Graph-based image segmentation

Václav Hlaváč

Czech Technical University in Prague

Czech Institute of Informatics, Cybernetics and Robotics

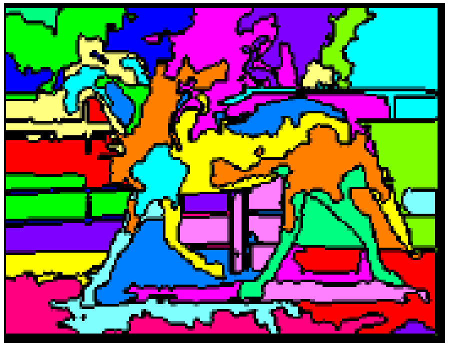
160 00 Prague 6, Jugoslávských partyzánů 1580/3, Czech Republic

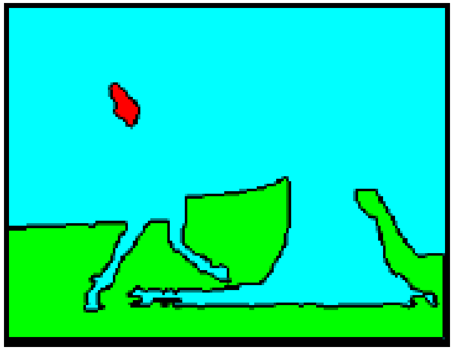
vaclav.hlavac@cvut.cz, http://people.ciirc.cvut.cz/hlavac/

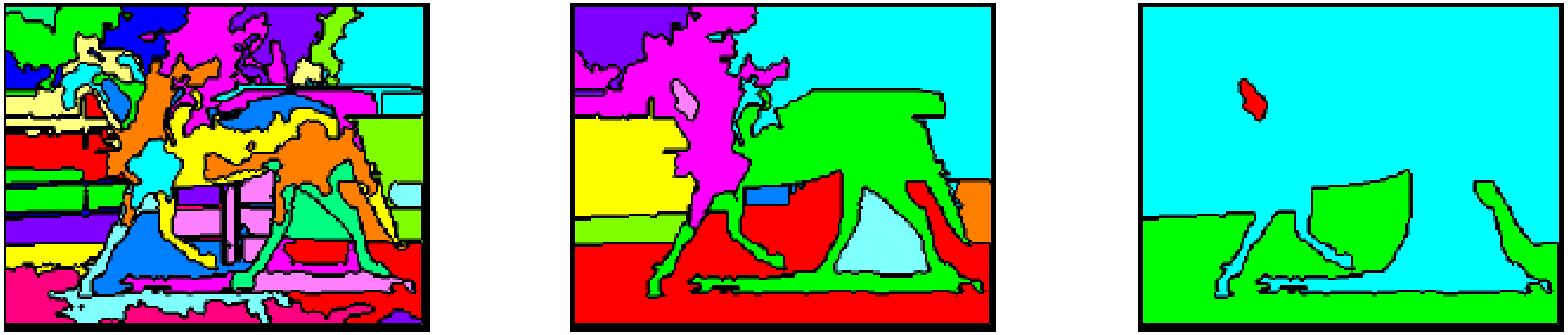
Courtesy D. Hoiem ,S. Lazebnik, Jianbo Shi, F. Malmberg, R. Krupička

Types of segmentations


Input image

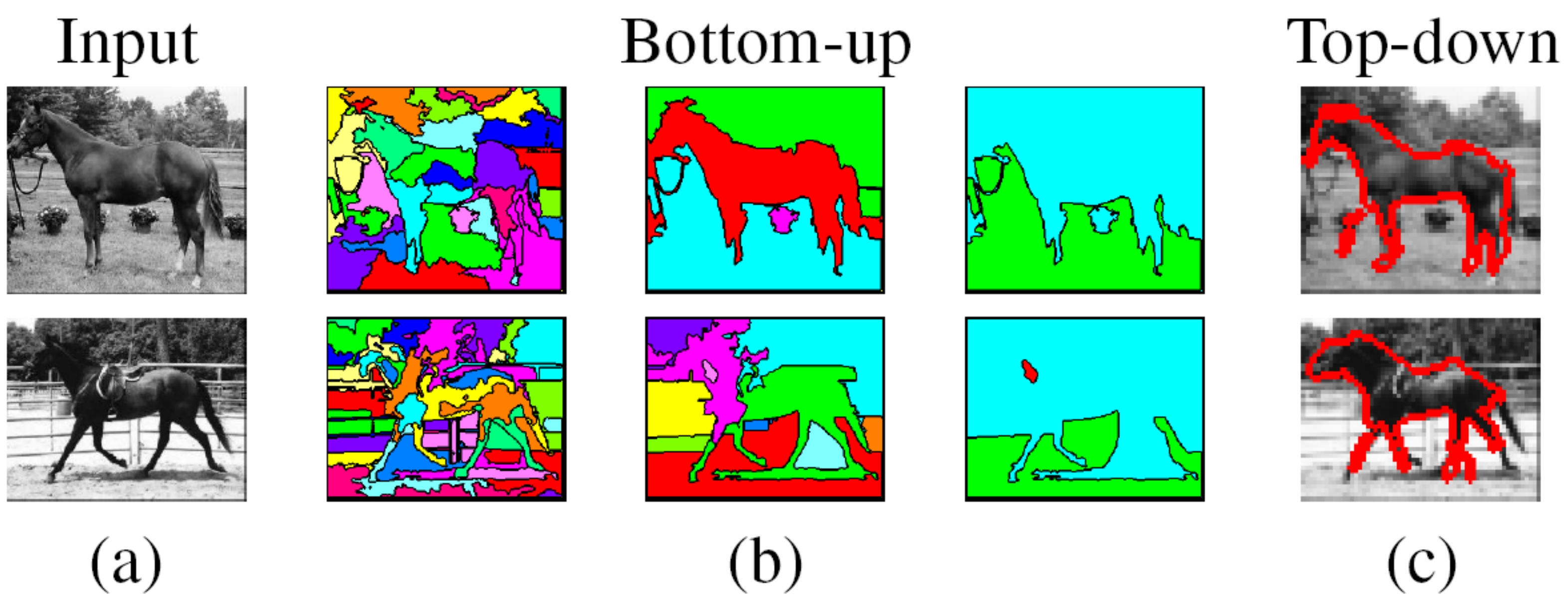

Oversegmentation


Undersegmentation


Multiple Segmentations

Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object



[Levin and Weiss 2006]

Graph-based image segmentation, main ideas

- Convert an image into a graph.
  - Graph vertices correspond to individual pixels.
  - Additional graph vertices and edges encode other constraints.
    Example: a special node (source) denotes objects and a special node (sink) denotes background in object/background segmentation.
    The source/sink concepts come from flow networks.
- Manipulate the graph to segment the image.

Image segmentation using graphs, seminal papers

- Y. Boykov, M.-P. Jolly: Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images, ICCV 2001.
  - A: Pixel classified as object or background. Novelty: adding interactivity.
  - Minimize energy function $E(A) = B(A) + \lambda R(A)$, where $B(A)$ = the cost of all edges between object pixels and background pixels; $R(A)$ = the cost of deciding if a pixel is object or background.
- P.F. Felzenszwalb, D.P. Huttenlocher: Efficient Graph-based Image Segmentation. International Journal of Computer Vision, 2004.
  - Cluster the vertices based on edge weight.
- C. Rother, V. Kolmogorov, A. Blake: GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04), 2004.

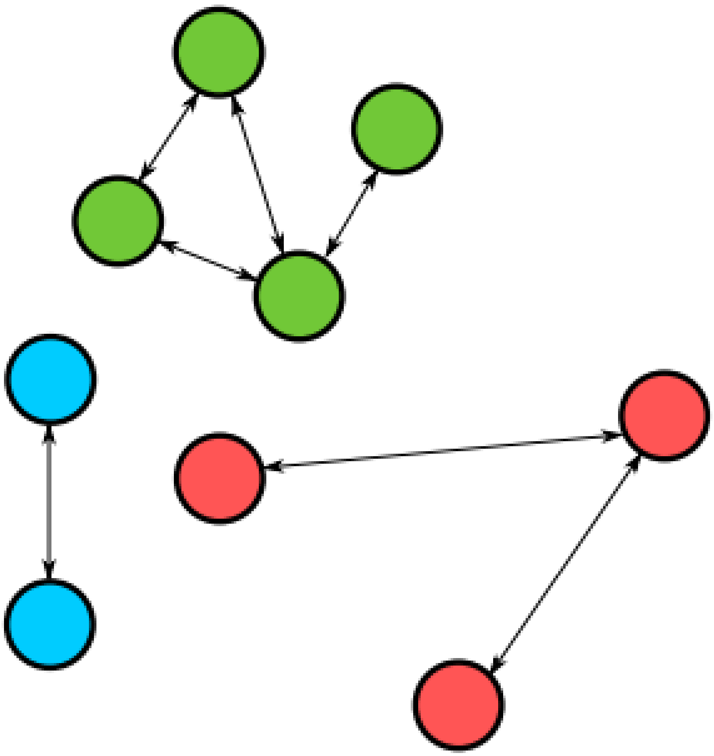
Subgraphs and connected components

- If G and H are graphs such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then H is a subgraph of G.

- If H is a connected subgraph of G and
  - $v \neq w$ in G for all vertices $v \in H$ and $w \notin H$,
  - (for any pair of vertices $v, w \in H$ it holds that $e_{v,w} \in E(H)$ if $e_{v,w} \in E(G)$),

  then H is a connected component of G.

Example, connected componens



A graph with three connected components.

Graph segmentation

- To segment an image represented as a graph, we want to partition the graph into a number of separate connected components.

- The partitioning can be described either as a vertex labeling or as a graph cut.

Vertex labeling

- We associate each vertex with an element in some set L of labels, e.g., $L = \{\text{object}, \text{background}\}$.

- *Definition, vertex labeling*

  A (vertex) labeling λ of $G(V,e)$ and labels L is a map $\lambda : V \rightarrow L$.

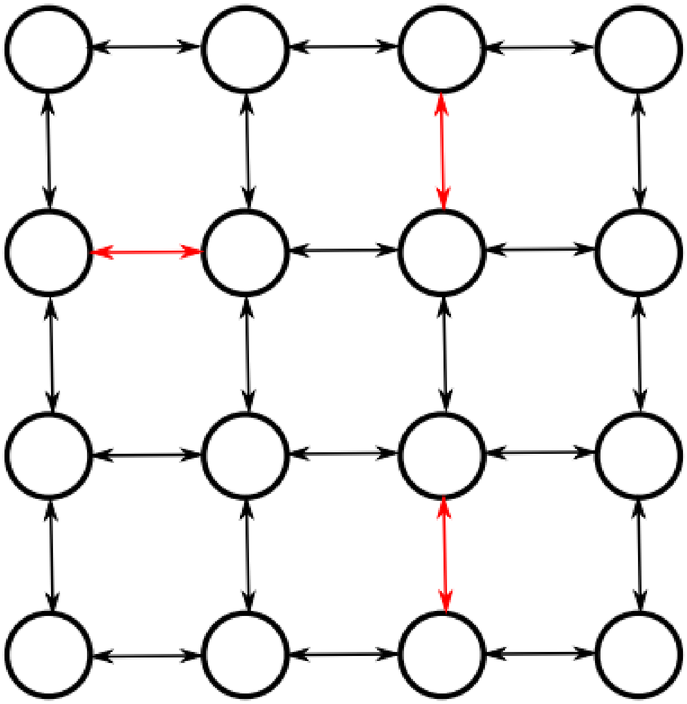
Graph cuts

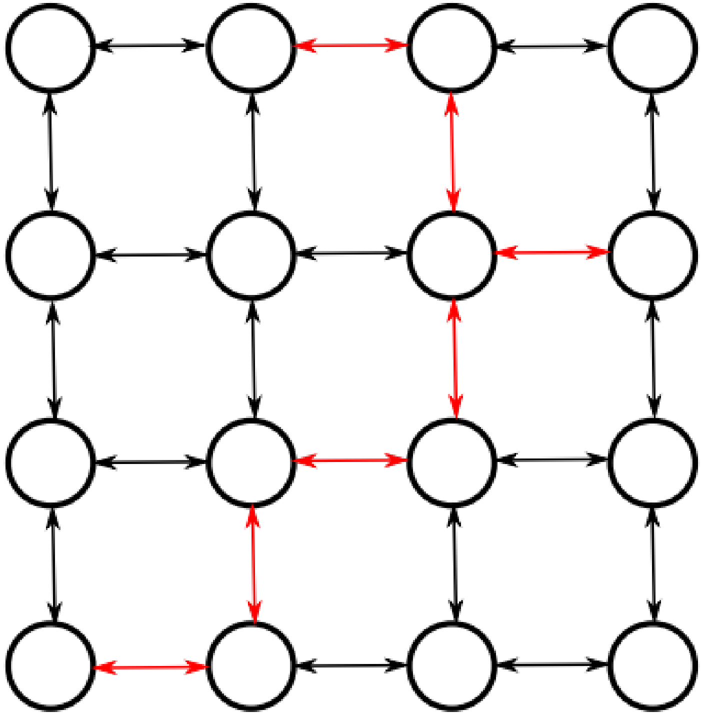
- Informally, a (graph) cut is a set of edges that, if they are removed from the graph, separate the graph into two or more connected components.

