Combinatorial Optimization in Computer Vision (IN2245)

Frank R. Schmidt
Csaba Domokos

Winter Semester 2015/2016

6. Graph Cut

Graph Cut

Graph Cut

A **directed graph** or **digraph** is a triple $(V, \mathcal{E}, c)$ with

- the finite vertex set $V = \{1, \dots, N\}$,
- the finite edge set $\mathcal{E} \subset \{(i,j) \in V \times V | i \neq j\}$ and
- the weight function $c \colon \mathcal{E} \to \mathbb{R}$.

A **cut** is a disjoint partition (S, T) of V, i.e., $V = S + T$, and its **cut value** is

$$\operatorname{Cut}(S,T) = \sum_{e \in \mathcal{E} \cap S \times T} c(e).$$

Given two different vertices $s, t \in V$, (S,T) is called an **s-t cut** if $s \in S$ and $t \in T$. Given such s and t, we call $(V, \mathcal{E}, c, s, t)$ a **network**.

The **cut problem** resp. **s-t cut problem** is to find a cut resp. s-t cut (S,T) that minimizes its cut value $\operatorname{Cut}(S,T)$.

Submodularity of MinCut

Theorem 1. *Let $G = (V, \mathcal{E}, c, s, t)$ be a network, $V_0 := V - \{s,t\}$ and*

$$E \colon \mathcal{P}(V_0) \to \mathbb{R} \qquad A \mapsto \operatorname{Cut}(A + \{s\}, V_0 - A + \{t\}).$$

Then E is submodular iff $c(e) \geqslant 0$ for all $e \in \mathcal{E}$.

Proof. For each $A \subset V_0$, define $x \colon V \to \mathbb{B}$ via $x_i := [i \in A + \{s\}]$. Then

$$E(A) = \overline{E}(x) := \sum_{(i,j) \in \mathcal{E}} c(i,j) \cdot x_i \overline{x}_j.$$

Observing that

$$\frac{\partial^2}{\partial x_i \partial x_j} \overline{E}(x) = \begin{cases} -c(i,j) & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i,j \in V$$

proves the theorem.

Network Flow

Maximum Flow

Ford and Fulkerson proved that there is a relationship between the discrete graph cut problem and the continuous **maximum flow** problem.
Let $(V, \mathcal{E}, c, s, t)$ be a network with positive edge weights. We will refer to $c(e)$ as **capacity** of $e \in \mathcal{E}$. A function $f \colon \mathcal{E} \to \mathbb{R}^+$ is called a **flow** if

$$\begin{aligned} f(e) &\leqslant c(e) && \text{for all } e \in \mathcal{E} \\ \sum_{(i,j) \in \mathcal{E}} f(i,j) - \sum_{(j,i) \in \mathcal{E}} f(j,i) &=: \operatorname{div} f(i) = 0 && \text{for all } i \in V - \{s,t\} \end{aligned}$$

The negative of the **divergence** $\operatorname{div} f$ is sometimes referred to as **excess**, since it measures the amount of flow that enters a node but does not exit it.
The **flow value** of f is

$$\operatorname{Flow}(f) := \operatorname{div} f(s) = -\operatorname{div} f(t)$$

The **maximum flow problem** looks for a flow f that maximizes $\operatorname{Flow}(f)$.

MaxFlow-MinCut Theorem

Theorem 2. *Let $G = (V, \mathcal{E}, c, s, t)$ be a network. Then*

$$\max_{f \text{ is flow}} \operatorname{Flow}(f) = \min_{(S,T) \text{ is } s-t \text{ cut}} \operatorname{Cut}(S,T)$$

We assume that $c(i,j) > 0$. If this is not the case, we can remove the zero-edges from $\mathcal{E}$ without changing the MaxFlow or the MinCut problem.

Given a flow f of G, we define the **residual graph** $G_f := (V, \mathcal{E}_f)$ via

$$\mathcal{E}_f = \{e \in \mathcal{E} | f(e) < c(e)\} \cup \{e^{-1} \in \mathcal{E}^{-1} | f(e) > 0\}$$

where $\mathcal{E}^{-1} = \{(i,j) \in V^2 | (j,i) \in \mathcal{E}\}$ refers to the set of **reversed edges**.

