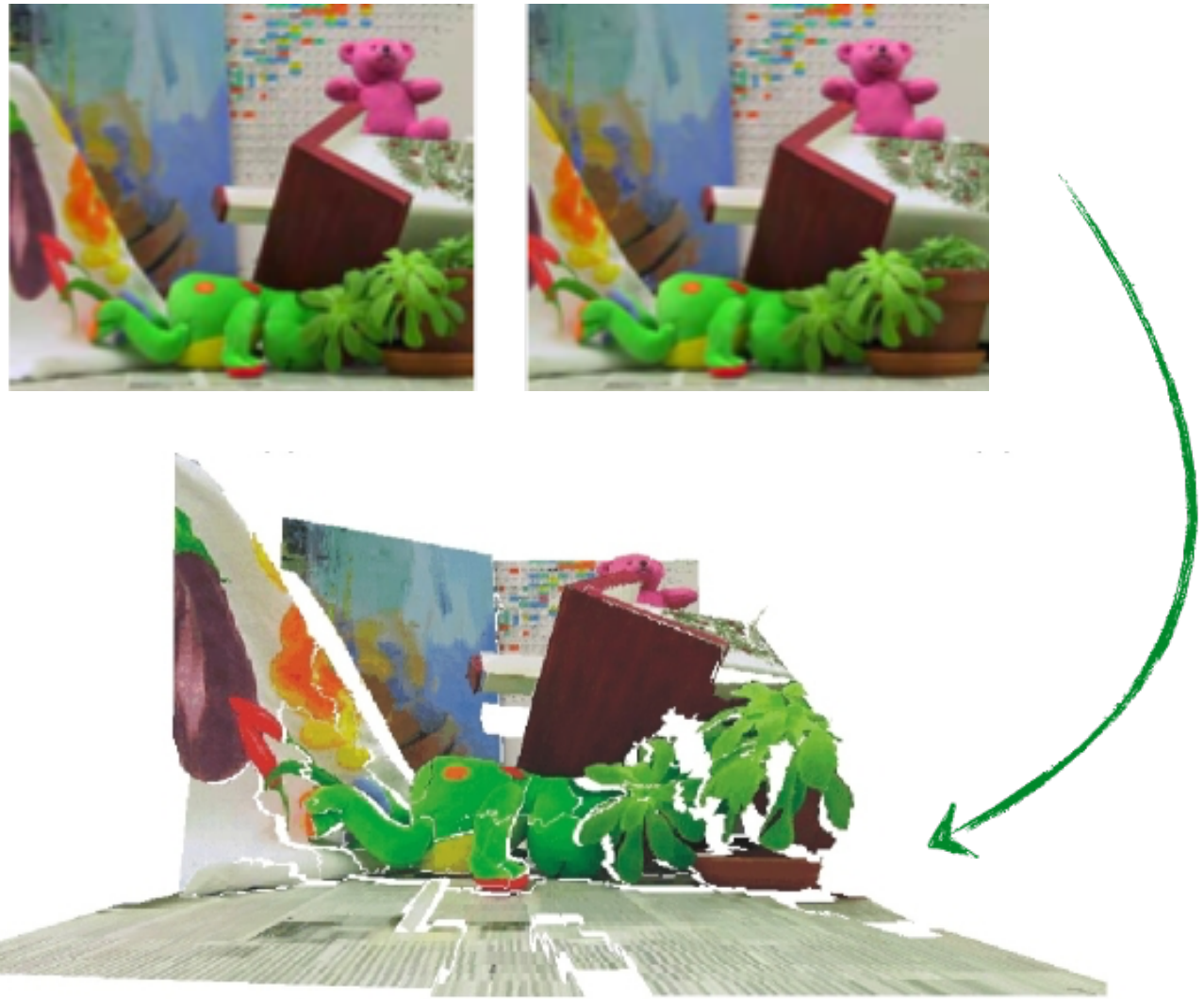
image 1

image 2

Dense depth map

Slide from L. Lazebnik.

