UNIVERSITAT DE BARCELONA

>>> IMAGE PROCESSING AND COMPUTATIONAL PHOTOGRAPHY

SESSION 6: CUT

Oriol Pujol

TODAY'S LECTURE

- Analyzing images as graphs
- Application: Retargeting (Seam carving)
- Semi-automatic segmentation: Grab-cuts

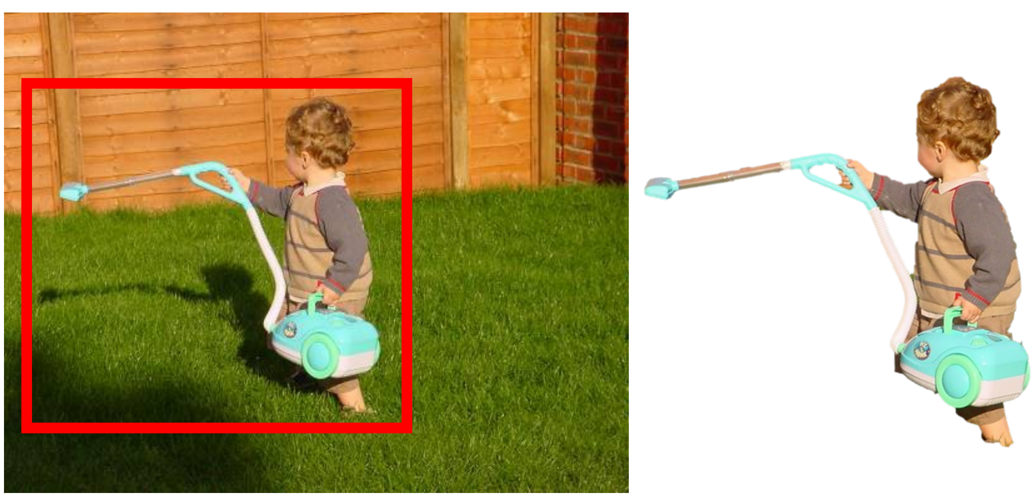
RETARGETING

Context-aware image resizing. Remove seams (horizontal or vertical) with the least information.

SEMI-AUTOMATIC SEGMENTATION

Given a region of interest or line segments in the desired regions provided by the user
Obtain a complete context-aware segmentation/partition into the desired object and background.

Techniques:
- Intelligent scissors
- Grab-cuts



SUMMARY OF BIG IDEAS

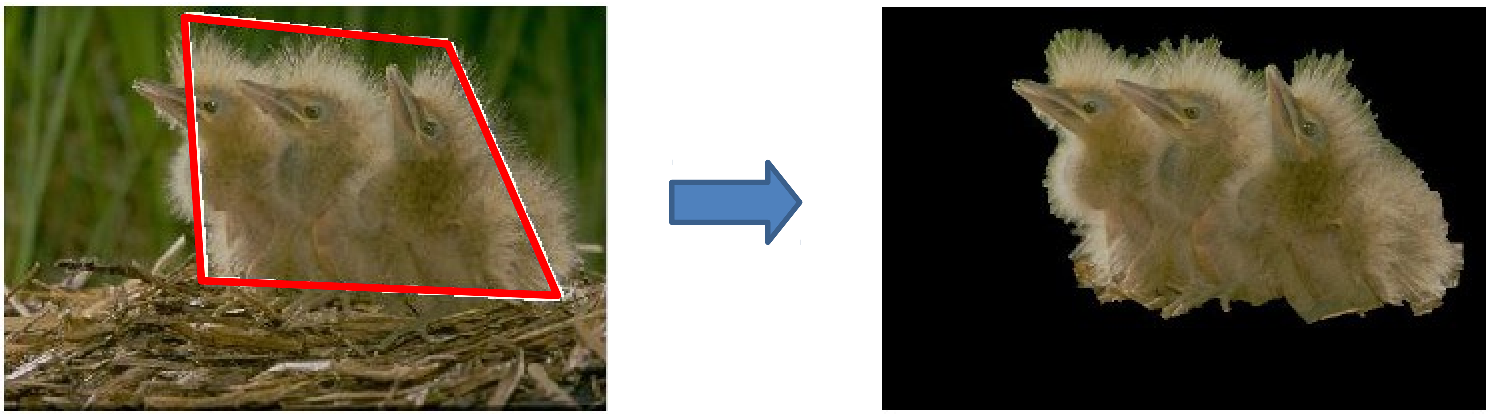
- Treat image as a graph
  - Pixels are nodes
  - Between-pixel edge weights based on gradients
  - Sometimes per-pixel weights for affinity to foreground/background

- Good boundaries are a short path through the graph (Intelligent Scissors, Seam Carving)

- Good regions are produced by a low-cost cut (GrabCuts, Graph Cut Stitching)

SEMI-AUTOMATED SEGMENTATION

User provides imprecise and incomplete specification of region – your algorithm has to read his/her mind.



Key problems

1. What groups of pixels form cohesive regions?
2. What pixels are likely to be on the boundary of regions?
3. Which region is the user trying to select?

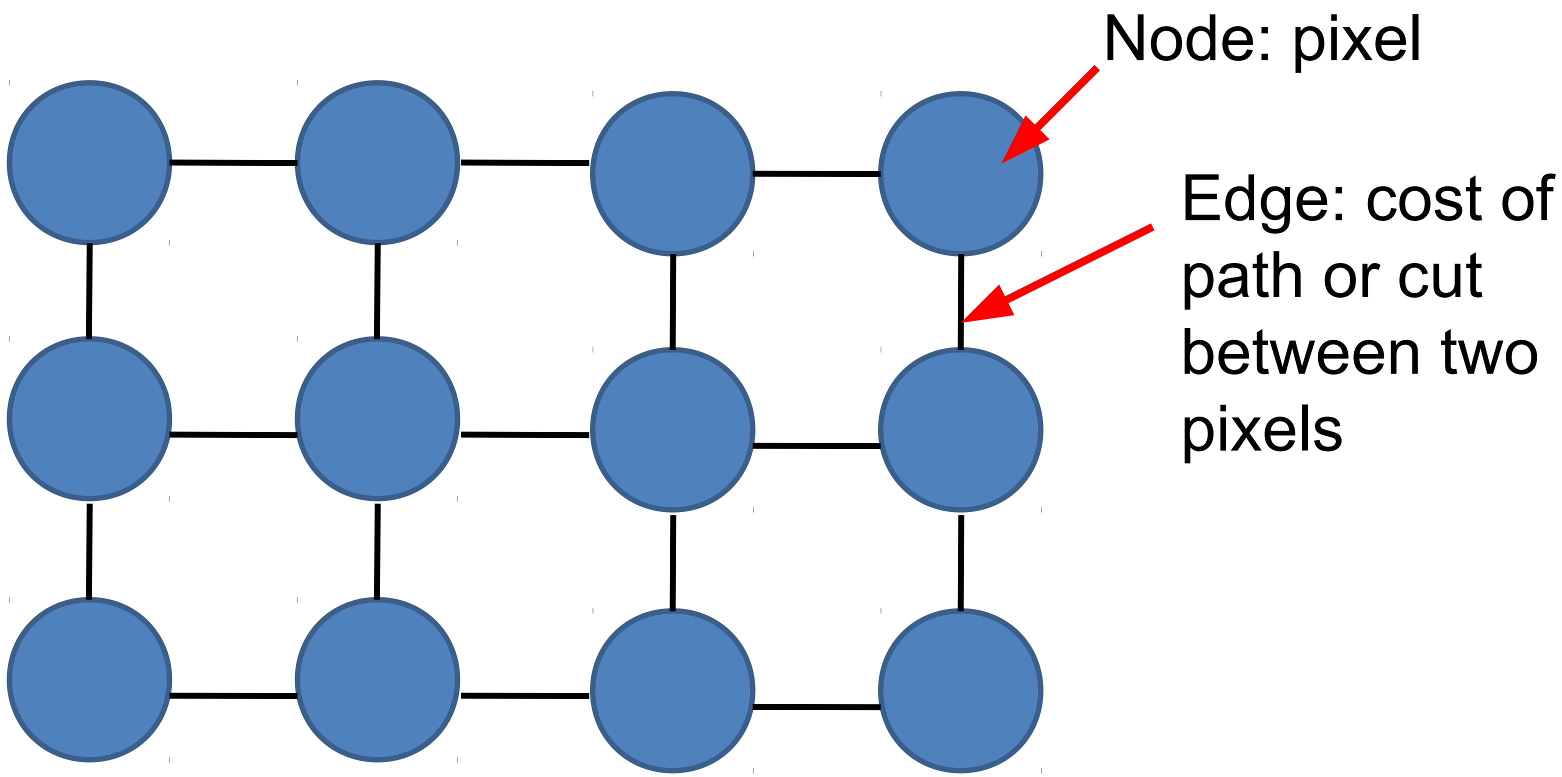
WHAT MAKES A GOOD REGION?

- Contains small range of color/texture
- Looks different than background
- Compact

WHAT MAKES A GOOD REGION?

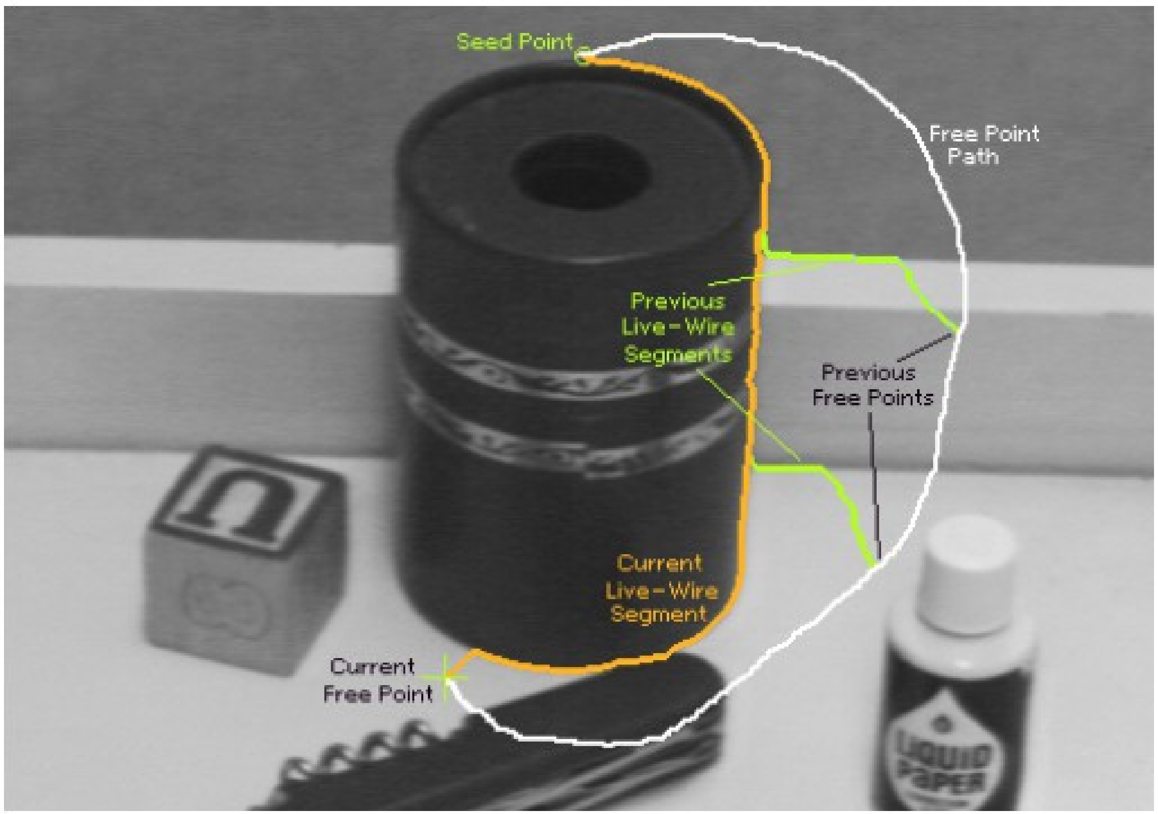
- High gradient along boundary
- Gradient in right direction
- Smooth