Given $U \subset V$, we define the **divergence** of U as

$$\operatorname{div}_f(U) := \sum_{(u,j) \in \mathcal{E} \cap U \times \overline{U}} f(u,j) - \sum_{(i,u) \in \mathcal{E} \cap \overline{U} \times U} f(i,u) = \sum_{u \in U} \operatorname{div} f(u)$$

Proof. We will first show that for an arbitrary flow f and cut (S,T), we always have $\mathrm{Flow}(f) \leqslant \mathrm{Cut}(S,T)$:

$$\begin{aligned}\mathrm{Flow}(f) &= \mathrm{div}\, f(s) = \mathrm{div}_f(S)\\ &= \sum_{(u,j)\in\mathcal{E}\cap S\times T} f(u,j) - \sum_{(i,u)\in\mathcal{E}\cap T\times S} f(i,u) \leqslant \mathrm{Cut}(S,T)\end{aligned}$$

Let us now assume that f^* is a maximal flow and define $S \subset V$ as the set of all vertices that are path-connected to s in the *residual graph* G_{f^*}.

Case 1 $t \notin S$

Every edge $(i,j) \in \mathcal{E}$ that leaves S is not an edge of $\mathcal{E}_{f^*}$,i.e., $f^*(i,j) = c(i,j)$. The reverse edge of every edge $(i,j) \in \mathcal{E}$ that enters S is also not an edge of $\mathcal{E}_{f^*}$,i.e., $f^*(i,j) = 0$. Therefore,

$$\mathrm{Cut}(S, V-S) = \mathrm{div}_f(S) = \mathrm{Flow}(f)$$

Proof. (Cont.)

Case 2 $t \in S$

There is a path of edges $e_1, \ldots, e_k \in \mathcal{E}_{f^*}$ that connects s with t. We define the positive residuals

$$\epsilon_i := \begin{cases} c(e_i) - f^*(e_i) & \text{if } e_i \in \mathcal{E} \\ f^*(e_i^{-1}) & \text{if } e_i^{-1} \in \mathcal{E}\end{cases}$$

Setting $\epsilon = \min_{i=1,\ldots,k} \epsilon_i > 0$, we can define a new flow

$$\hat{f}(e) := \begin{cases} f^*(e) + \epsilon & \text{if } e = e_k \text{ for some } k \\ f^*(e) - \epsilon & \text{if } e^{-1} = e_k \text{ for some } k \\ f^*(e) & \text{otherwise}\end{cases}$$

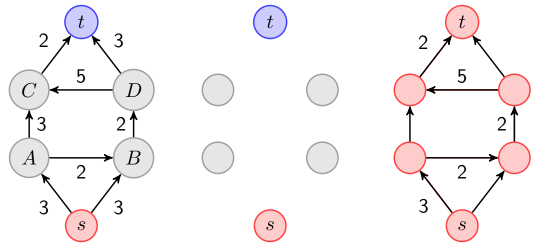
with $\mathrm{Flow}(\hat{f}) = \mathrm{Flow}(f^*) + \epsilon$ which contradicts the optimality of f^*.

From the MaxFlow-MinCut theorem, we obtain the following algorithm to compute the minimal $s-t$ cut (S,T) in the network $G = (V, \mathcal{E}, c, s, t)$.

1. Set $f: \mathcal{E} \to \mathbb{R}_0^+$ as $f(e) = 0$ for all $e \in \mathcal{E}$.
2. If there is no path in G_f that connects s with t, go to Step 5.
3. Let $e_1, \ldots, e_k$ be a path from s to t, ϵ_i the positive residuals as in the proof of the MaxFlow-MinCut theorem and $\epsilon := \min_{1\leqslant i\leqslant k} \epsilon_i$.
4. Replace the flow f with the **augmented flow**