- *Definition, Graph cuts*
  - Let $S \subseteq E$, and $G' = (V, E \setminus S)$. If, for all $e_{v,w} \in S$, it holds that $v \neq w \in G'$, then S is a (graph) cut on G.

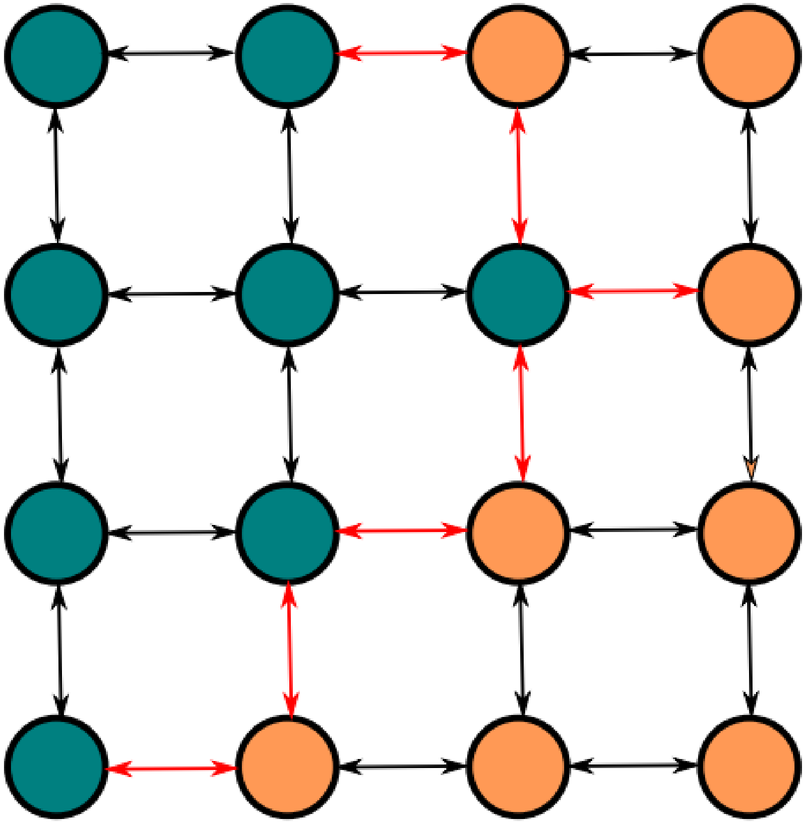
Example, cuts

- A set of edges (red) that
  - Do not form a cut
  - Form a cut





Relation between labelings and cuts



Graph-based image segmanation

- Useful graph algorithms
  - Minimal spanning tree

    Kruskal's algorithm, using for image segmentation
  - Shortest path

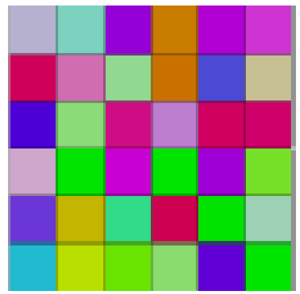
    Dijkstra's algorithm, using for intelligent scissors
  - Source S – Sink T; max flow or min cut

    Segmentating object from background

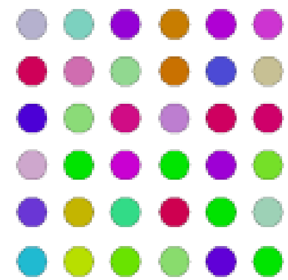
Main ideas

- Convert an image into a graph
  - Vertices for the pixels
  - Edges between the pixels
  - Additional vertices and edges to encode other constraints
- Manipulate the graph to segment the image

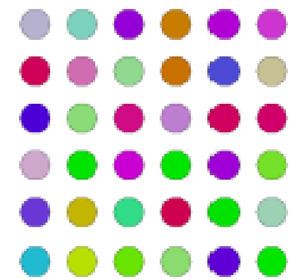
An image represented as a graph



image

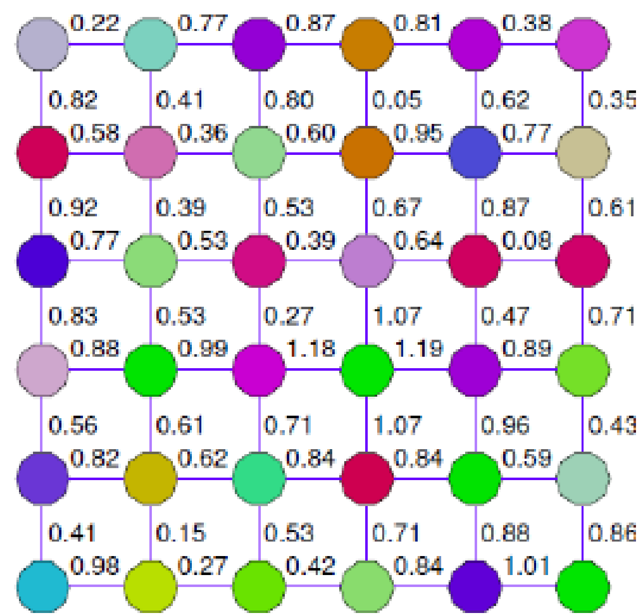


nodes



nodes and edges

- Nodes correspond to pixels.
- Edges connect neighboring pixels. 4-neighbors are considered in the example.
- Edges weights express the similarity between the neighboring pixels (binary relation).

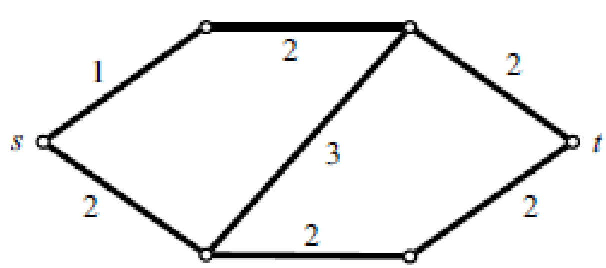


graph with weighted edges

Undirected/directed graph

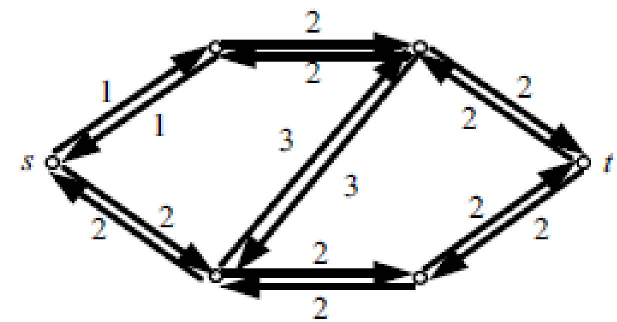
Undirected graph

- $G = (V, E)$ is composed of vertices V and undirected edges E representing a relation between two vertices.
- If a weight w_e is assigned to all edges then the graph becomes undirected weighted graph.



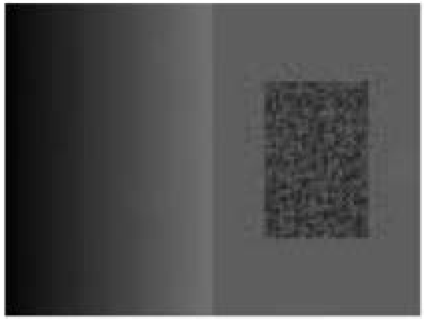
Directed graph

- $G = (V, E)$ is composed of vertices V and directed edges E representing an ordered relation between two vertices.
- Oriented edge $e = (u, v)$ has the tail u and the head v (denoted by the arrow). The edge e is different from $e' = (v, u)$ in general.

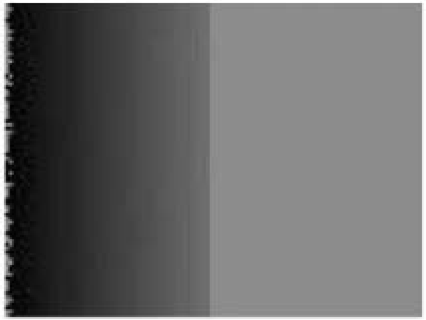


Graph based image segmentation

- Bottom-up segmentation



Original Image



Incorrect Segmentation

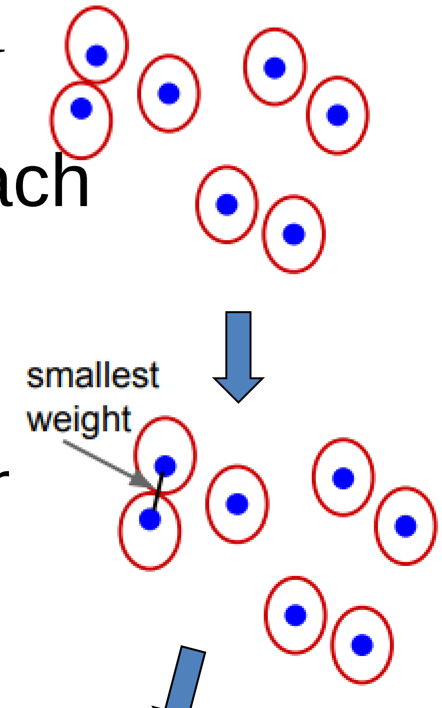
Correct Segmentation

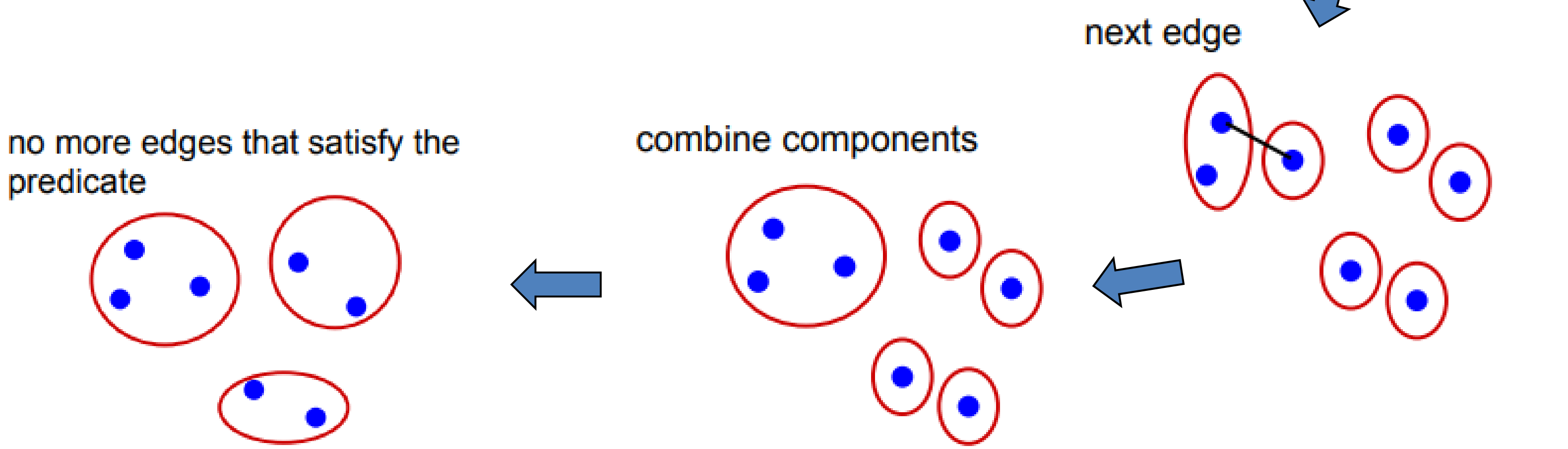
- Based on Kruskal’s minimum-spanning-tree algorithm