IMAGE AS A GRAPH



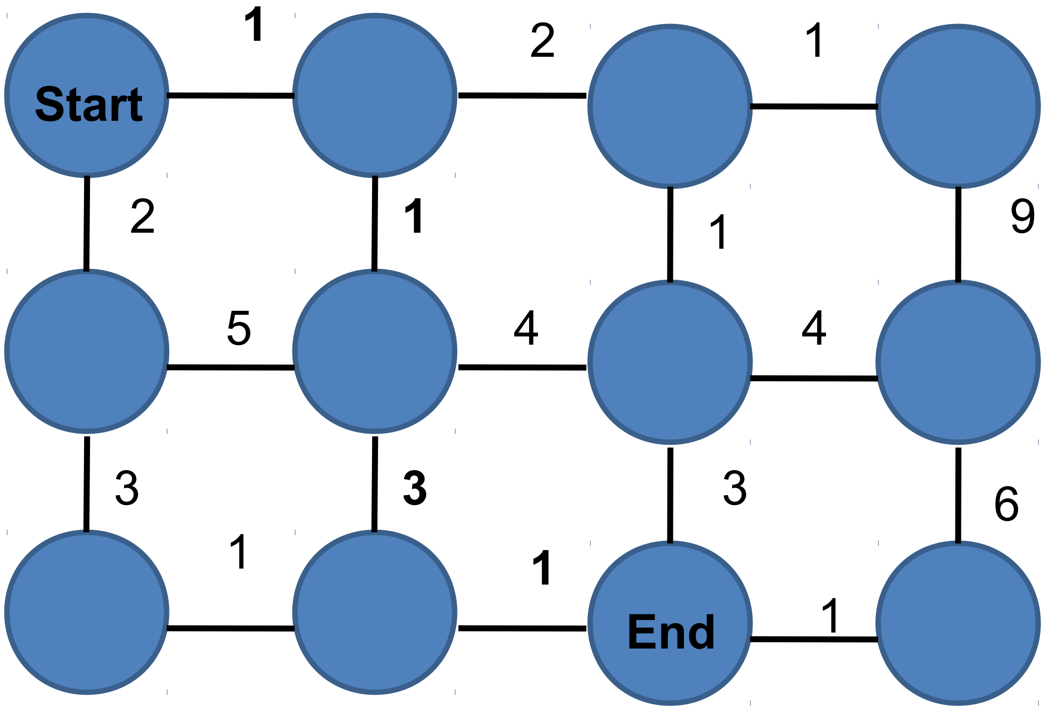
INTELLIGENT SCISSORS

Mortenson and Barrett “Intelligent scissors for Image Composition” (SIGGRAPH 1995)



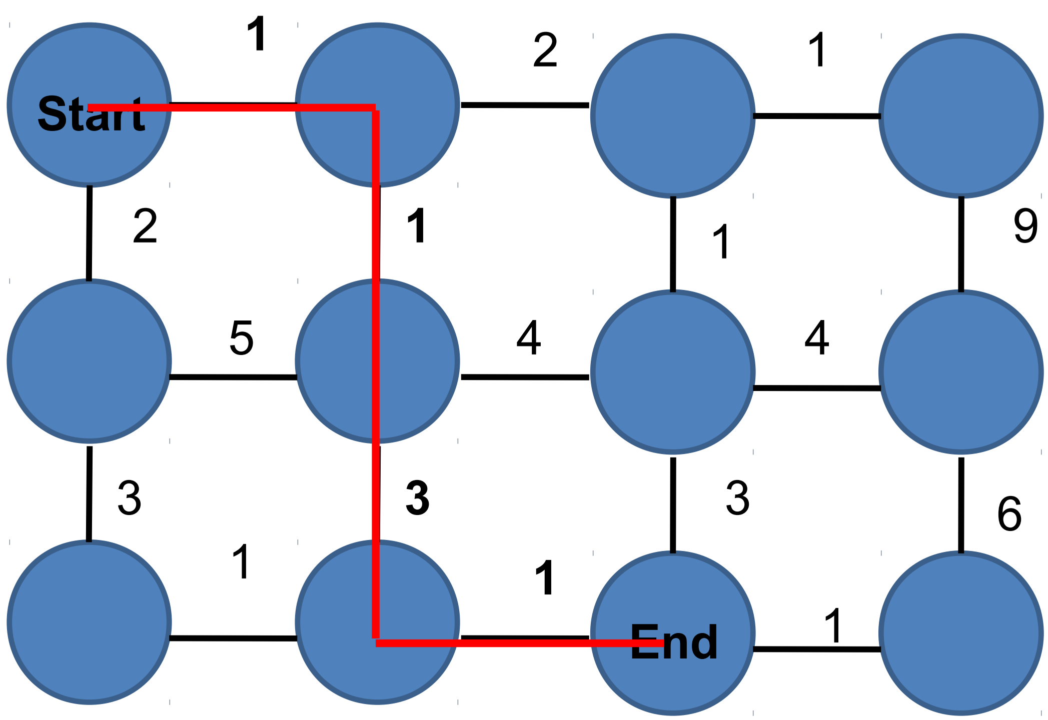
INTELLIGENT SCISSORS

Mortenson and Barrett "Intelligent scissors for Image Composition" (SIGGRAPH 1995)



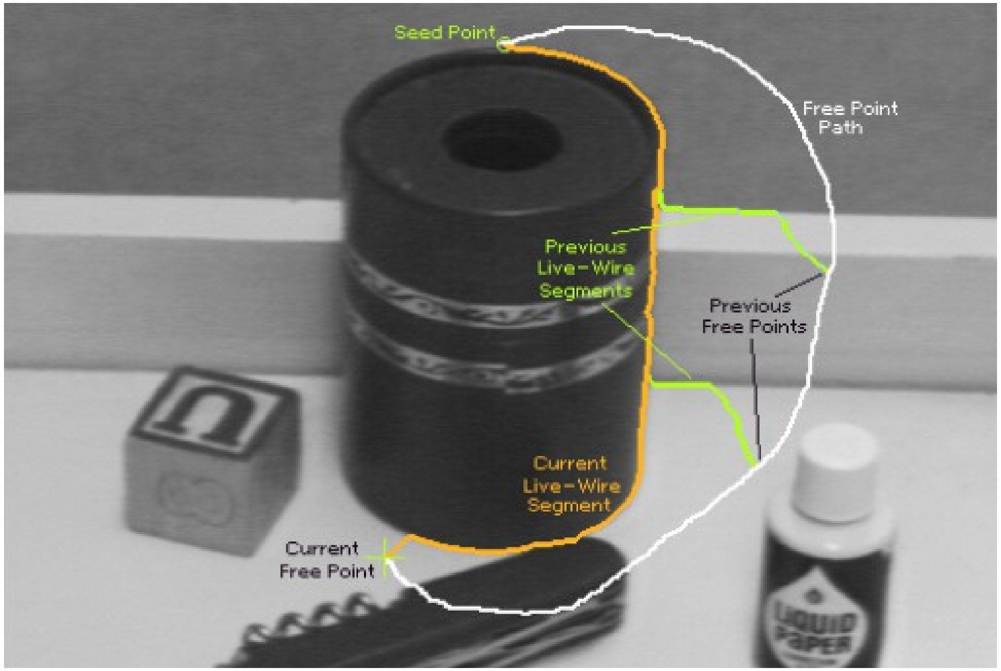
INTELLIGENT SCISSORS

Mortenson and Barrett “Intelligent scissors for Image Composition” (SIGGRAPH 1995)



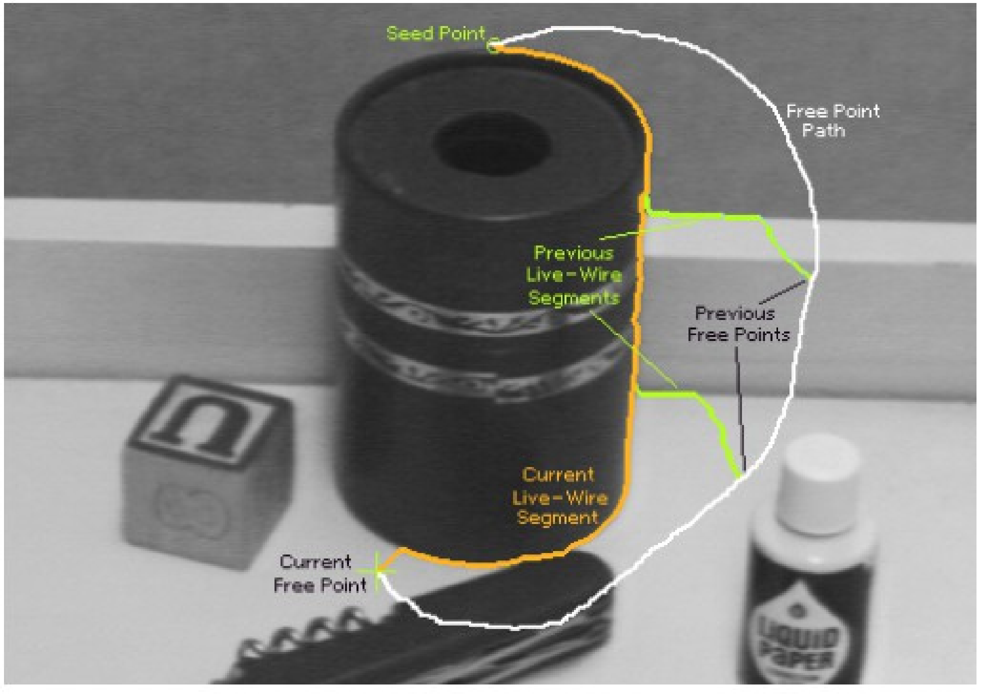
INTELLIGENT SCISSORS

- Formulation: find good boundary between a free hand point and a seed point.
- Challenges
  – Minimize interaction time
  – Define what makes a good boundary
  – Efficiently find it



INTELLIGENT SCISSORS: METHOD

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get path from seed to cursor, (we may choose a new seed), repeat.