Two-View Stereo

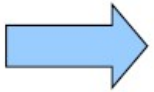


Slides from S. Lazebnik, S. Seitz, Y. Furukawa

Multi-label Problems

Left Camera Image

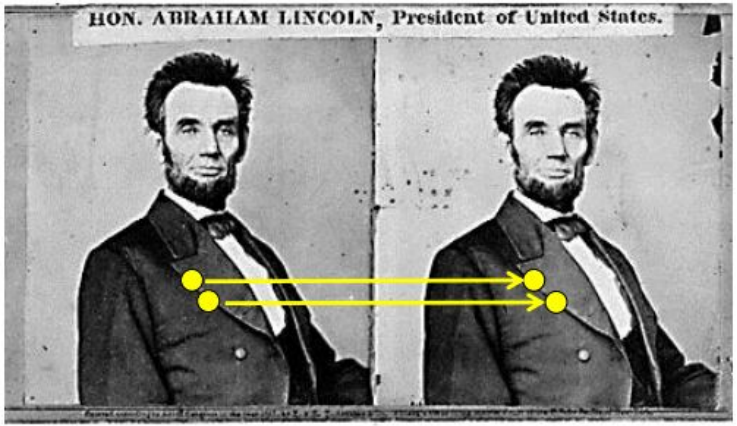
Right Camera Image



Dense Stereo Result

- Choose the disparities from the discrete set: $(1,2,\ldots,L)$

Stereo as Energy Minimization



What defines a good stereo correspondence?

1. Match quality
   - Want each pixel to find a good match in the other image
2. Smoothness
   - If two pixels are adjacent, they should (usually) move about the same amount

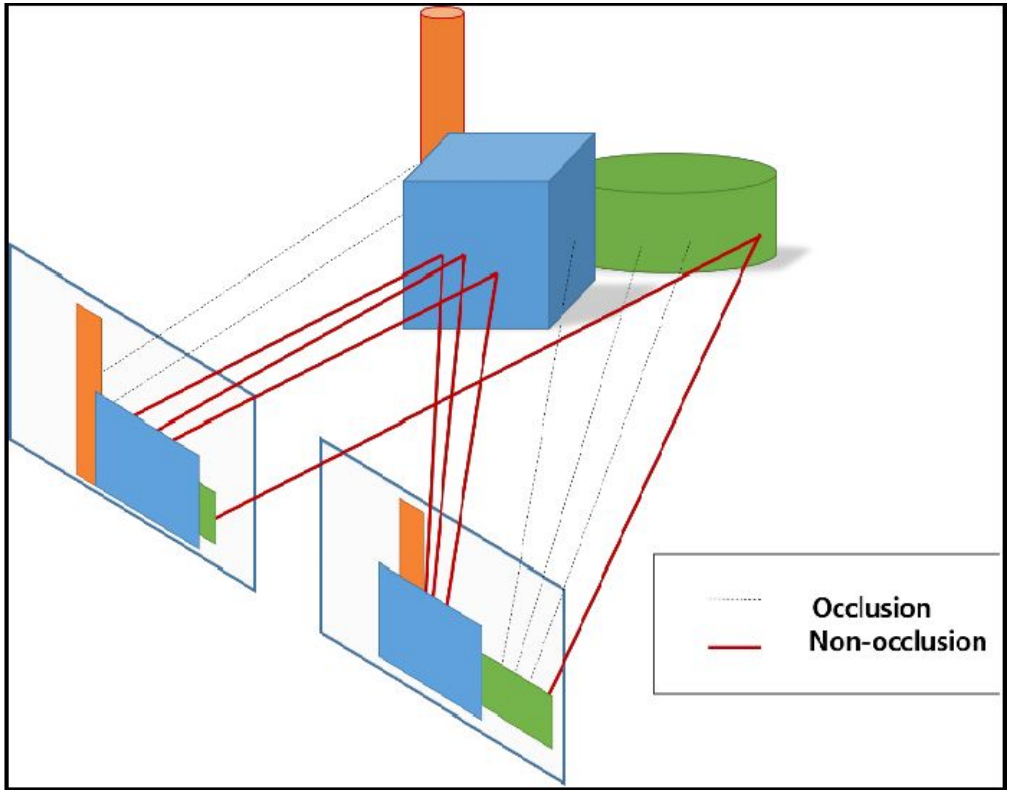
Energy Function

The energy of a configuration f is defined as:

$$\mathrm{E}(f) = \mathrm{E}_{\mathrm{data}}(f) + \mathrm{E}_{\mathrm{occlusion}}(f) + \mathrm{E}_{\mathrm{smoothness}}(f) + \mathrm{E}_{\mathrm{uniqueness}}(f). \quad (1)$$

This energy has four terms. Each term promotes a desired property of the configuration we are looking for. The data term measures how well matched pairs fit, the occlusion term minimizes the number of occluded pixels, the smoothness term penalizes the nonregularity of the configuration, and the last term enforces the uniqueness.