Graph based image segmentation

Define $G(V,E)$ and maximal distance M

1. Start with segmentation S_0, where each vertex v_i is in its own component
2. Merge nearest components
3. Repeat step 3 until distance betweer components is lesser than M





Grid graph based

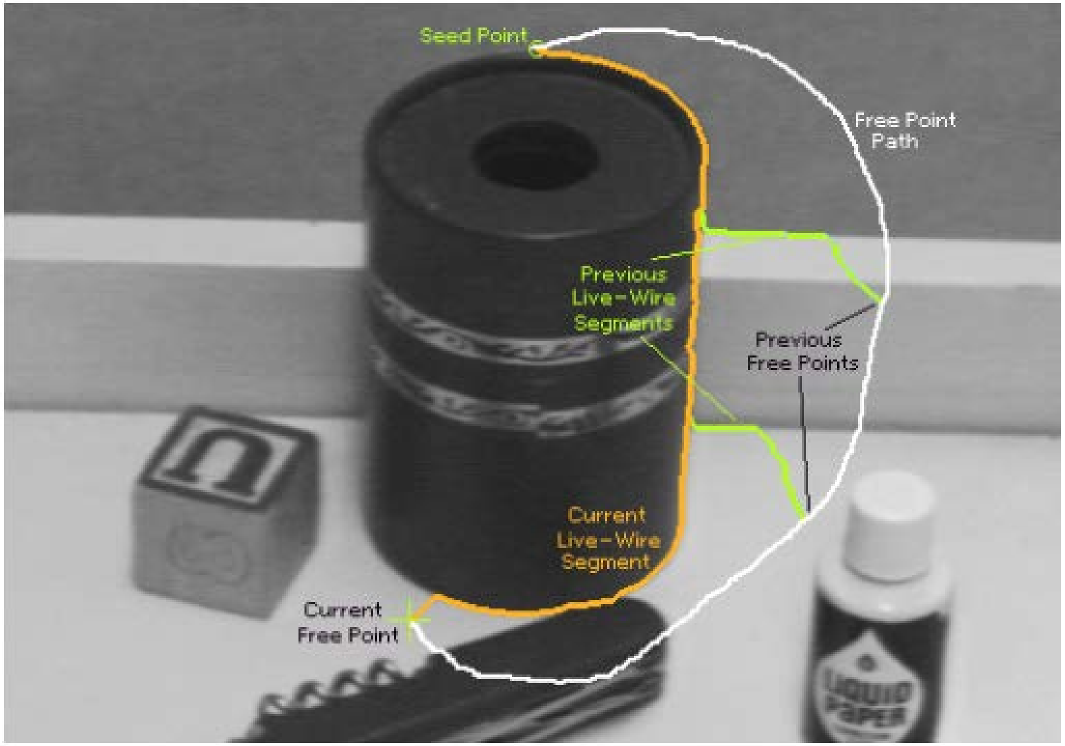
- Every pixel is connected to its 4 neighboring pixels
- Weights are determined by the difference in intensities
  - For color images - the algorithm runs three times using R values, then using G values and finally B values. Two pixels in the same component only if they appear in the same component in all three colors.
- Features
  - Preserves small components, doesn’t have problem with small changes in gradient

Nearest-neighbor image segmentation

- Project every pixel into feature space defined by (x,y,r,g,b)
- Weight between pixels ale determined using Euclidian distance
- Edges are chosen for only top 10 nearest neighbors in feature space
- Features
  - Non Spatially connection regions of the image can be placed in the same component. (see flowers or tower and lights)

Using shortest path algorithm

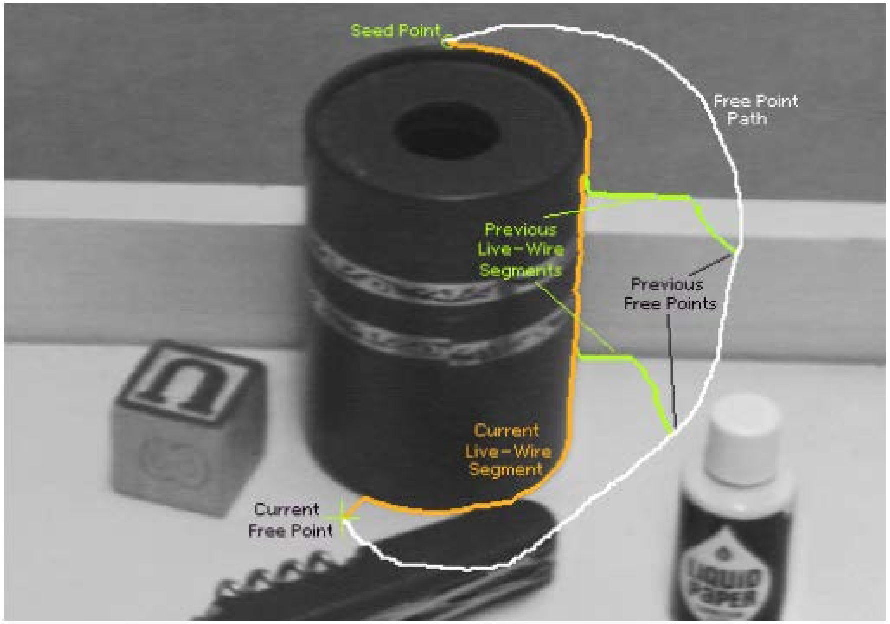
- Dijkstra's shortest path algorithm
- Used for intelligent scissors



Mortenson and Barrett (SIGGRAPH 1995)

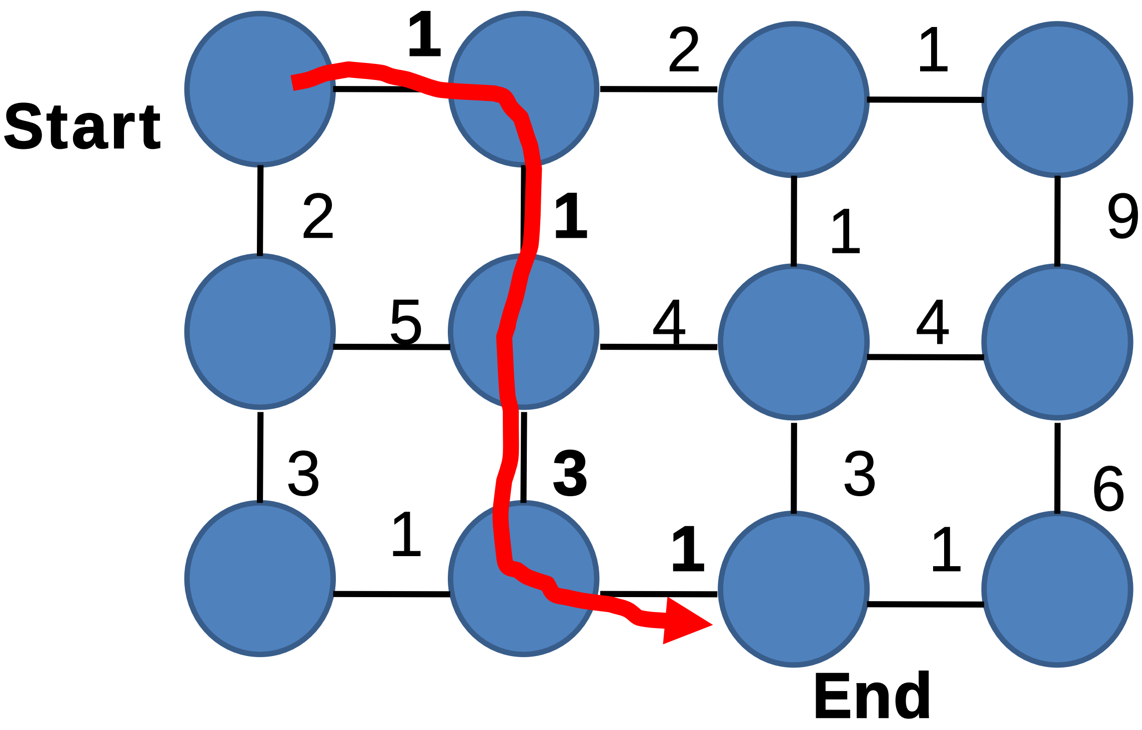
Intelligent scissors

- Formulation: find good boundary between seed points
- Challenges
  - Minimize interaction time
  - Define what makes a good boundary
  - Efficiently find it



Intelligent scissors

A good image boundary has a short path through the graph.



Mortenson and Barrett (SIGGRAPH 1995)

Dijkstra's shortest path algorithm

Initialize, given seed *s*:

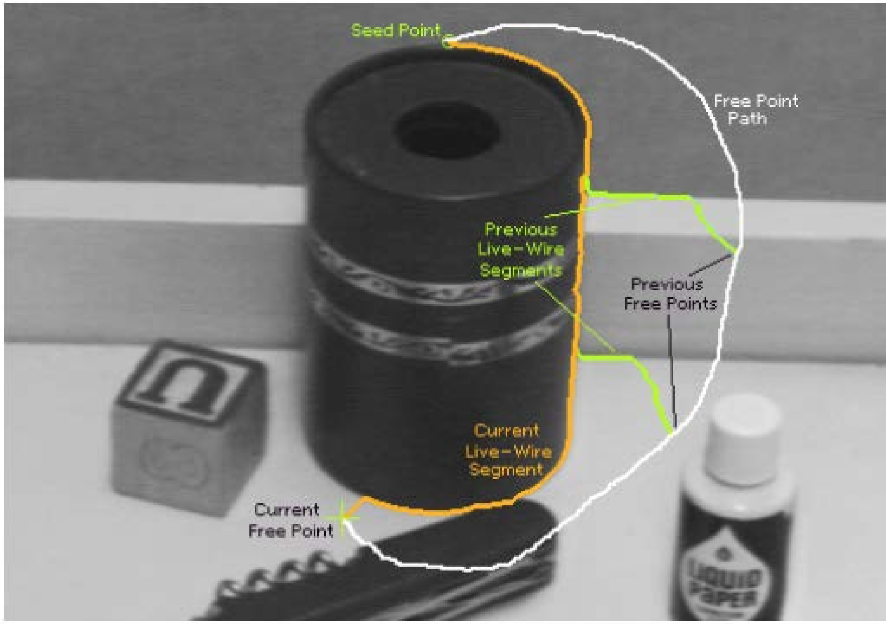
- Compute $cost_2$(q, r) % cost for boundary from pixel *q* to neighboring pixel *r*
- cost(*s*) = 0 % total cost from seed to this point
- **A** = {*s*} % set to be expanded
- **E** = { } % set of expanded pixels
- **P**(q) % pointer to pixel that leads to *q*

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*) + $cost_2$(*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)

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, choose new seed, repeat



Intelligent scissors: method (2)