INTELLIGENT SCISSORS: METHOD

1. Define boundary cost between neighboring pixels (GOAL: Follow edges)
   a) Lower if edge is present (e.g., with edge(im, ‘canny’))
   b) Lower if gradient is strong
   c) Lower if gradient is in direction of boundary

GRADIENT, EDGES, AND PATH COST

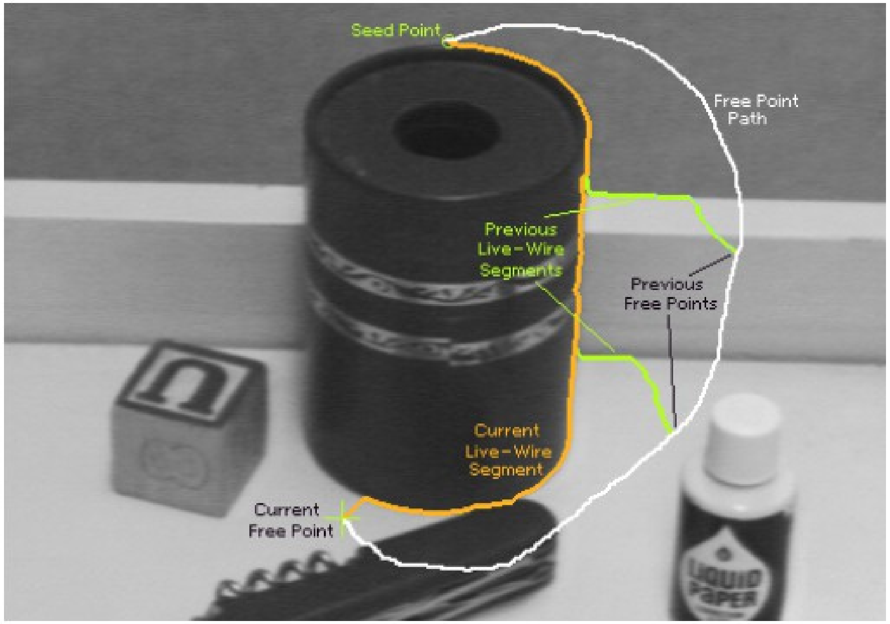
Gradient Magnitude

Path Cost

Edge Image

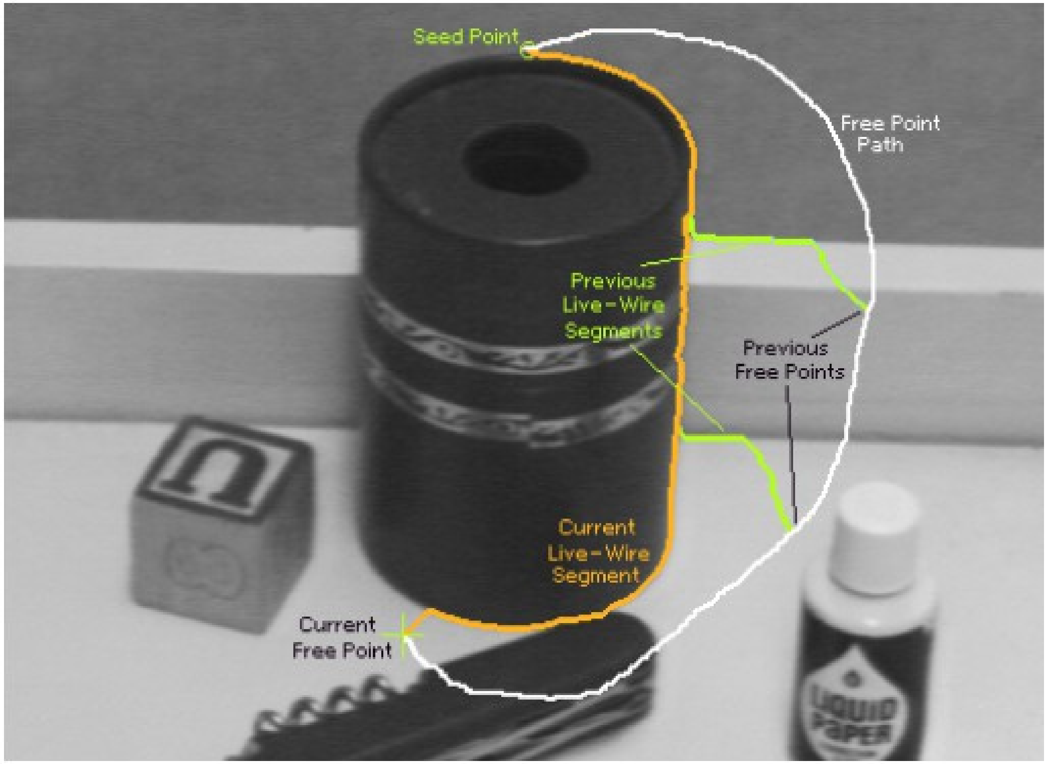
INTELLIGENT SCISSORS: METHOD

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
   - Snapping



INTELLIGENT SCISSORS: METHOD

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
   – Djikstra’s shortest path algorithm



DJIKSTRA'S SHORTEST PATH ALGORITHM

Initialize, given seed *s*:

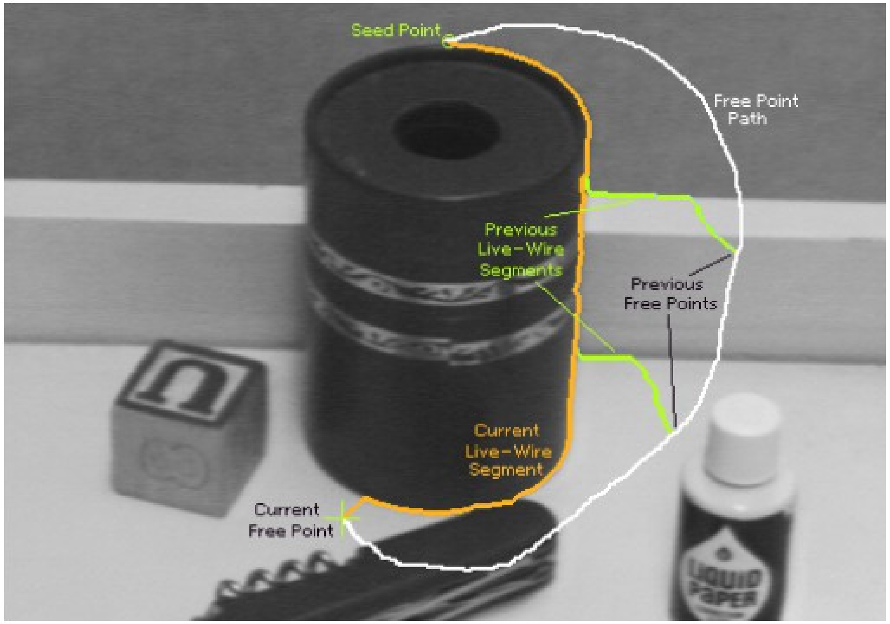
- Compute cost2(q, r) % cost for boundary from pixel *q* to neighboring pixel *r*
- cost(*s*) = 0 % total cost from seed to this point
- **A** = {*s*} % set to be expanded
- **E** = { } % set of expanded pixels
- **P**(q) % pointer to pixel that leads to *q*

Loop while **A** is not empty

1. *q* = pixel in **A** with lowest cost
2. Add q to **E**
3. for each pixel *r* in neighborhood of *q* that is not in **E**
   a) cost_tmp = cost(*q*) + cost2(*q*,*r*)
   b) if (*r* is not in **A**) OR (cost_tmp < cost(*r*))
      i. cost(*r*) = cost_tmp
      ii. **P**(*r*) = *q*
      iii Add *r* to **A**

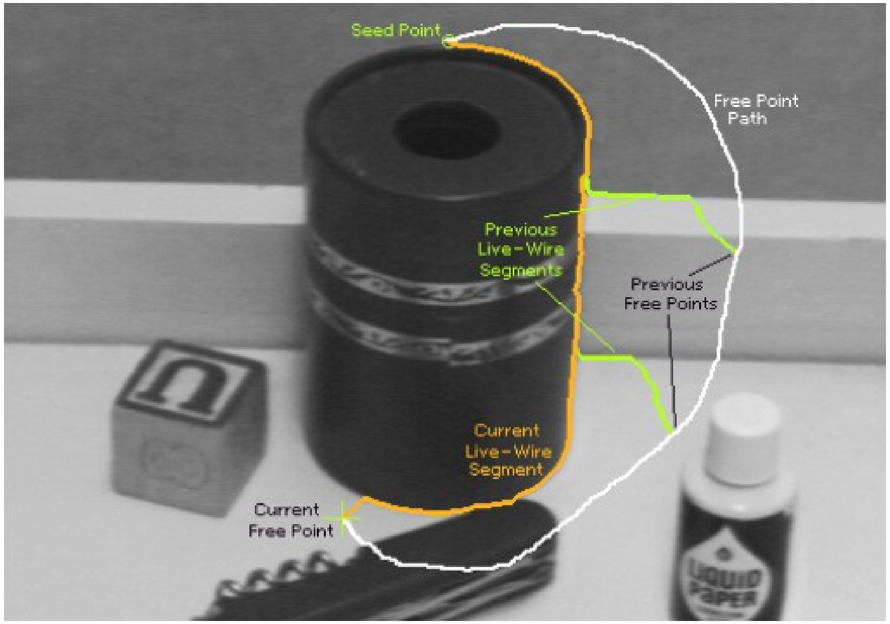
INTELLIGENT SCISSORS: METHOD

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get new seed, get path between seeds, repeat