Occlusions



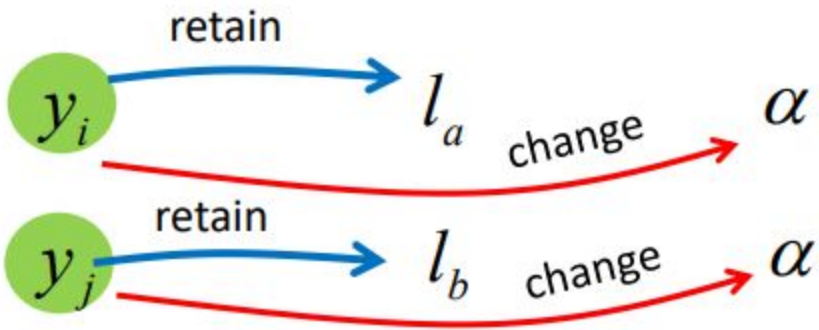
Uniqueness constraint

In an image pair each pixel has at most one corresponding pixel

- In general one corresponding pixel
- In case of occlusion there is none

Expansion Move

$$f(a) = 1 \text{ and } d(a) = \alpha \quad \Rightarrow \quad f'(a) = 1,$$
$$f(a) = 0 \text{ and } d(a) \neq \alpha \quad \Rightarrow \quad f'(a) = 0.$$



- any active assignment with disparity α remains active
- any inactive assignment with disparity different from α remains inactive
- any other assignment can change state (active/inactive).

α - Expansion

Algorithm 1: An iteration of the expansion move algorithm

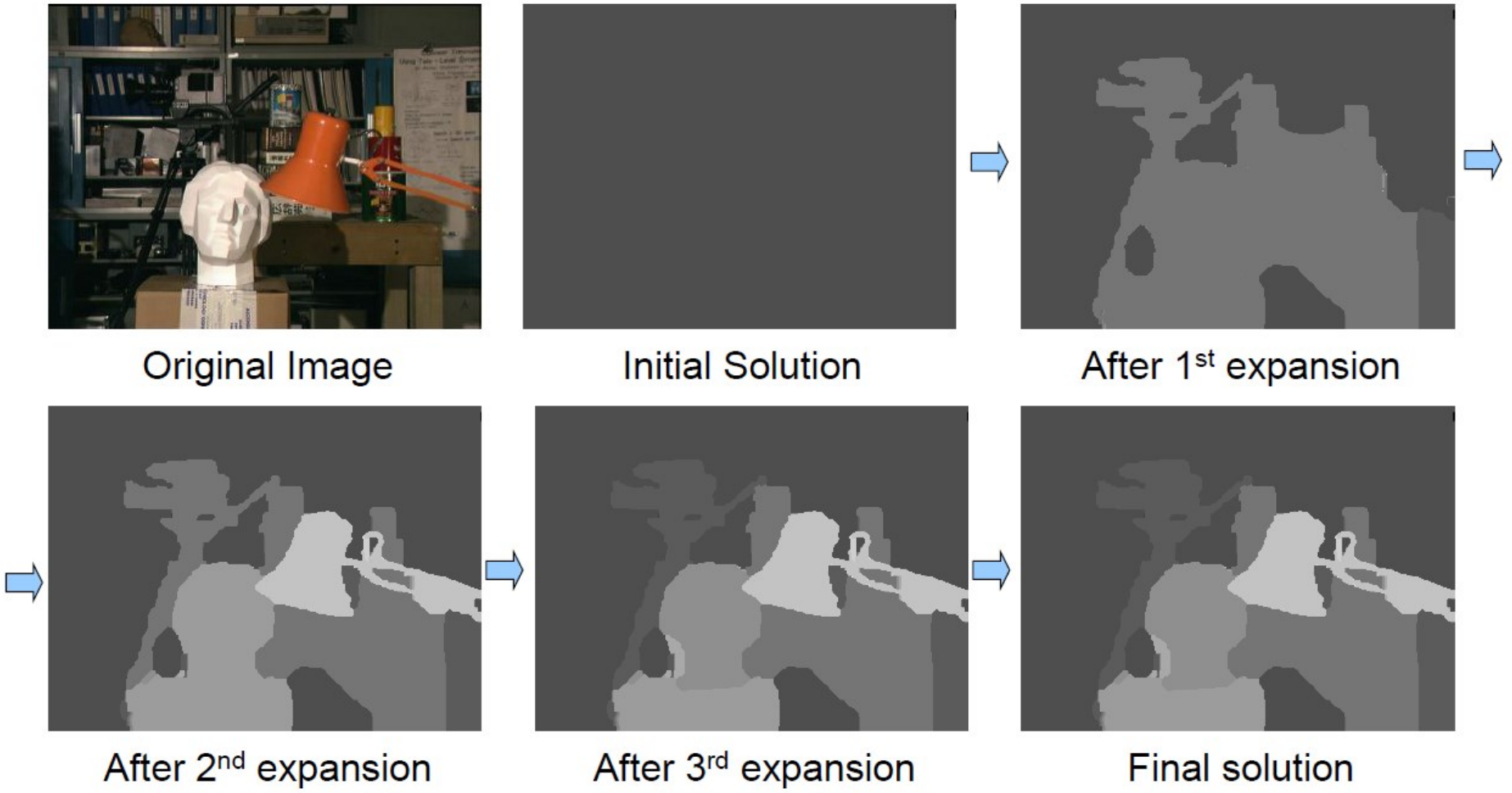
Input: a unique configuration f, interval of disparities $\mathrm{I}_{\mathrm{disp}}$, achieved α-expansions array `done`
Output: updated unique configuration f with smaller or equal energy