Define boundary cost between neighboring pixels

a) Lower if edgel is present (e.g., with edge(im, ‘canny’))

b) Lower if gradient magnitude is strong

c) Lower if gradient direction matches the boundary

Gradients, edgels, and path cost

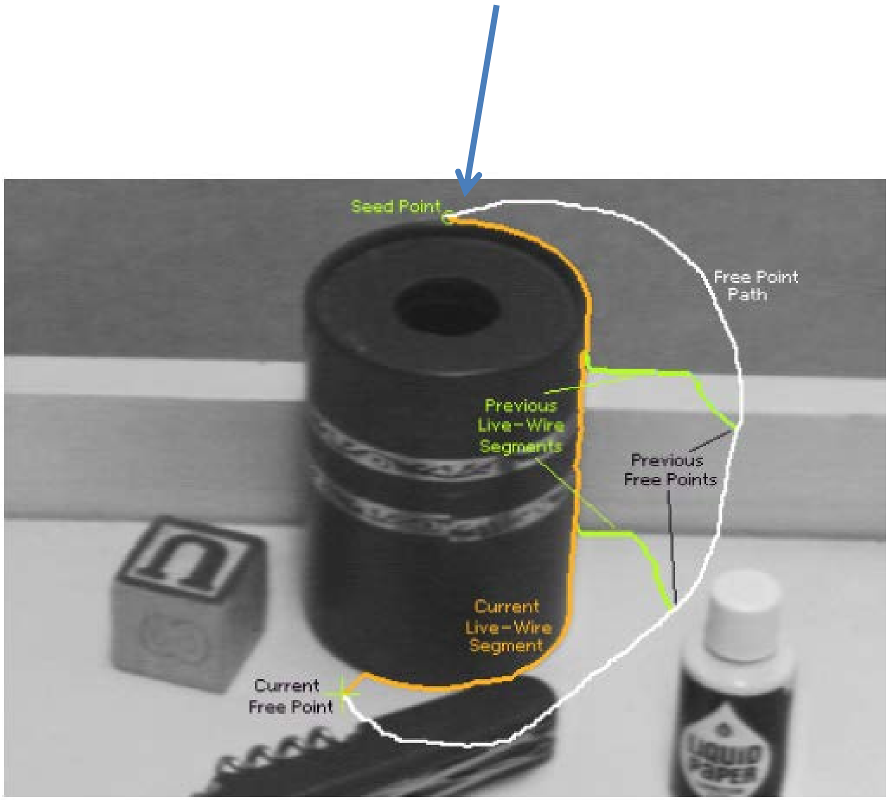
Gradient magnitude

Edgel image

Path cost

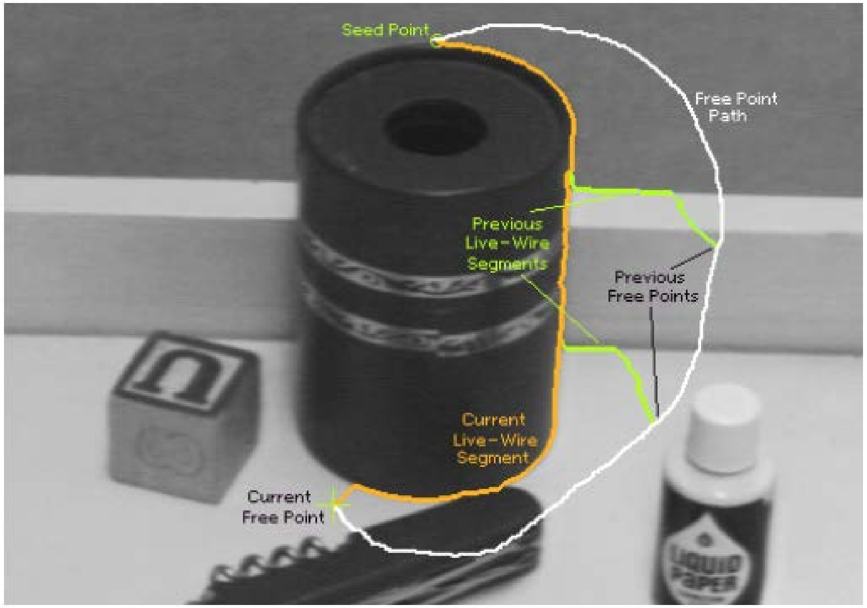
Intelligent scissors: method (3)

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



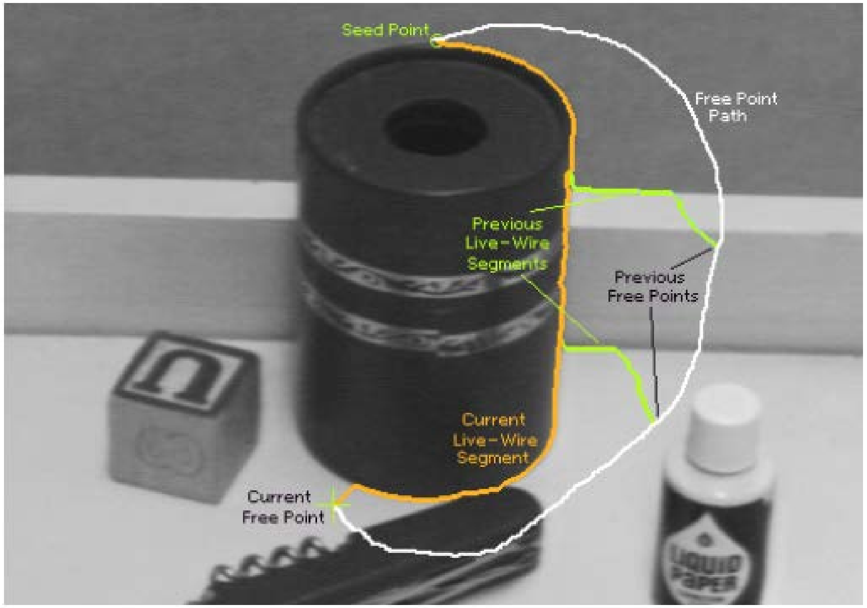
Intelligent scissors: method (4)

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



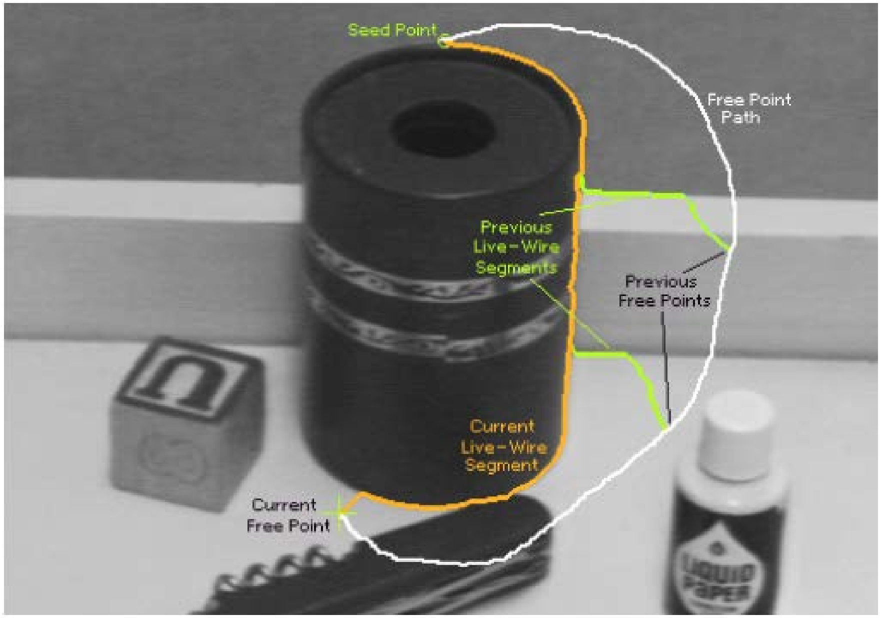
Intelligent scissors: method (5)

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



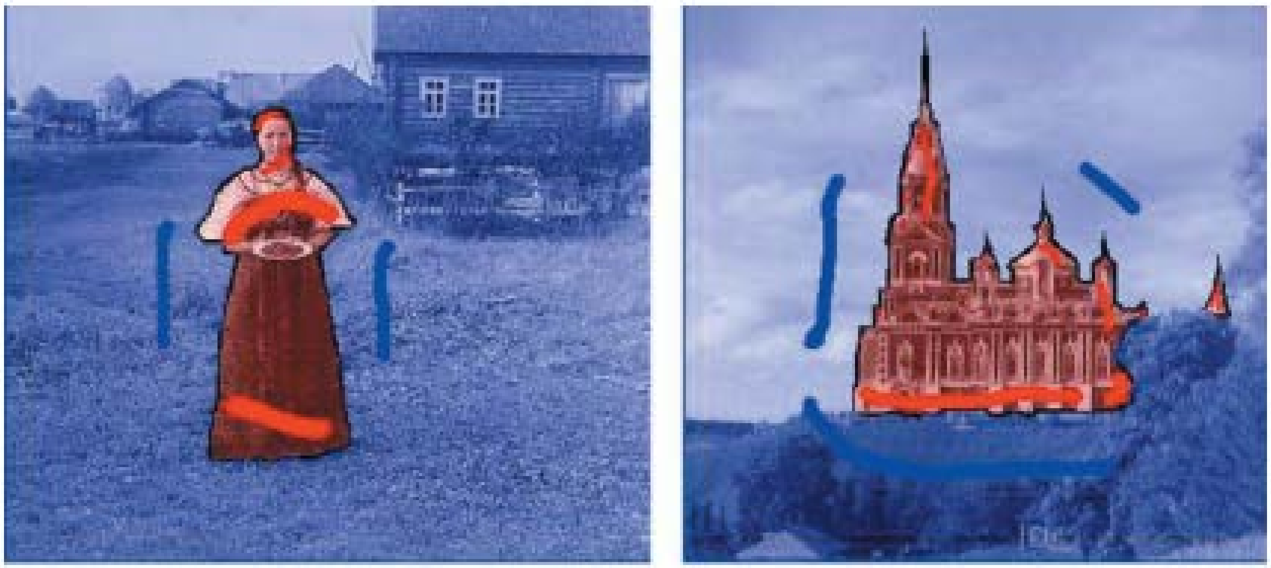
Intelligent scissors: improving interaction

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



Using minimal graph cuts in image segmantation

- Main aim is to segment the object from background
  - User defines „seeds“ for object and background



(a) A woman from a village (b) A church in Mozhaisk (near Moscow)

Flow network, flow

- A flow network is a directed graph with nonnegative edge weights (called also capacities).
- A flow is a real-valued (often integer) function, which satisfies the following three properties:

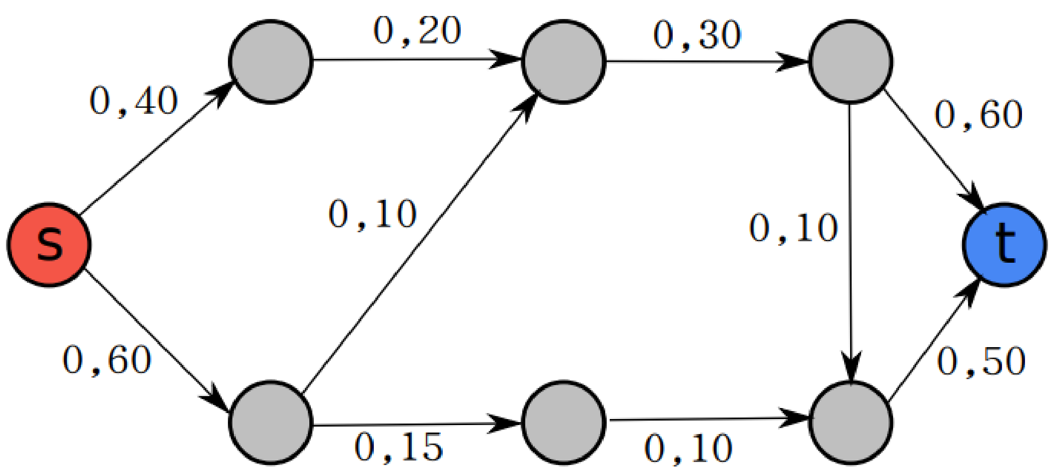
1. Capacity c constraint
   For all $u, v \in V$, $f(u, v) \le c(u, v)$.
2. Skew symmetry
   For all $u, v \in V$, $f(u, v) = -f(v, u)$.
3. Flow conservation
   For all $u \in (V \setminus \{s, t\})$, $\sum_{v \in V} f(u, v) = 0$.

A cut of a graph

- A cut is a set of edges $C \subset E$ such that two vertices (called terminals) became separated on the induced graph $G' = (V, E \setminus C)$.
- Denoting a source terminal as s and a sink terminal as t, a cut (S, T) of $G = (V, E)$ is a partition of V into S and $T = V \setminus S$, such that $s \in S$ and $t \in T$.

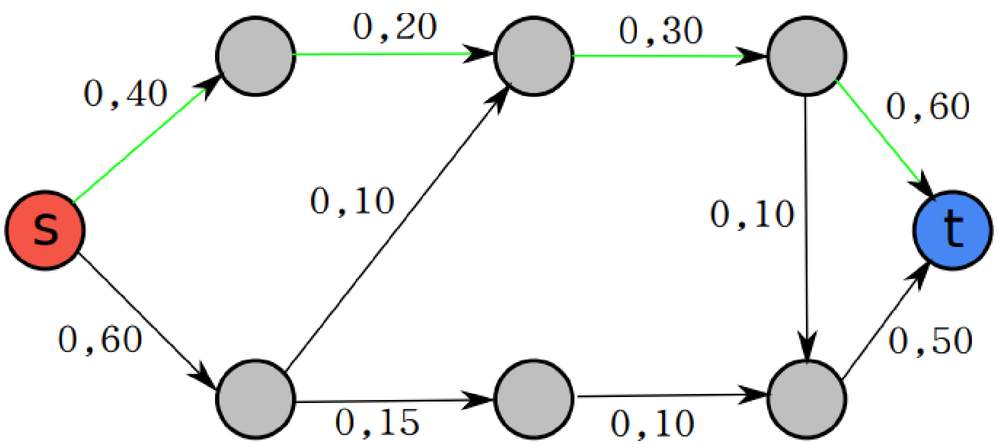
Max flow (1)

- Directed graph with one source & one sink node
- Directed edge = pipe
- Edge label = capacity
- What is the max flow from source to sink?