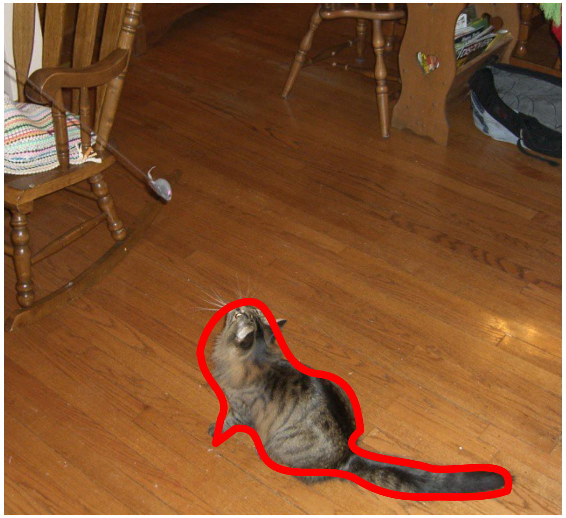


INTELLIGENT SCISSORS: IMPROVING INTERACTION

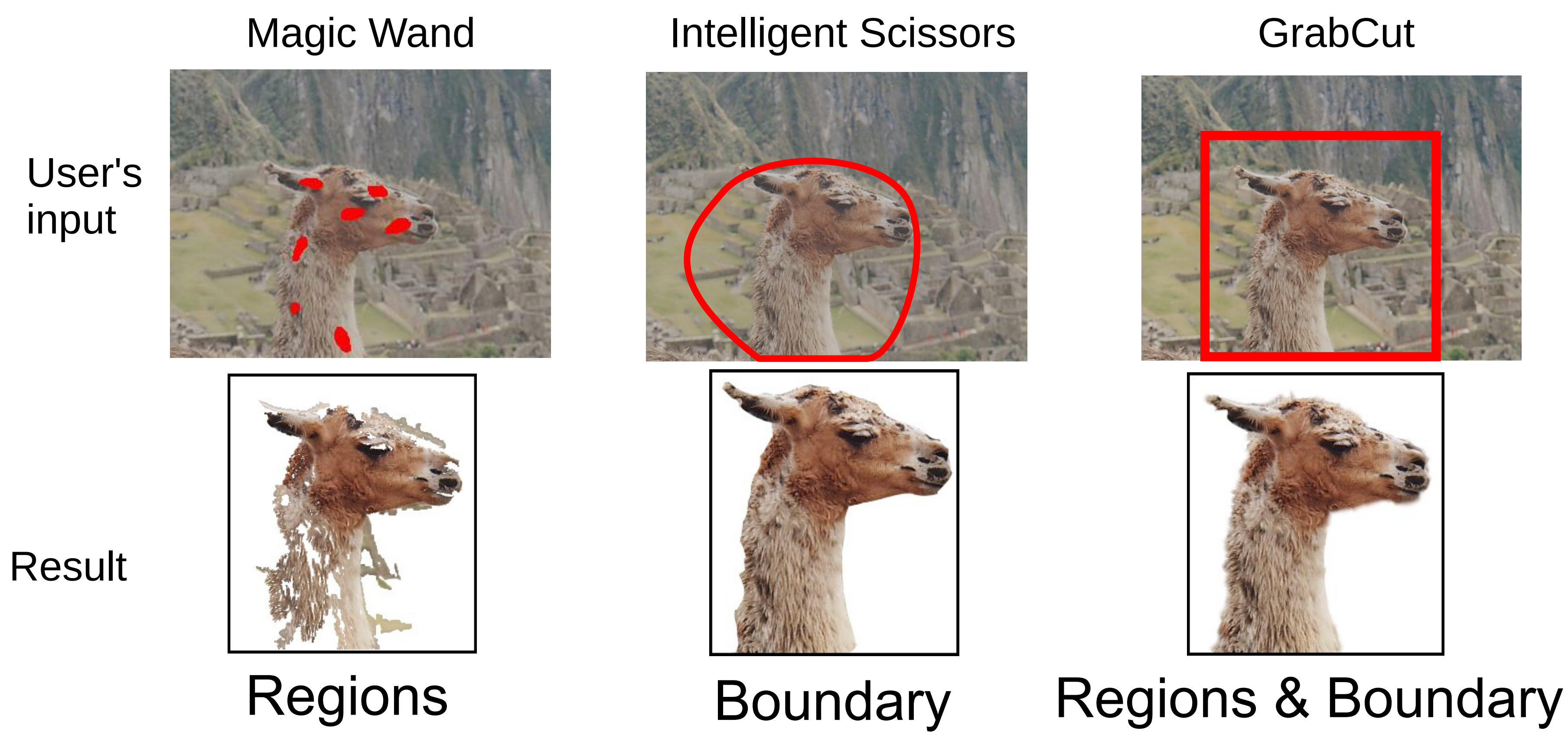
1. Snap when placing first seed
2. Automatically adjust to boundary as user drags
3. Freeze stable boundary points to make new seeds



Where will intelligent scissors work well, or have problems?