foreach α *(in randomly ordered* I_{disp}*)* **do**
  if *not* `done`$[\alpha]$ **then**
    Find the α-expansion move f^* of f that decreases the most the energy:

$$f^* \leftarrow \arg\min_{f' \, \alpha-\text{expansion move of } f} \mathrm{E}(f')$$

    if $\mathrm{E}(f^*) < \mathrm{E}(f)$ **then**
      $f \leftarrow f^*$
      `done`$[:] \leftarrow$ false
    `done`$[\alpha] \leftarrow$ true
    if *done[:]=true* **then**
      return f

$\alpha-$ Expansion

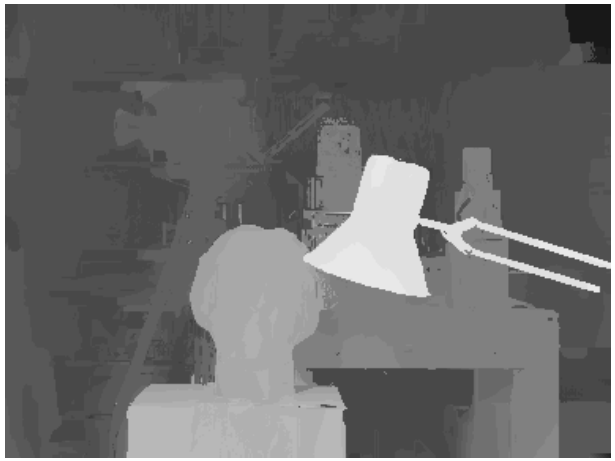
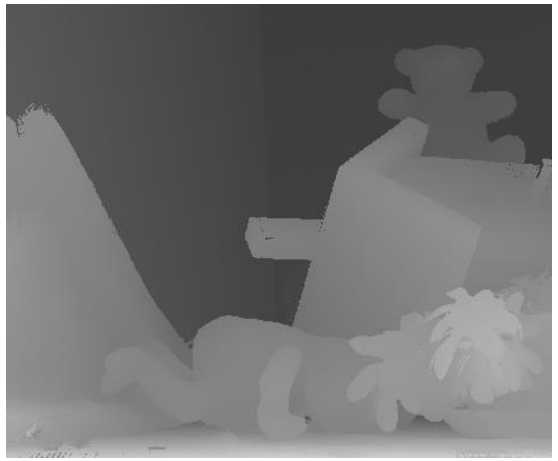
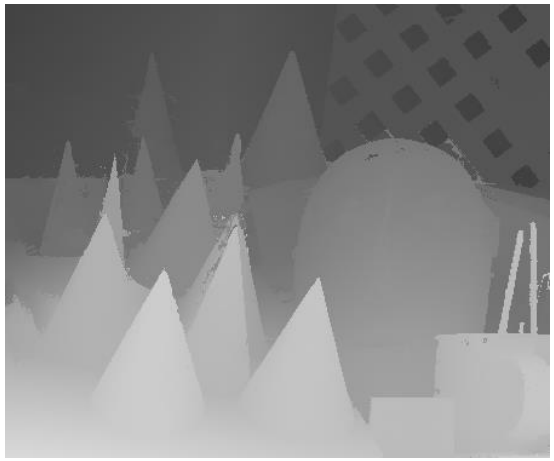


[Image courtesy: Lubor Ladicky]

[Boykov et al. 2001]