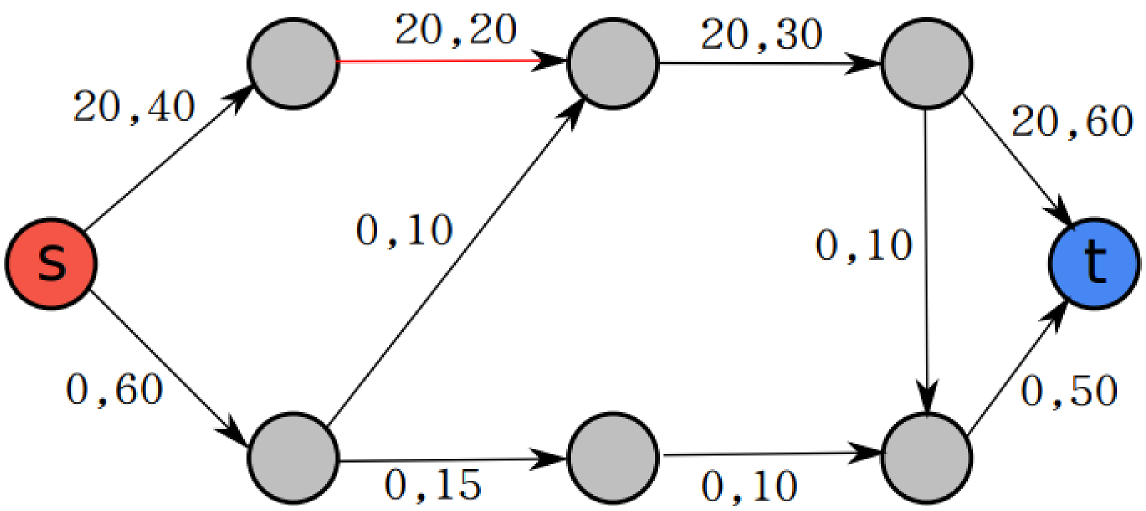
Max flow (2)

- Graph with one source & one sink node
- Edge = pipe
- Edge label = capacity
- What is the max flow from source to sink?
- 1st step: find any path with free capacity
  - Path can go in the opposite way if there is any flow. The value of the flow is then substracted.



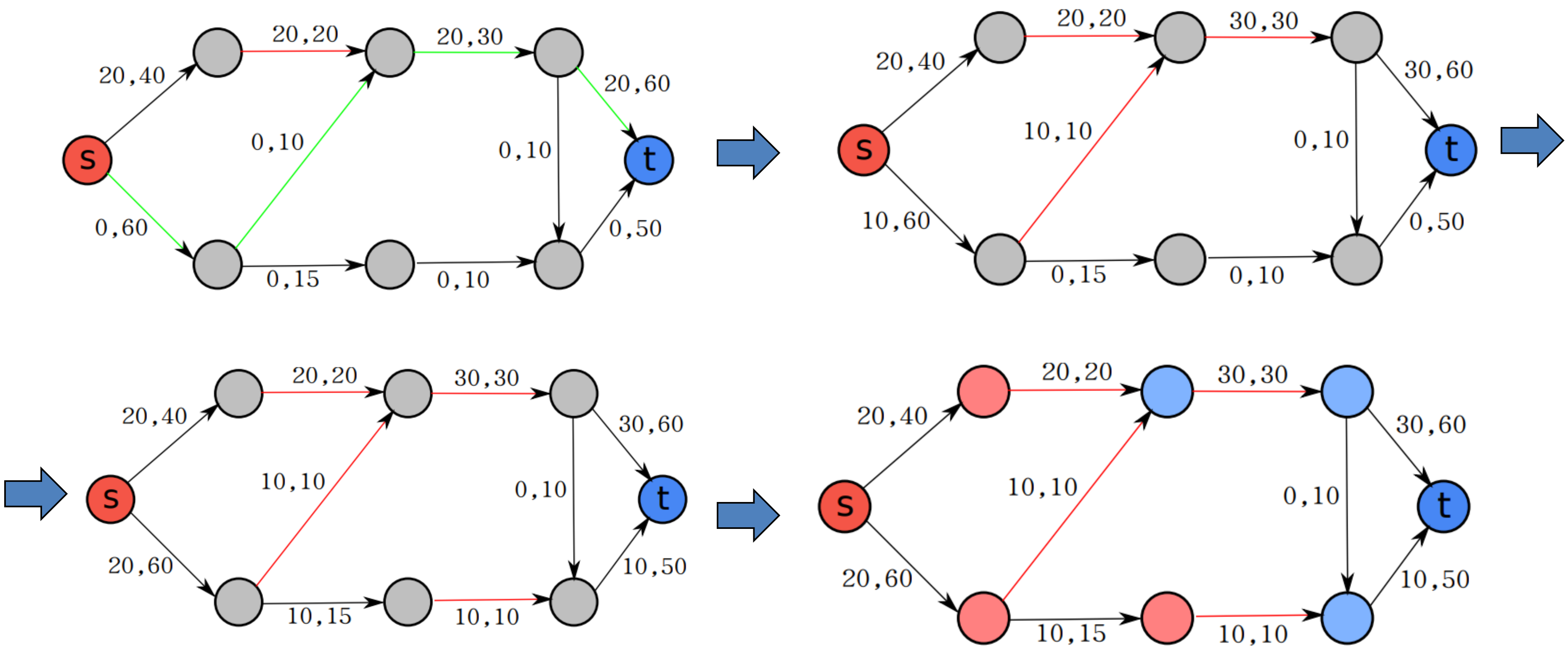
Max flow (3)

- 2nd step: Fill the path with the maximal capacity



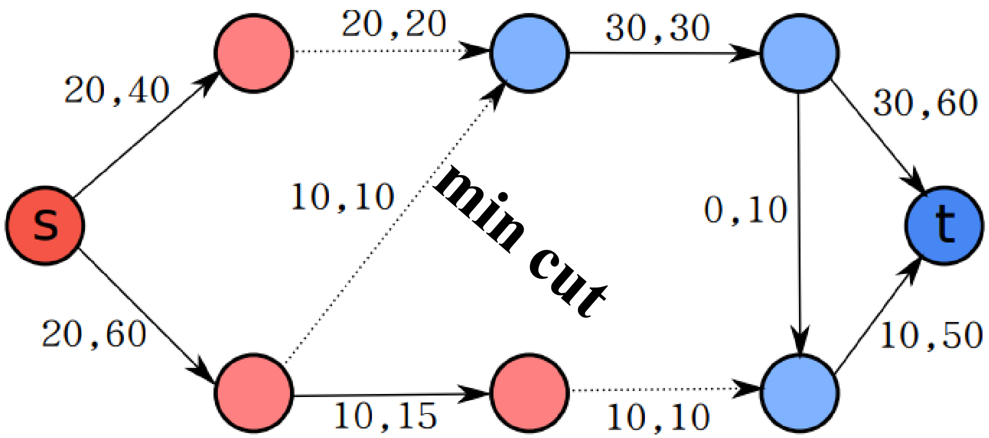
Max flow (4)

- 3rd step: return to step 1 until there is no free path



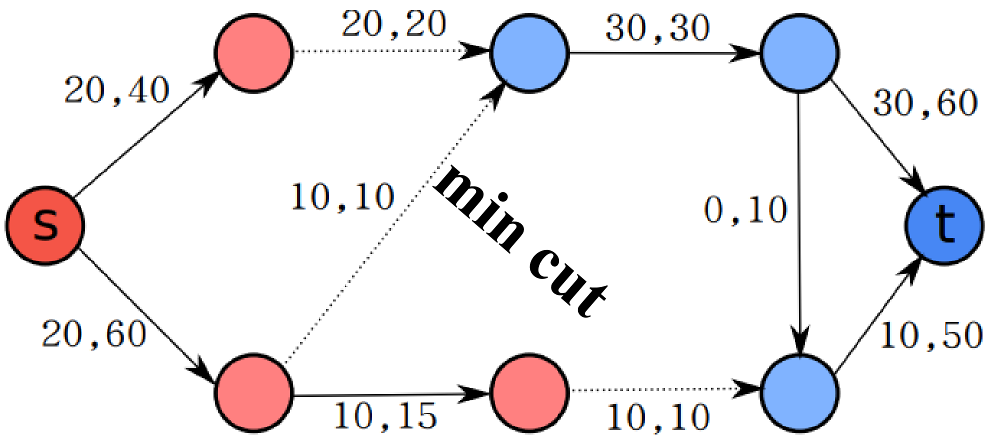
Max flow (5)

- What is the max flow from the source *s* to sink *t* ?
- Look at a residual graph
  - min cut is at the boundary between two connected components

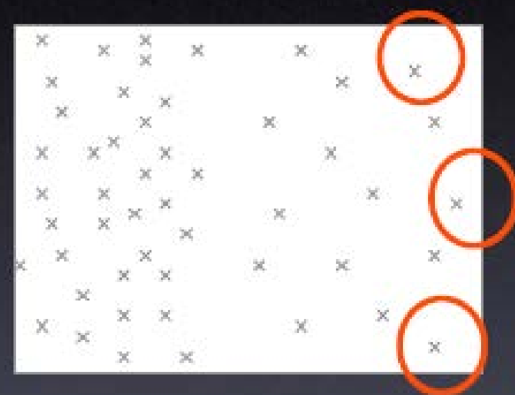


Equivalence of min cut and max flow

- The three following statements are equivalent
  - The maximum flow is *f*
  - The minimum cut has weight *f*
  - The residual graph for flow *f* contains no directed path from source to sink



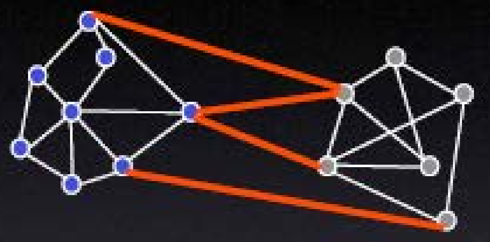
Problem with min cuts



Min. cuts favors isolated clusters

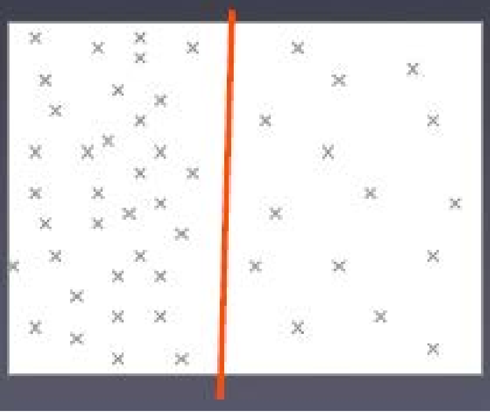
Normalize cuts in a graph

- (edge) Ncut = balanced cut

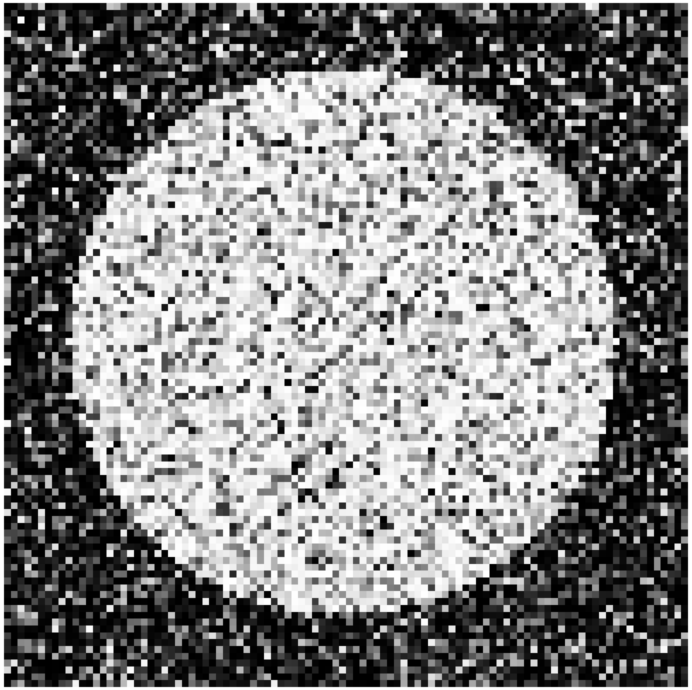


$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

NP-Hard!