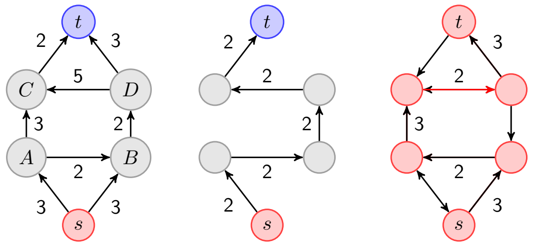
$$\hat{f}(e) := \begin{cases} f^*(e) + \epsilon & \text{if } e = e_k \text{ for some } k \\ f^*(e) - \epsilon & \text{if } e^{-1} = e_k \text{ for some } k \\ f^*(e) & \text{otherwise}\end{cases}$$

   and go to Step 2.
5. Let S be the path-connected component of s in G_f and $T = V - S$.



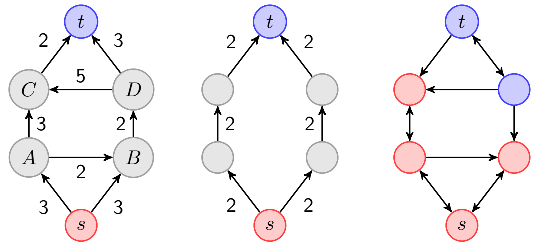
Network G Flow f Residual Graph G_f

Iteration 0: The flow value is 0.



Network G Flow f Residual Graph G_f

Iteration 1: The flow value is 2.



Network G Flow f Residual Graph G_f

Iteration 2: The flow value is 4.

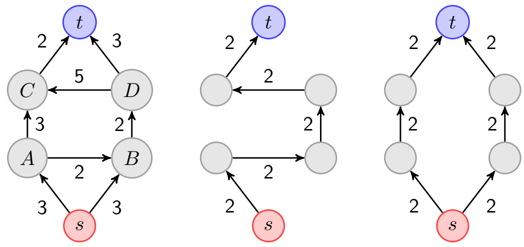
Theorem 3. *For $x_1, \ldots, x_n \in \mathbb{B}$ the following holds*

$$\overline{x}_1 + \left[\sum_{i=1}^{n-1} x_i\overline{x}_{i+1} - x_{i+1}\overline{x}_i\right] + x_n = 1.$$

Proof. Exercise. □

If we augment in a network G a flow along the path $(s, i_1, \ldots, i_n, t)$ from s to t, we can understand this as reducing the capacity along this path and adding capacity along some edges along the reversed path.

This theorem shows that the Ford-Fulkerson algorithm can be understood as a reparametrisation of the energy that describes the graph cut problem. It changes the energy by increasing the constant during each iteration.



Network G Flow f_0 Flow f_1

$$\begin{aligned}E(x) =& 3\overline{x}_A + 3\overline{x}_B + 2x_A\overline{x}_B + 3x_A\overline{x}_C + 2x_B\overline{x}_D + 5x_D\overline{x}_C + 2x_C + 3x_D\\ =& 2 + 1\overline{x}_A + 3\overline{x}_B + 2x_B\overline{x}_A + 3x_A\overline{x}_C + 2x_D\overline{x}_B + 3x_D\overline{x}_C + 2x_C\overline{x}_D + 3x_D\\ =& 4 + 1\overline{x}_A + 1\overline{x}_B + 2x_A\overline{x}_B + 1x_A\overline{x}_C + 2x_C\overline{x}_A + 2x_D\overline{x}_B + 5x_D\overline{x}_C + 1x_D\end{aligned}$$

Properties

Theorem 4. *If the Ford-Fulkerson algorithm terminates, it computes a maximal flow resp. a minimal cut of $G = (V, \mathcal{E}, c, s, t)$.*

Theorem 5. *The Ford-Fulkerson algorithm terminates for $G = (V, \mathcal{E}, c, s, t)$ if $c\colon \mathcal{E} \to \mathbb{N}_0$ or $c\colon \mathcal{E} \to \mathbb{Q}_0^+$.*

Proof. The maximal flow is bounded from above by $K := \operatorname{Cut}(s, V - s)$. If $c\colon \mathcal{E} \to \mathbb{N}^+$, the flow is increased in each augmentations step by at least 1. Thus, the algorithm terminates after no more than K iterations.
If $c\colon \mathcal{E} \to \mathbb{Q}$, the capacity $c(e_i)$ of each edge e_i $(i = 1, \dots, |\mathcal{E}|)$ can be written as $\frac{p_i}{q_i}$. $c(e_i)$ is therefore a multiple of $\epsilon = \prod_{i=1}^{|\mathcal{E}|} \frac{1}{q_i}$ and each augmentation step increases the flow by at least ϵ. Thus, the algorithm terminates after no more than $\frac{K}{\epsilon}$ iterations.