Graph Cut

- GC can also be used to minimize $E(d) = \sum_{p} D(d_p) + \lambda \sum_{p,q} V(d_p, d_q)$

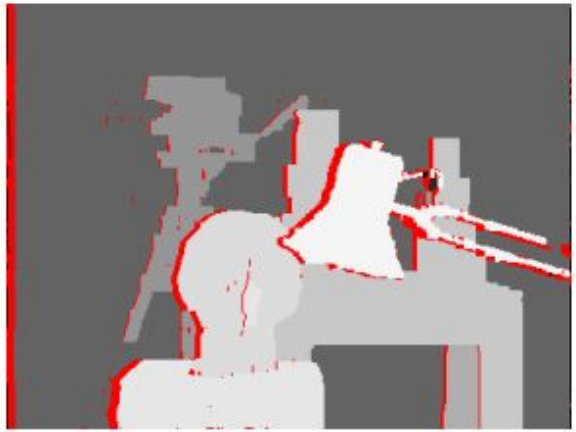




Results from Taniai et al.

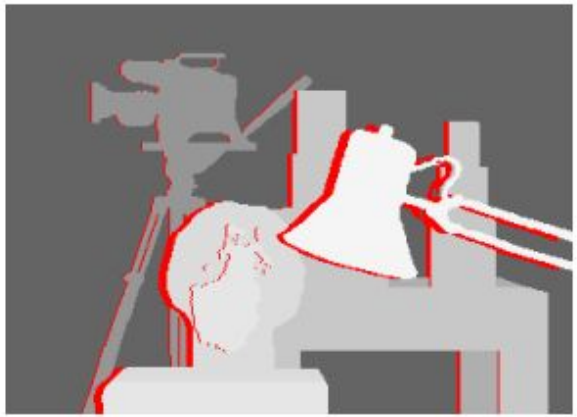
Taniai et al. Graph cut based continuous stereo matching using locally shared labels. In CVPR 2014.
Kolmogorov et al. What energy functions can be minimized via graph cuts? PAMI 2004
Boykov et al. Fast approximate energy minimization via graph cuts. In ICCV 1999.

Results in the paper

Left image

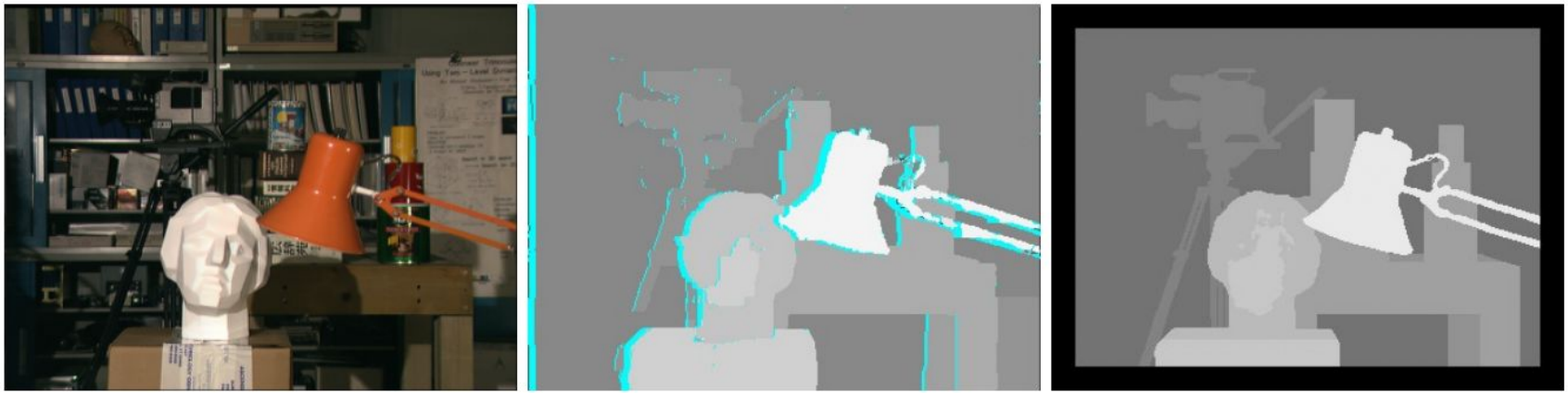


Our results



Ground truth

What you need to do?



(a) Tsukuba (left image)

(b) Result of algorithm run without cutting the image into several strips

(c) Groundtruth (black pixels mean "no data")

Figure 7: Results of the algorithm on the *Tsukuba* pair.