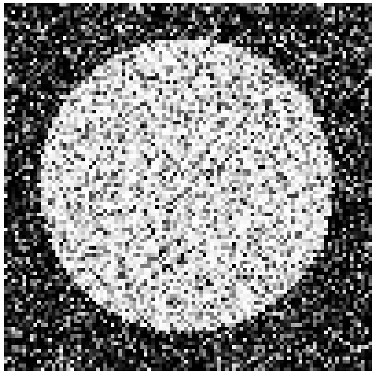
Pixel-based statistical model



P(foreground | image)

has limitations because of existing relations to other pixels.

Solution: encode dependences between pixels



P(foreground | image)

Normalizing constant called “partition function”

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

Labels to be predicted

Individual predictions

Pairwise predictions

Writing likelihood as an energy

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

$$\Big\Downarrow \ -\log(.)$$

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

Cost of assignment y_i

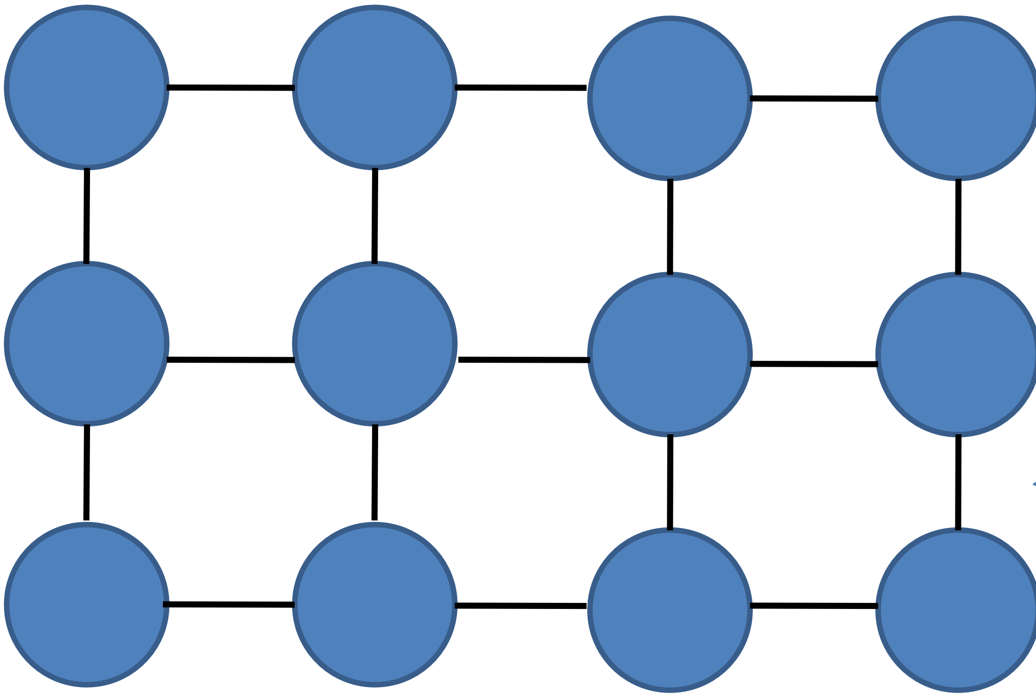
Cost of pairwise assignment $y_i\ , y_j$

Notes on energy-based formulation

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

- Primarily used when one only cares about the most likely solution (not the confidences).
- Can think of it as a general cost function.
- Can have larger “cliques” than 2. The clique is the set of variables that go into a potential function.

Markov Random Fields



Node y_i: pixel label

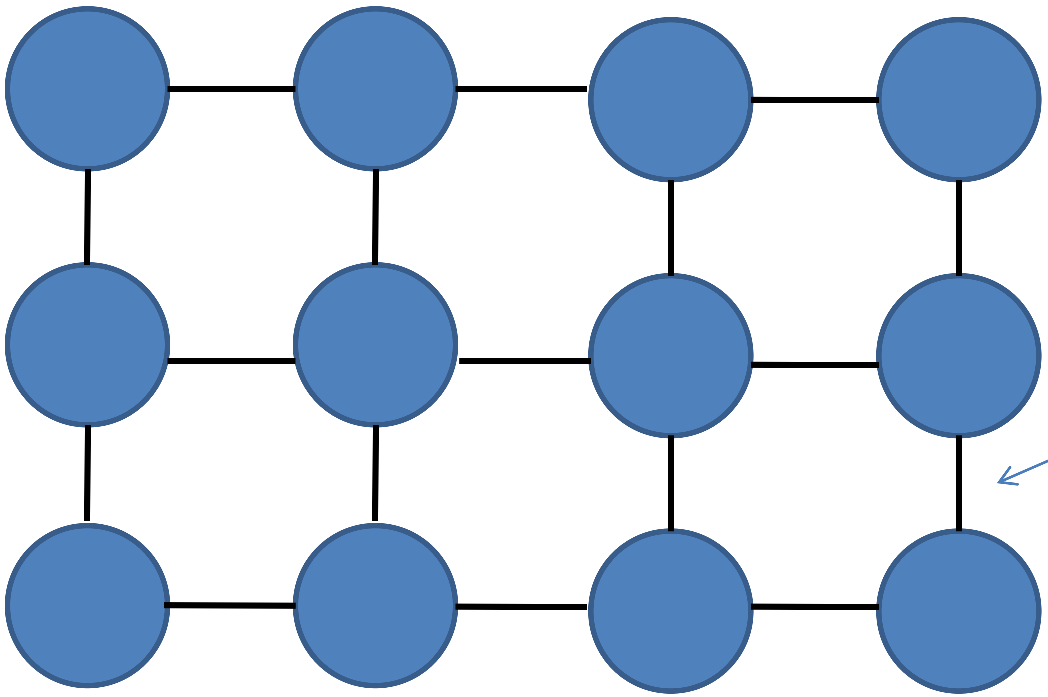
Edge:
constrained pairs

Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

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

Label smoothing grid example



Unary potential

$0\text{: } -\log P(y_i = 0 \mid data)$

$1\text{: } -\log P(y_i = 1 \mid data)$

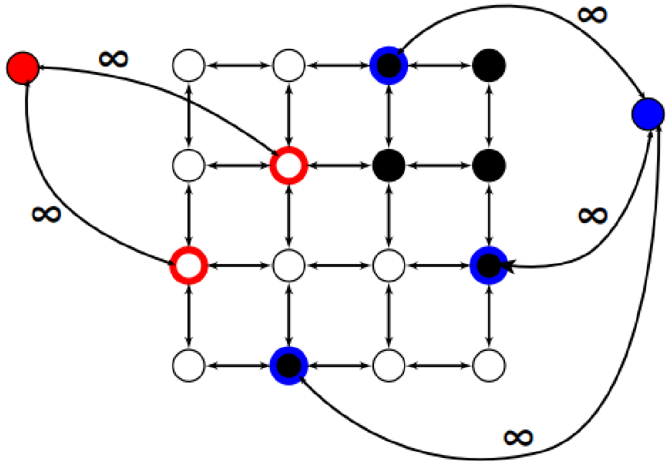
Pairwise Potential

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & K \\ 1 & K & 0 \end{array} \qquad K>0$$

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

Creating the graph from image

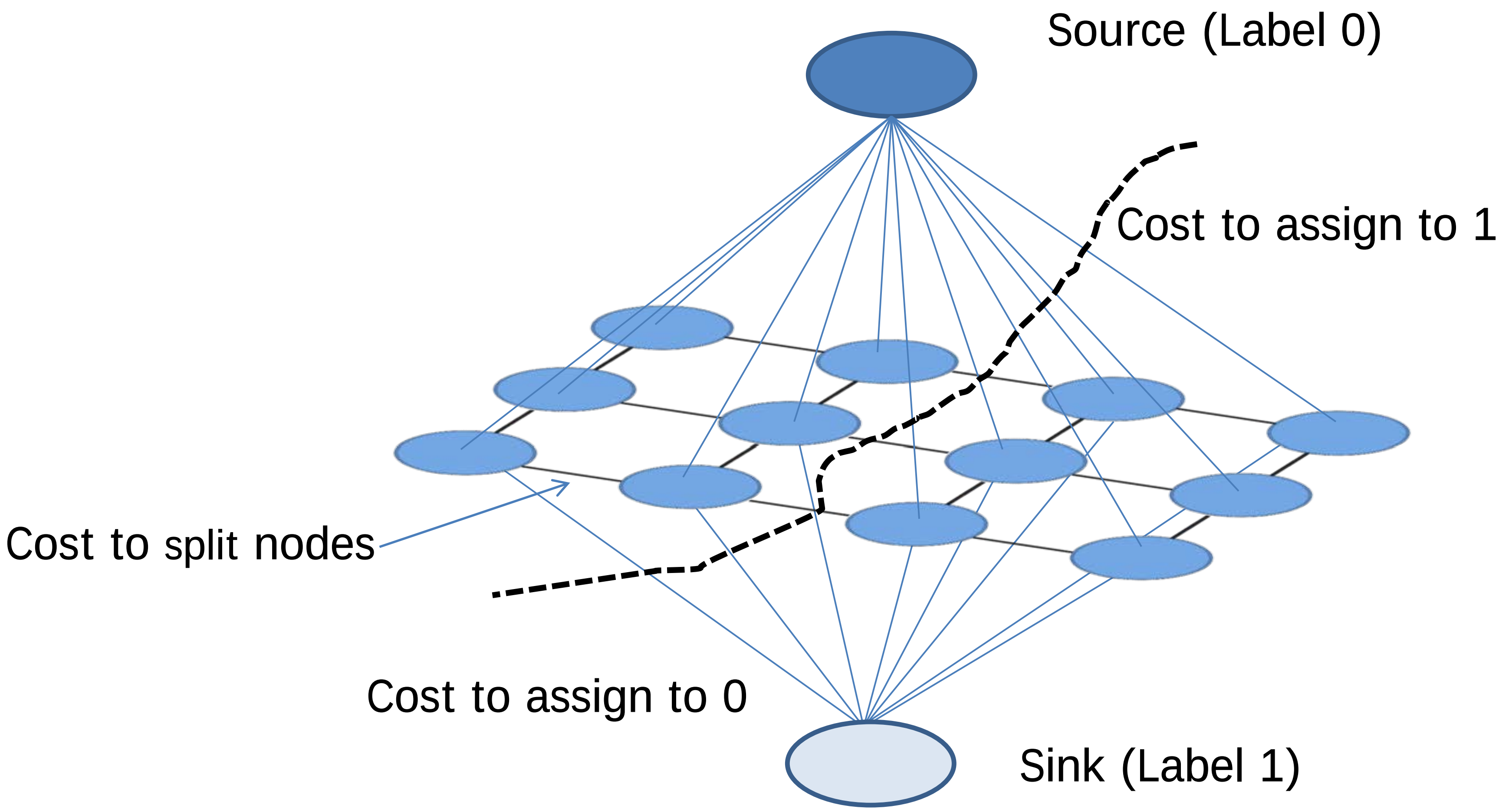
- Each pixel has a corresponding vertex
- Additionally, a source (“object”) and a sink (“background”)
- Each pixel vertex has an edge to its neighbors (e.g. 4 adjacent neighbors in 2D), an edge to the source, an edge to the sink
- Pixels connected with seeds have an infinite capacity