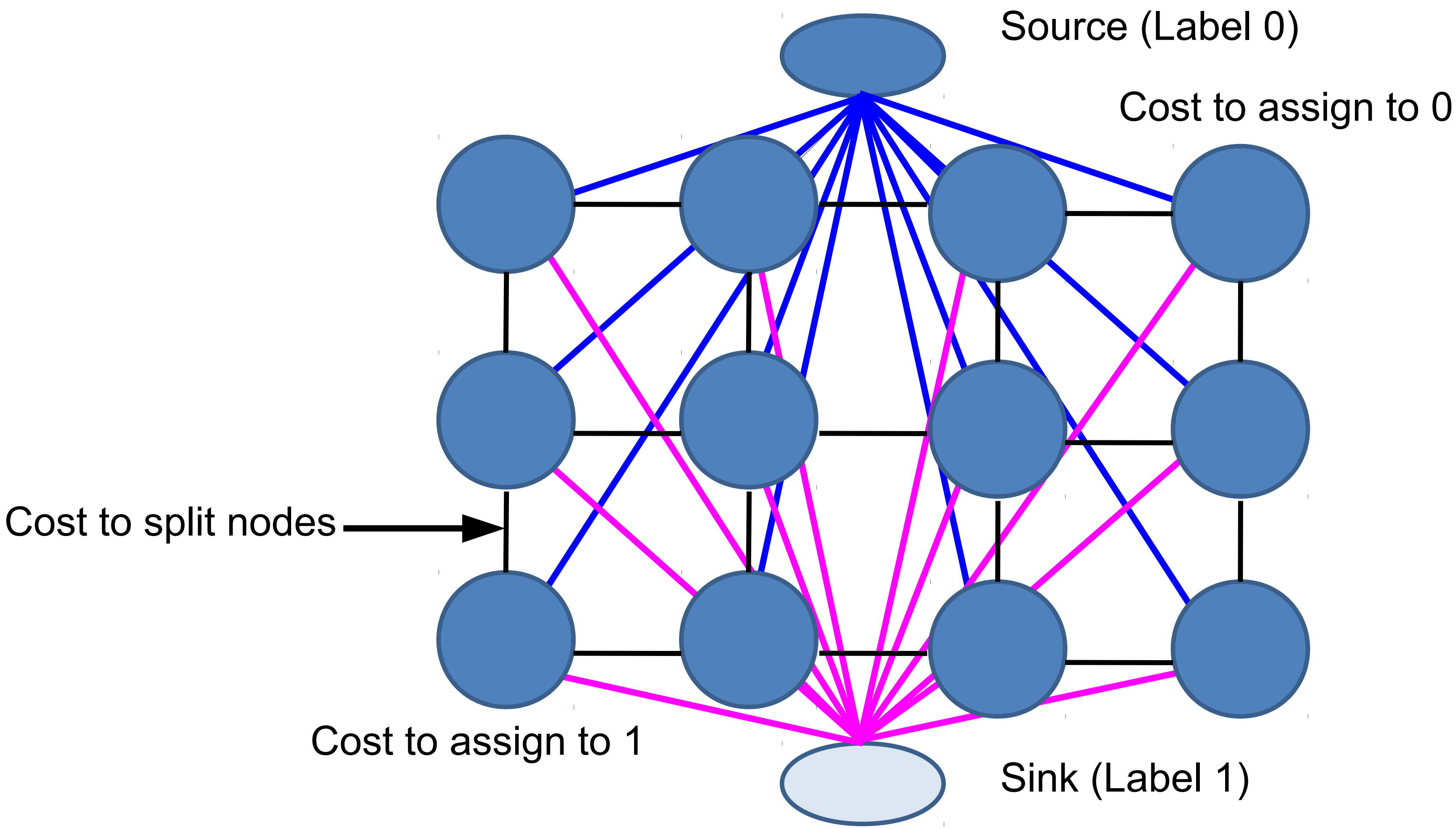
GRAB CUTS and GRAPH CUTS



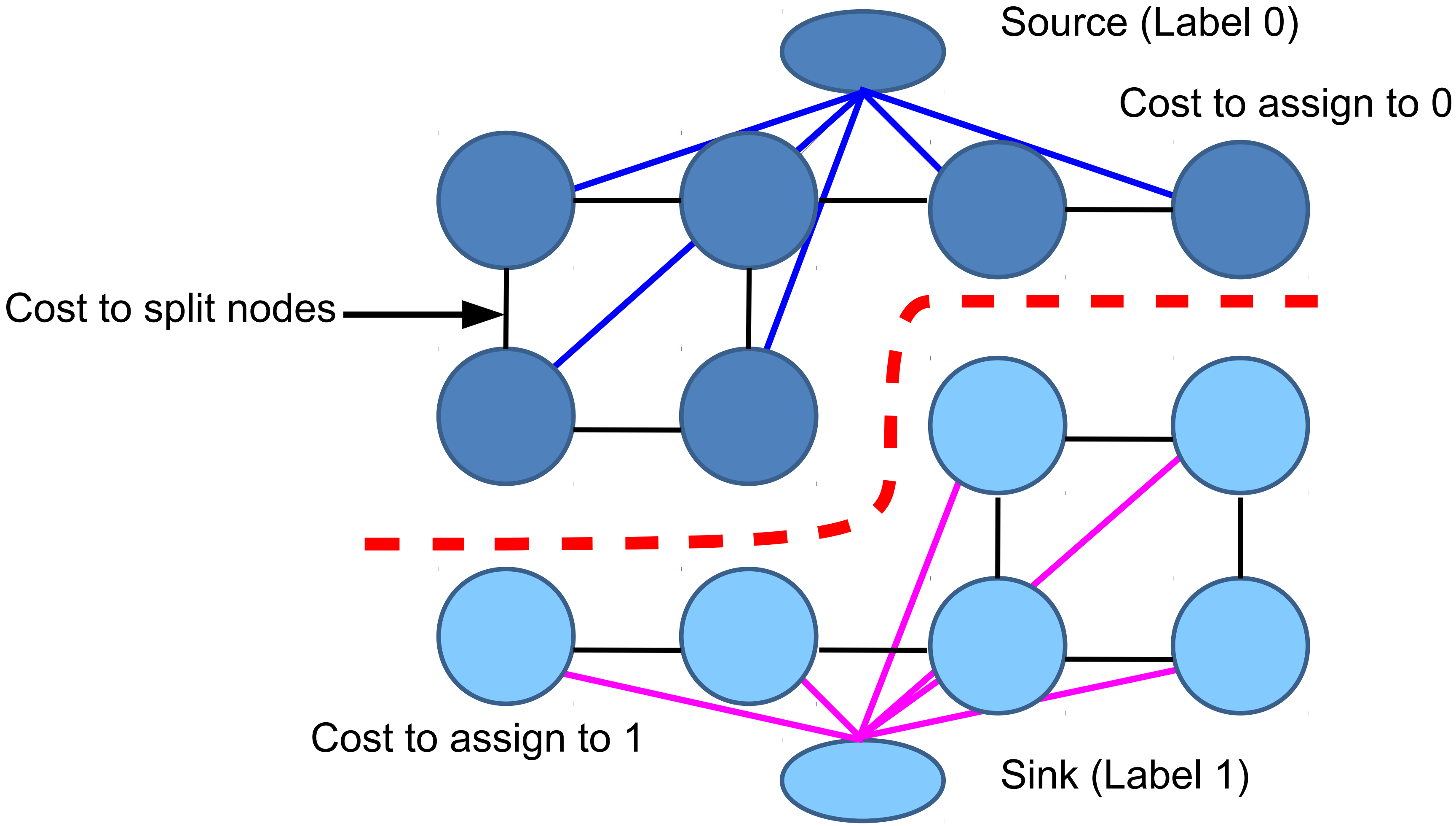
SEGMENTATION WITH GRAPH CUTS

Do you remember Ford-Fulkerson MIN-CUT/MAX-FLOW algorithm?
It is time to apply it.

SEGMENTATION WITH GRAPH CUTS



SEGMENTATION WITH GRAPH CUTS



SEGMENTATION WITH GRAPH CUTS

Nowadays vision of graph based segmentation relies on the minimization of a potential or energy function.

Formally, we are looking for the labels assignation that minimizes the potential function

$$l : \mathbf{R}^{M\times N} \rightarrow \{+1, -1\}^{M\times N}$$

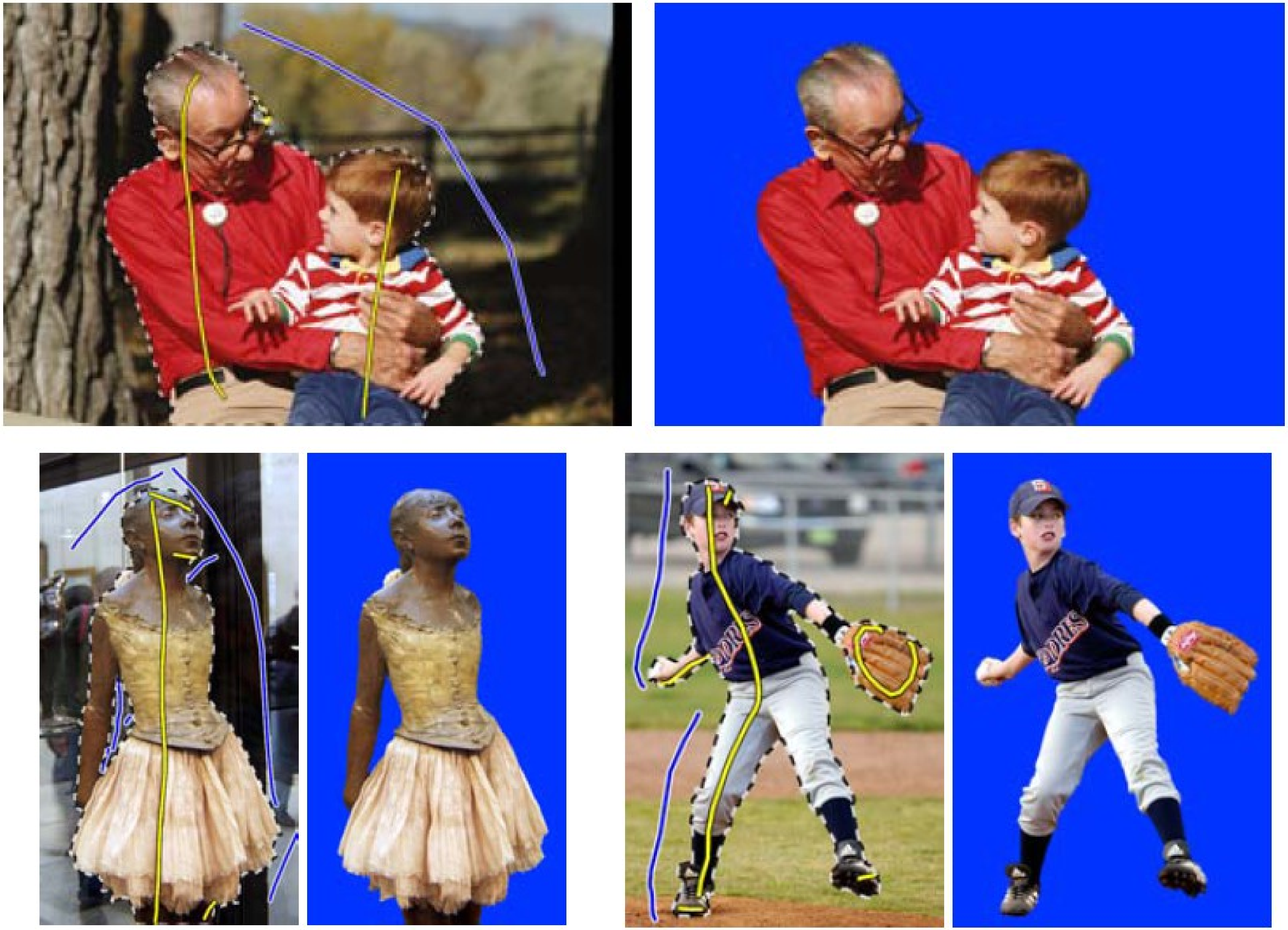
$$L = \arg\min_{l} \sum_{p} \Big(\psi(l(p), p) + \sum_{q\in\mathcal{N}(p)} \eta(p, q, l(p), l(q))\Big)$$

It is usually composed of two terms:

Unary potential: Defines the cost/probability/likelihood/confidence of a pixel belonging to one of the class labels.

Edge potential: Transition cost. Defines the cost of two adjacent pixels for changing the class label or staying with the same label. It encodes a smoothness constraint.

SEGMENTATION WITH GRAPH CUTS



UNARY POTENTIAL

Given a certain user input for the target region (foreground) and the rest (background).
GOAL: Model both regions on a feature space.

HOW?

Feature space: Color space (e.g. sRGB, CIELab, …)
Simple model: Gaussian model

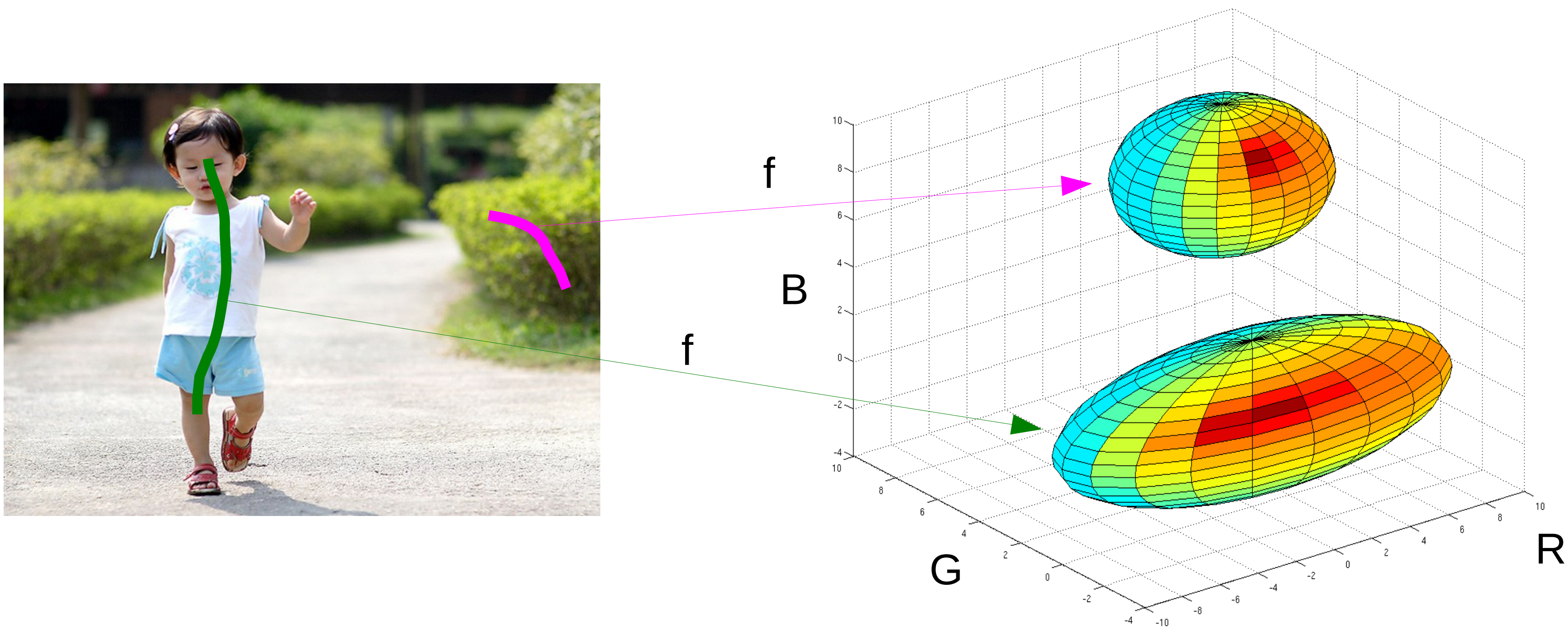
$$\phi(x, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-1/2(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

Unknowns: We have to estimate the mean and covariance matrix.

Simplifying the covariance matrix:

$$\Sigma = \sigma^2 I$$

UNARY POTENTIAL: COLOR MODEL



What are your opinions with respect to this modeling?