Edge weights between pixels

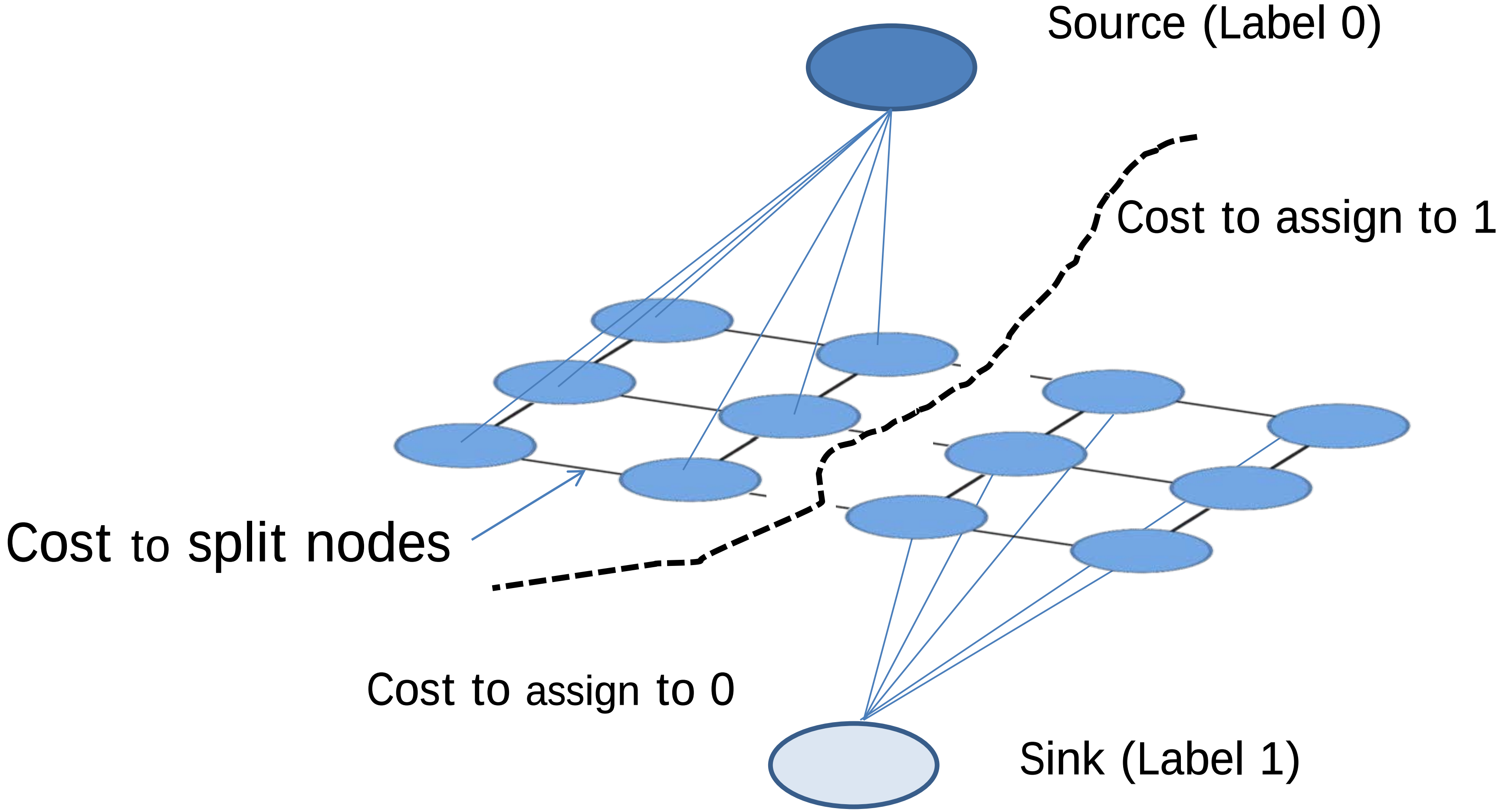
- Weight of edges between pixel vertices are determined by the function expressing dependence between two pixels
- Low score when boundary is likely to pass between the vertices
- High score when vertices are probably part of the same element
- E.g. the difference in pixel intensities, the image gradient

Solving MRFs with graph cuts



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

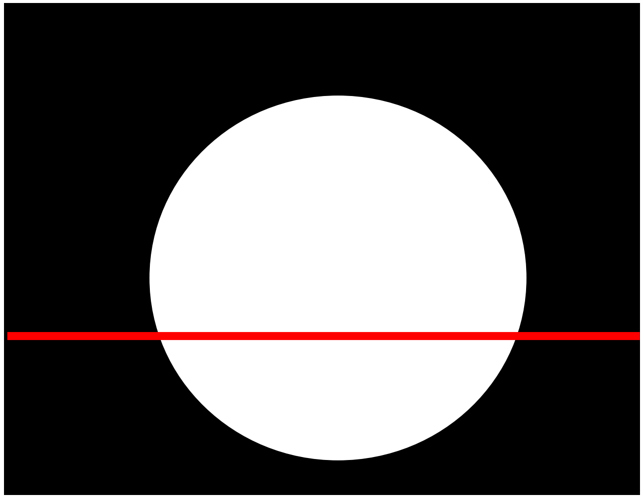
Solving MRFs with graph cuts

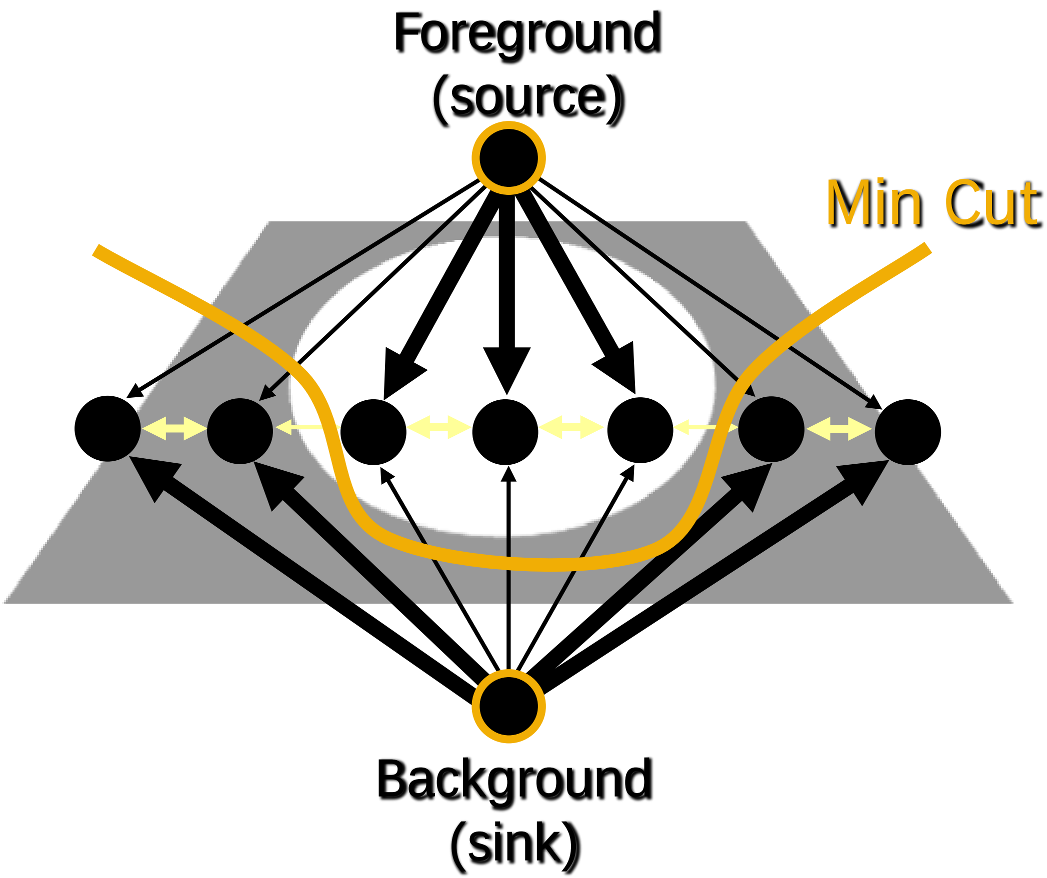


$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Graph cut, Boykov & Jolly 2001

Image

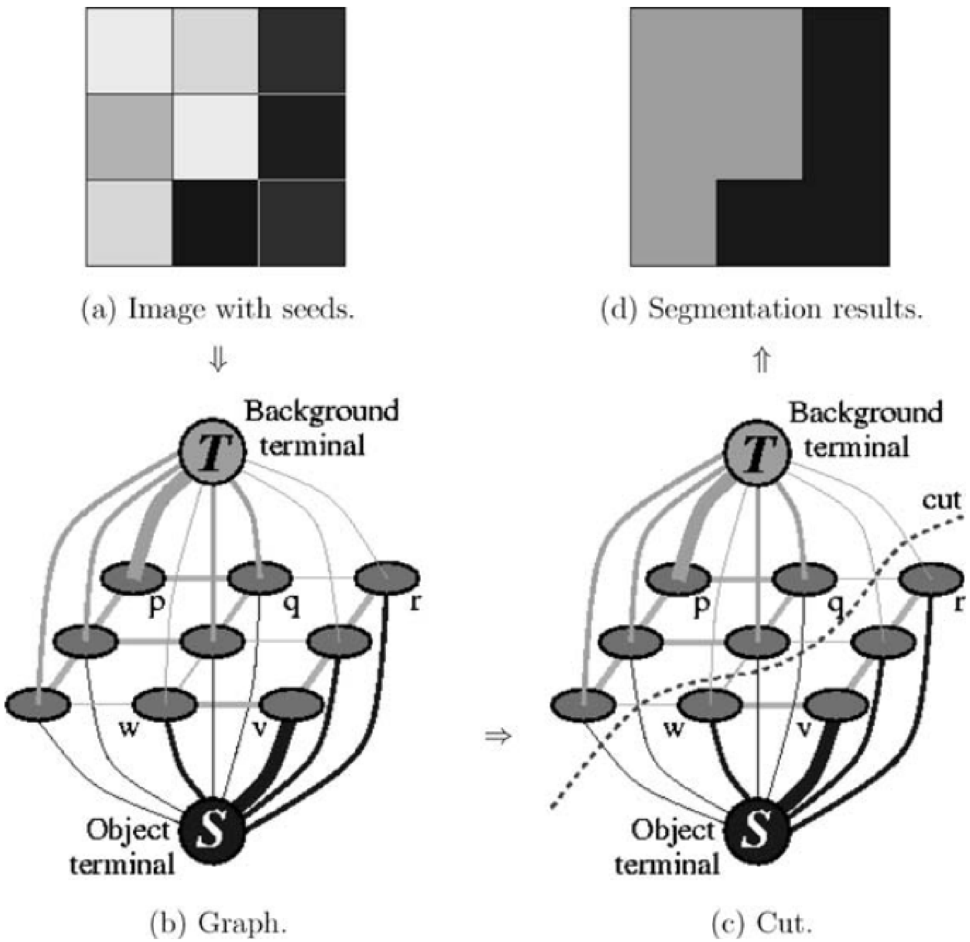




Cut: Separating source and sink; Energy: collection of edges

Min Cut: Global minimal energy in polynomial time

Minimal graph cuts and image labeling



Minimum graph cut segmentation of a 3x3 image. [Boykov and V. Kolmogorov]

Normalized cut

- A minimum cut penalizes large segments
- This can be fixed by normalizing the cut by component size
- The *normalized cut* cost is:

$$\frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

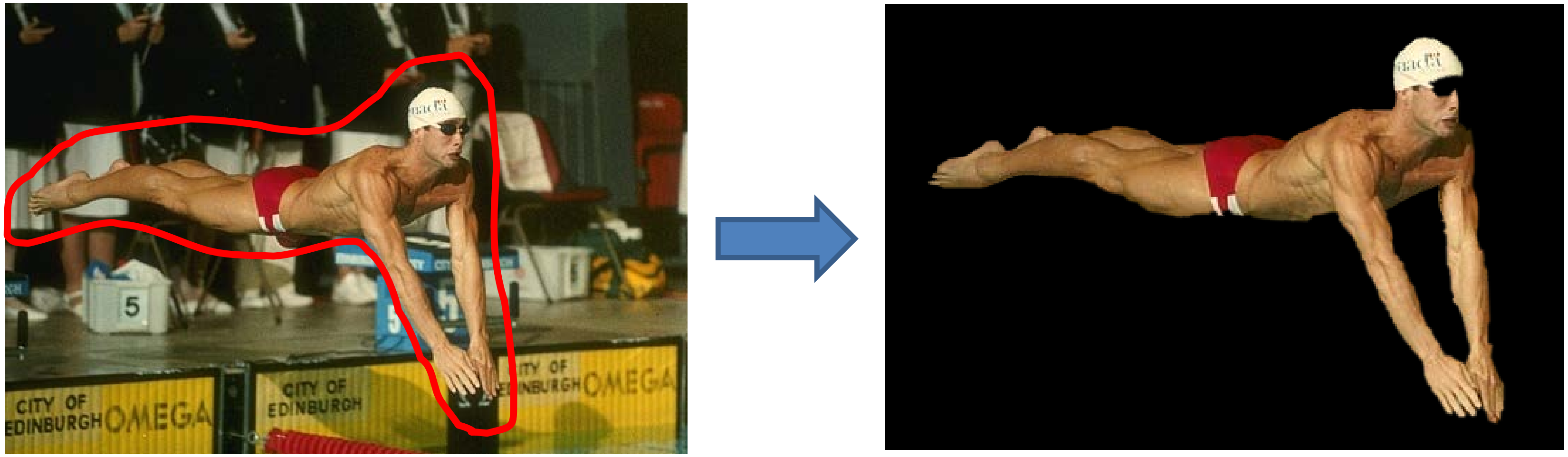
assoc(*A*, *V*) = sum of weights of all edges in V that touch A

- The exact solution is NP-hard but an approximation can be computed by solving a *generalized eigenvalue* problem

J. Shi and J. Malik. [Normalized cuts and image segmentation.] PAMI 2000

GrabCut segmentation

Carsten Rother et al. 2004



User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

- Less user input, rectangle only.
- Handles color

GrabCut segmentation

1. Define graph
   - usually 4-connected or 8-connected
     - Divide diagonal potentials by sqrt(2)
2. Define unary potentials
   - Color histogram or mixture of Gaussians for background and foreground

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

3. Define pairwise potentials

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

4. Apply graph cuts
5. Return to 2, using current labels to compute foreground, background models

GrabCuts and graph cuts

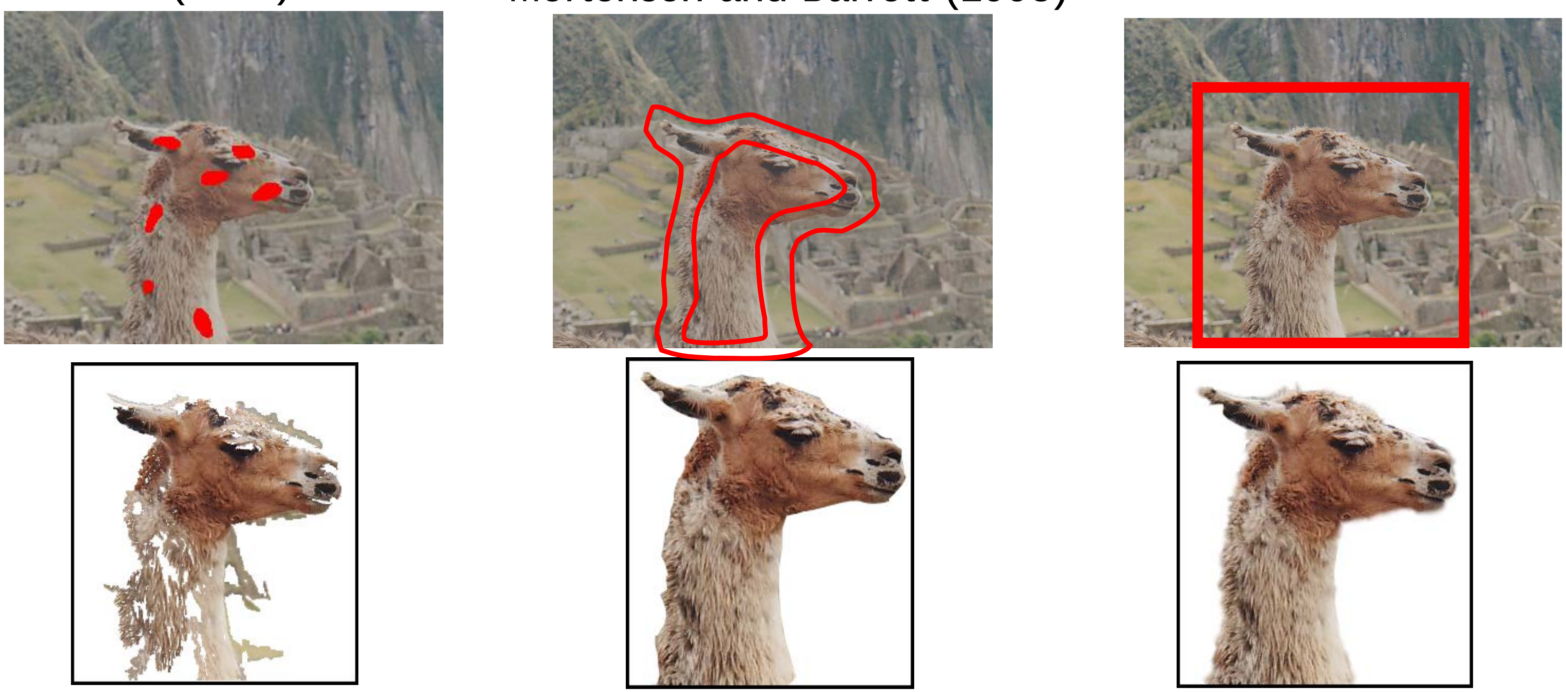
Magic Wand
(198?)

Intelligent Scissors
Mortensen and Barrett (1995)

GrabCut

User Input

Result

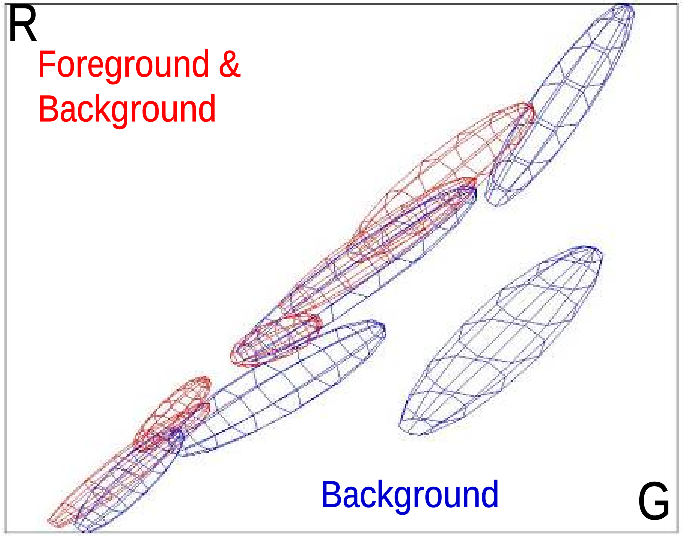


Regions

Boundary

Source: C. Rother

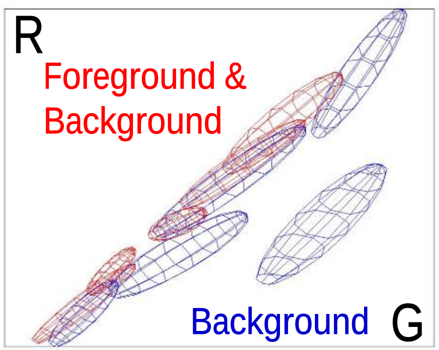
Color model (1)



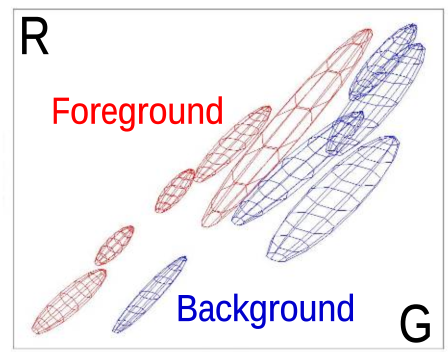
Gaussian Mixture Model (typically 5-8 components)

Source: K. Rother

Color model (2)



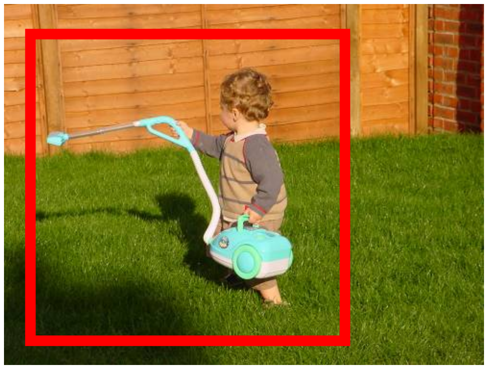
Iterated graph cut



Gaussian Mixture Model (typically 5-8 components)

Source: K. Rother

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

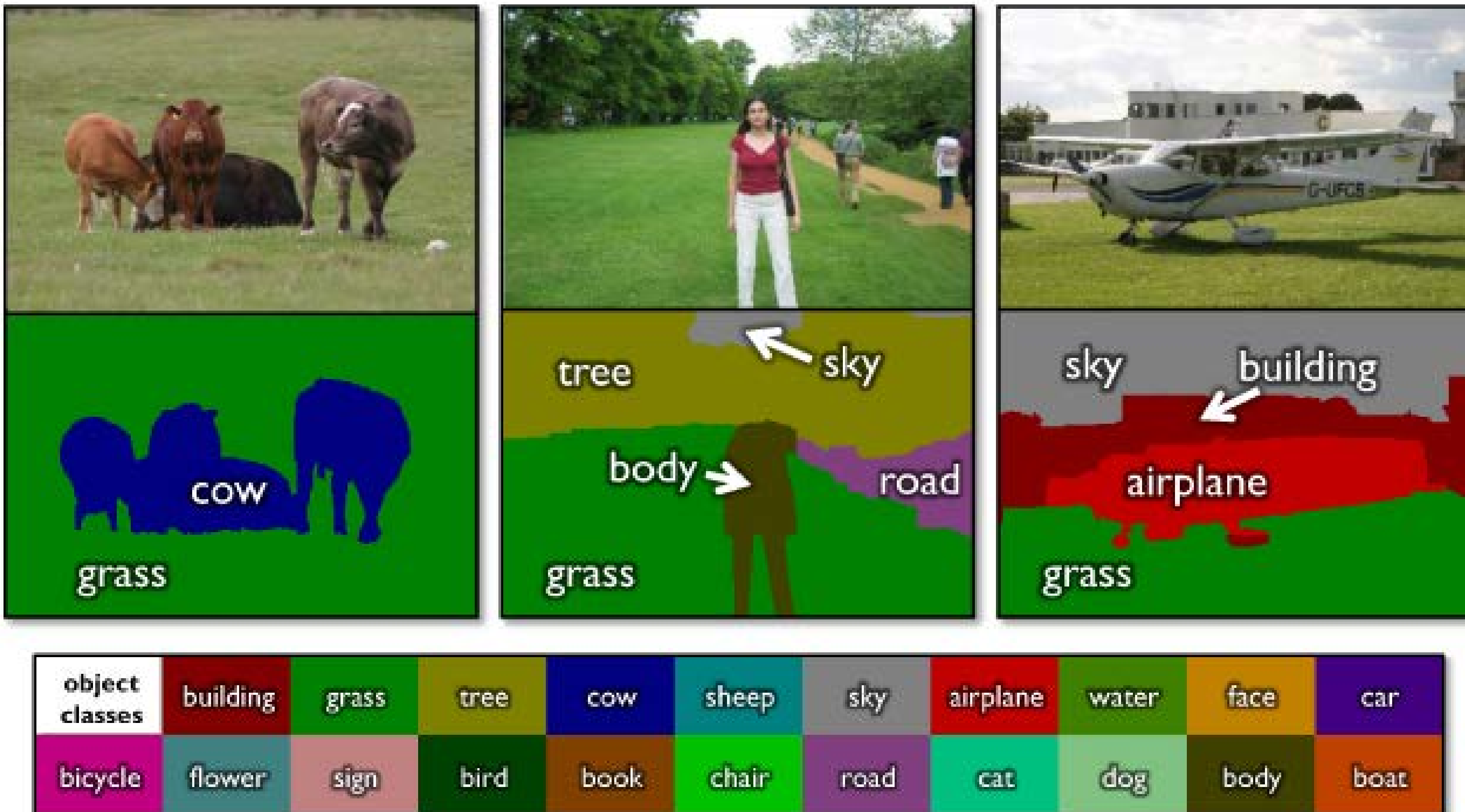


Easier examples

More difficult examples

	Camouflage &	Fine structure	Harder Case
Initial Rectangle			
Initial Result			

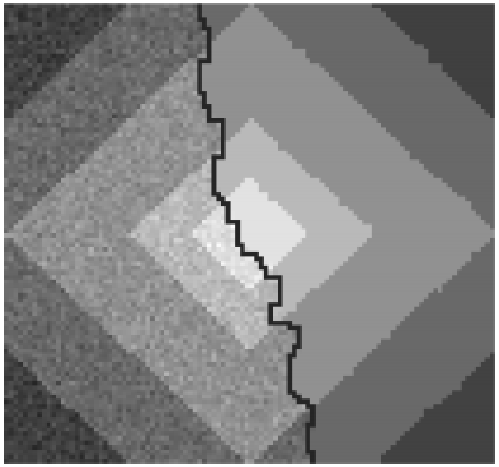
Using graph cuts for recognition



TextonBoost (Shotton et al. 2009 IJCV)

Other applications of minimal graph cuts

- Image restoration



- Stereo disparity