UNARY POTENTIAL

Given a certain user input for the target region (foreground) and the rest (background).
GOAL: Model both regions on a feature space.

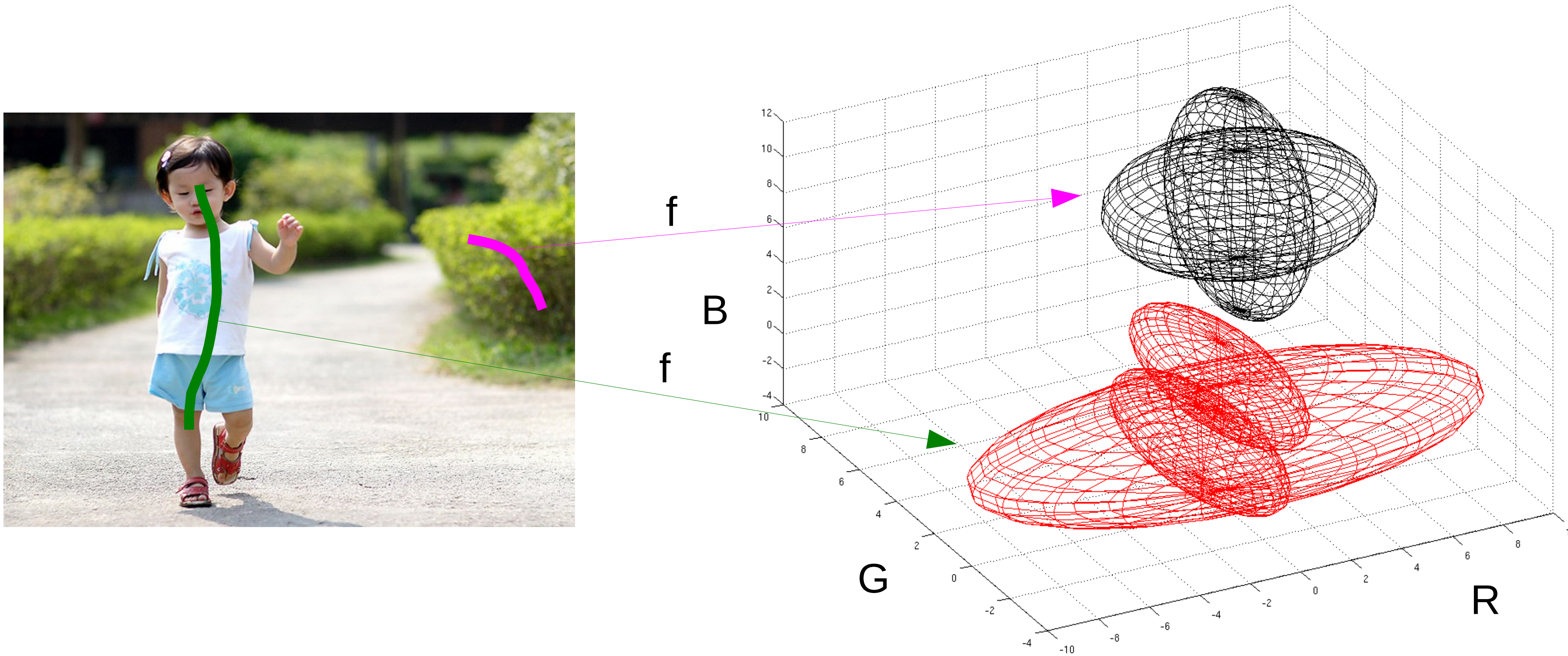
HOW?

Feature space: Color space (e.g. sRGB, CIELab, …)
A better model: Gaussian mixture model

$$\mathcal{M}(x) = \sum_i \alpha_i \phi(x, \mu_i, \Sigma_i)$$

Unknowns: We have to jointly estimate the mean and covariance matrix for each of the composing Gaussian functions and the mixing weighing vector. One can address this problem using **Expectation Maximization (EM) algorithm.**

UNARY POTENTIAL: COLOR MODEL



What are your opinions with respect to this modeling?

EDGE POTENTIAL: SMOOTHNESS

No user input needed
GOAL: Model a smoothness constraint.

HOW?

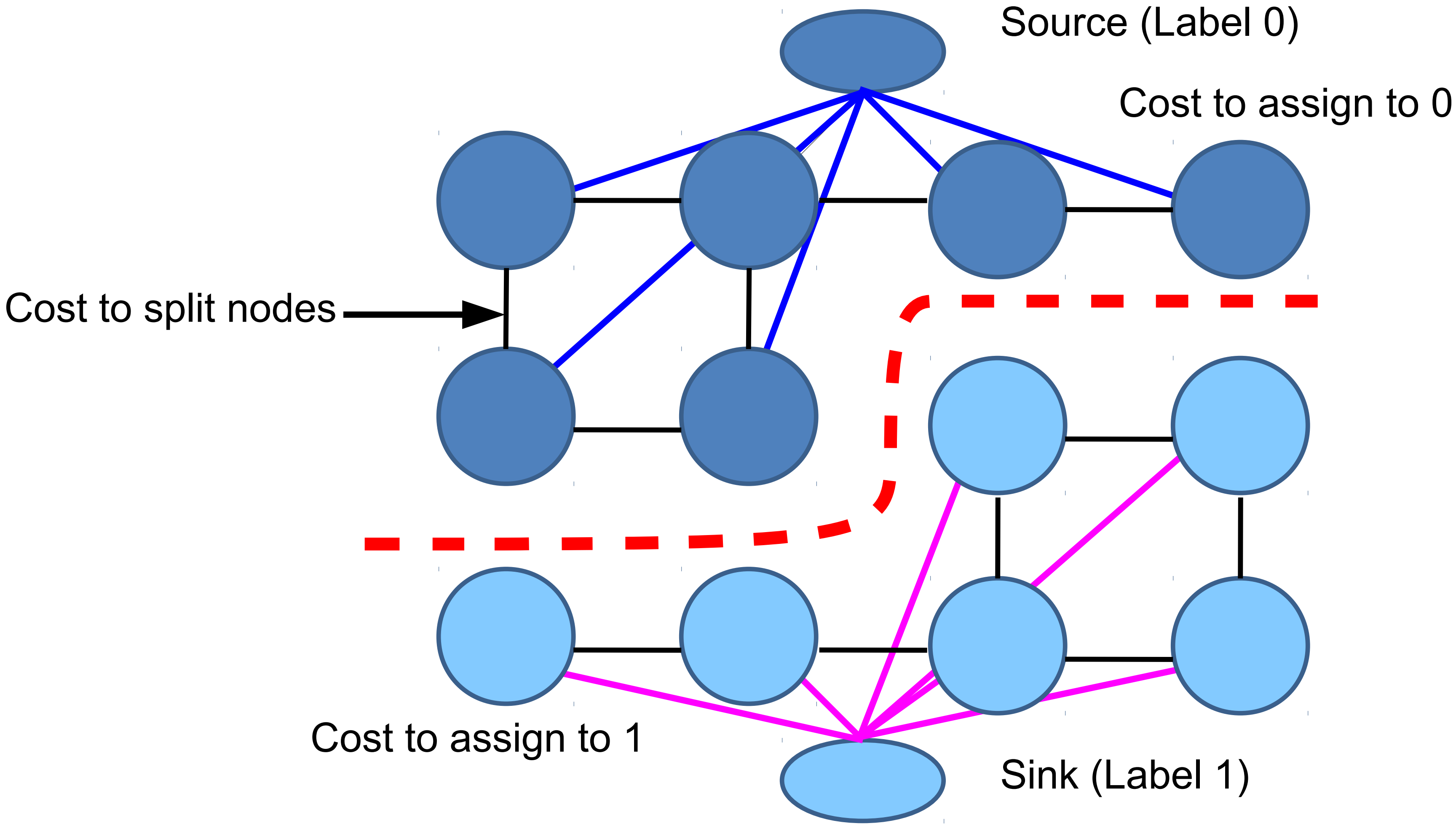
Feature space: Color space (e.g. sRGB, CIELab, …)
Model: Enforce smoothness. Penalize differences of features on contiguous pixels.

$$\eta(p, q, l(p), l(q)) = e^{\omega \| f(p) - f(c) \|^2}$$

GRAPH CUTS SEGMENTATION ALGORITHM

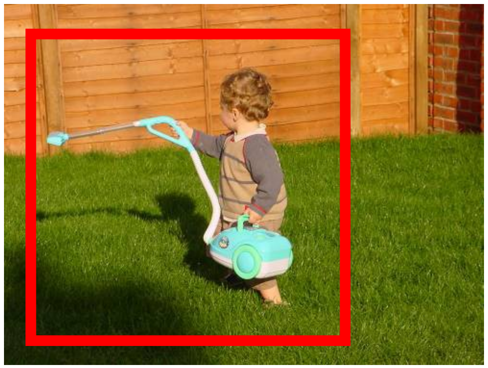
1. Define graph
   – usually 4-connected or 8-connected
2. Set weights to foreground/background
   – Color histogram or mixture of Gaussians for background and foreground
3. Set weights for edges between pixels
4. Apply min-cut/max-flow algorithm
5. Return to 2, using current labels to compute foreground, background models

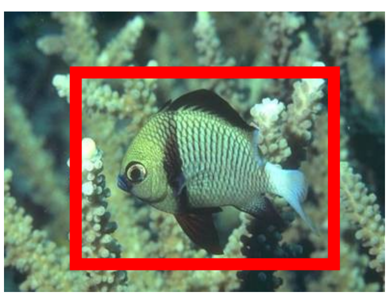
SEGMENTATION WITH GRAPH CUTS



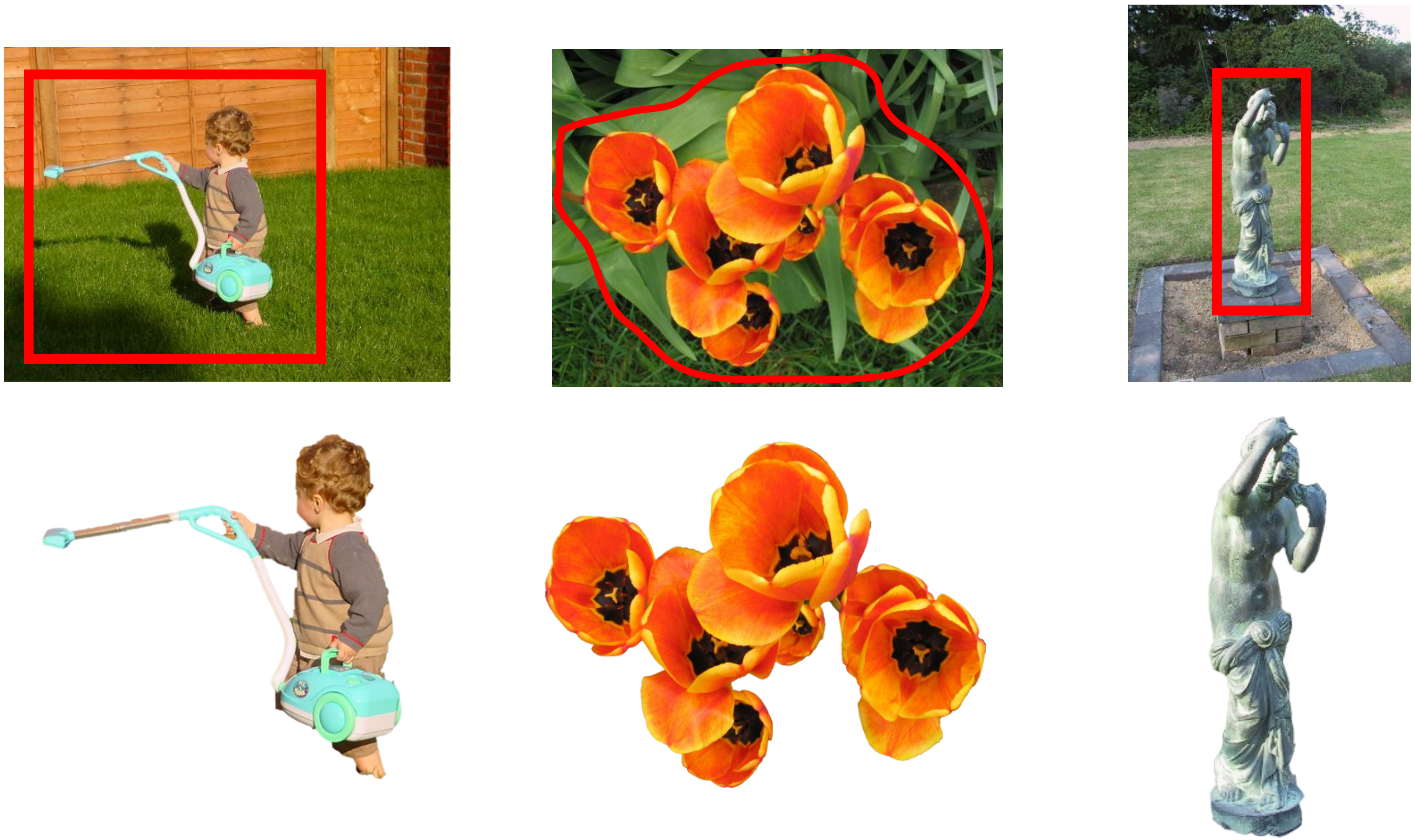
Which edges does min-cut cuts from the unary potential? And from the edge potential?

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



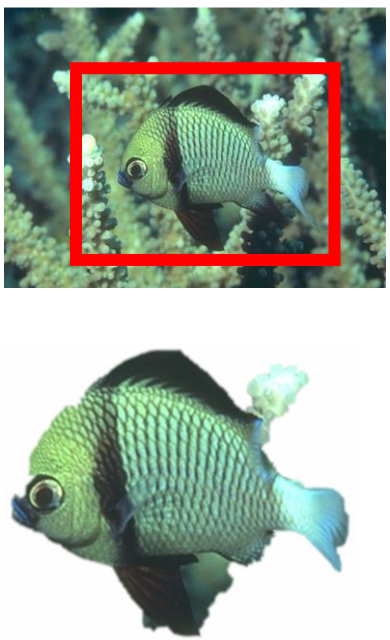


Easier examples

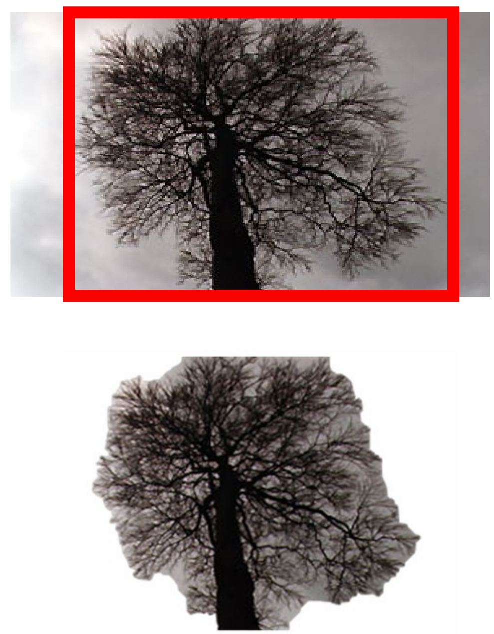


More difficult Examples

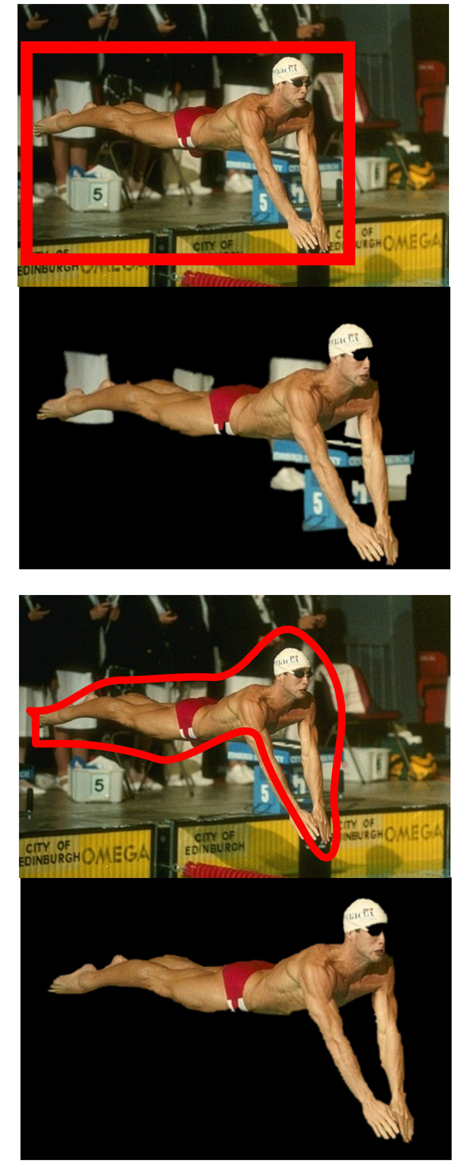
Camouflage & Low Contrast



Fine structure



Harder Case



Limitations of Graph Cuts

- Requires associative graphs
  - Connected nodes should prefer to have the same label
- Is optimal only for binary problems

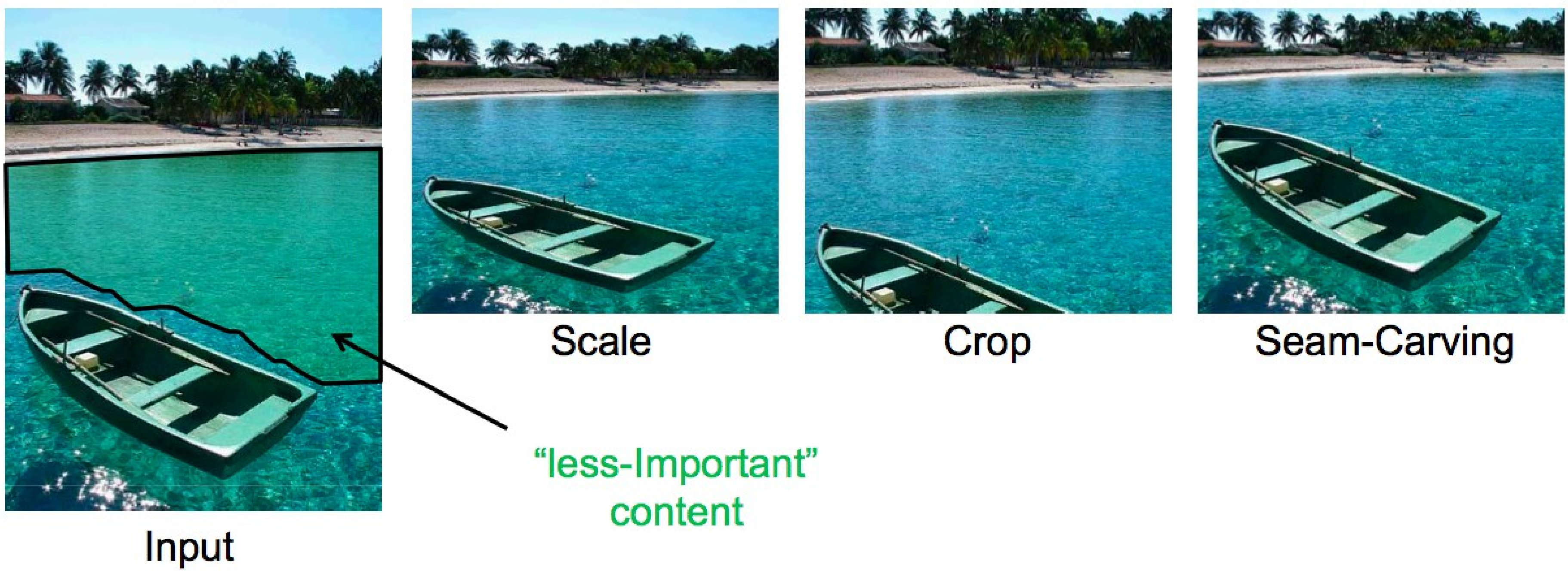
SEAM CARVING

Problem statement:

– Input Image I nxm, and new size n’xm’
– Output Image I’ of size n’xm’ which will be “good representative” of the original image I

- **To date, no clear definition, or measure, as to what a good representative is in this context!**

SEAM CARVING



SEAM CARVING

Basic Idea: remove unimportant pixels from the image

– Unimportant = pixels with less "energy"

$$E(I) = |\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|$$

Intuition for gradient-based energy:

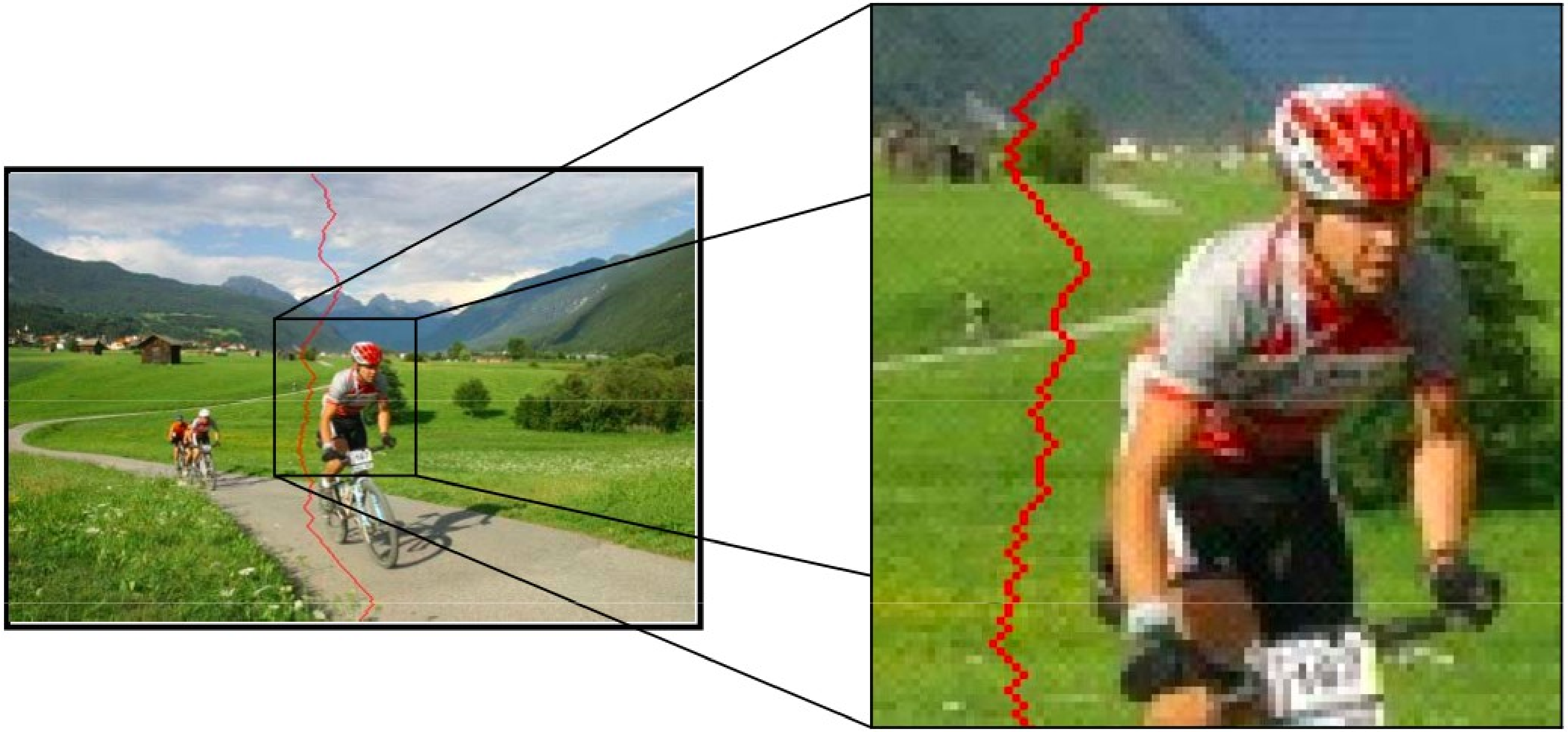
– Preserve strong contours

– Human vision more sensitive to edges – so try remove content from smoother areas

– Simple, enough for producing some nice results

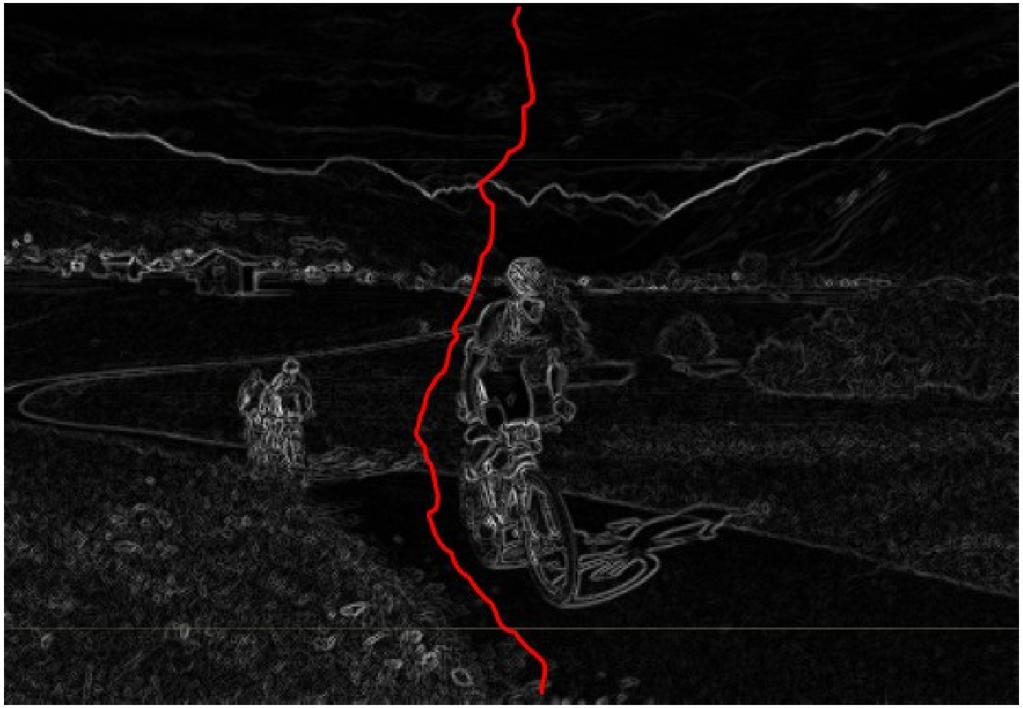
SEAM CARVING

Seam: A set of contiguous pixels from top to down or left to right with one single pixel per column or row, respectively



SEAM CARVING

Seam: A set of contiguous pixels from top to down or left to right with one single pixel per column or row, respectively



$$E(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \Rightarrow s^{*} = \arg\min_{s} E(s)$$

SEAM CARVING

Finding the optimal seam with dynamic programing:

$$M(i,j) = E(i,j) + min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

Step 1: Fill the matrix

5	8	12	3
9	2	3	9
7	3	4	2
4	5	7	8

5	8	12	3
9	2 + 5	3	9
7	3	4	2
4	5	7	8

SEAM CARVING

Step 2: Backtracking (or storing while creating)

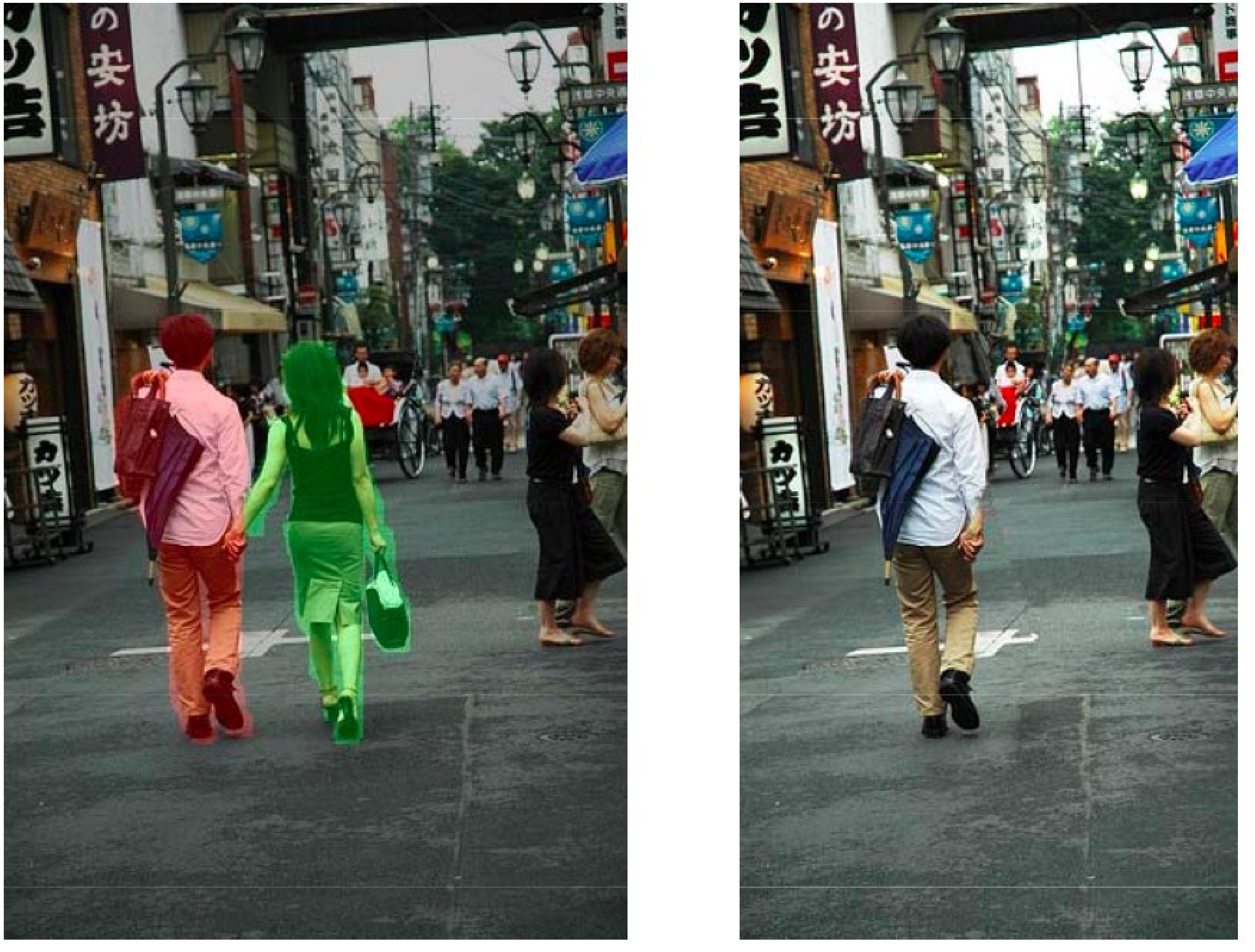
5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

SEAM CARVING: REMOVING OBJECTS



SEAM CARVING AND THE MISSING SHOE

SEAM CARVING AND IMAGE SYNTHESIS

SUMMARY OF BIG IDEAS

- Treat image as a graph
  - Pixels are nodes
  - Between-pixel edge weights based on gradients
  - Sometimes per-pixel weights for affinity to foreground/background
- Good boundaries are a short path through the graph (Intelligent Scissors, Seam Carving)
- Good regions are produced by a low-cost cut (GrabCuts, Seam Carving)