Example of Non-Termination *

There exists an example where the computed flow does not even converge to the maximal flow. To this end, let $g = \frac{\sqrt{5}-1}{2}$ the number of the **golden section**. It satisfies the relationship $1 - g = g^2$.

We define the network $G = (V, \mathcal{E}, c, s, t)$ as follows

$$
\begin{aligned}
V =& \{1, 2, 3, 4, 5, 6, s, t\} \\
\mathcal{E} =& \{s\} \times \{1, \dots, 6\} + \{1, \dots, 6\} \times \{t\} + \{1, \dots, 6\} \times \{1, \dots, 6\} \\
& - \{(2, 1), (4, 3), (6, 5)\} - \{(1, 1), \dots, (6, 6)\}
\end{aligned}
$$

$$
c(e) = \begin{cases} 1 & \text{if } e = (1, 2) \text{ or } e = (3, 4) \\ g & \text{if } e = (5, 6) \\ K & \text{otherwise} \end{cases},
$$

where $K > 1 + g$ is an arbitrary big number.
One can easily verify that the maximal flow of this network is $6K$.

Example of Non-Termination *

Lemma 1. *There is an instance of the Ford-Fulkerson algorithm that if applied to the previously defined G satisfies at each Iteration $n \geqslant 0$:*

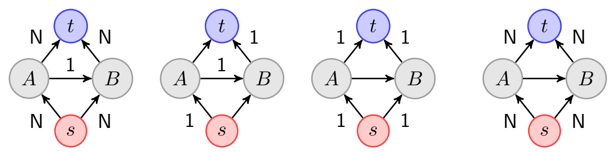
1. *There are edges $e_1 = (u_1, v_1), e_2 = (u_2, v_2), e_3 = (u_3, v_3)$ in G_f of pairwise non-incident vertices with the residual values* $\epsilon(e_1) = g^n, \epsilon(e_2) = g^{n+1}, \epsilon(e_3^{-1}) = 0.$
2. $\operatorname{Flow}(f) = \sum_{i=1}^{n} g^i < 1 + g.$

Proof.

Case 1: $n = 0$
The lemma is satisfied for $e_1 = (1, 2)$, $e_2 = (5, 6)$ and $e_3 = (3, 4)$.

Case 2: $n \to n + 1$
Augmenting the path $s, u_1, v_1, u_2, v_2, u_3, v_3, t$ proves the lemma for $n + 1$ and the edges $e_1 = (v_3, u_3)$, $e_2 = (u_1, v_1)$, $e_3 = (v_2, u_2)$.

Example of Pseudo-Polynomiality



Network G — Flow f Iteration: 1 — Flow f Iteration: 2 — Flow f Iteration: $2N$

A graph can take $2N$ iterations where N is the highest capacity in the graph. Thus, Ford-Fulkerson has a pseudo-polynomial running time for $c\colon \mathcal{E} \to \mathbb{N}_0$.

It was shown by Dinitz (1970) and independently by Edmonds and Karp (1972) that the maximal flow method becomes **polynomial** if one always uses the shortest path (amount of edges) from s to t in the augmentation step.

Graph Representable Energies

Submodular Energies with 2-Cliques

We saw that every graph cut problem can be represented as a submodular energy that uses cliques of size 2 or smaller. The opposite is also true:

Theorem 6. *If $E(x) = C + \sum_{i \in \Omega} C_i x_i + \sum_{i,j \in \Omega} C_{ij} x_i x_j$ is submodular, the minimization of E can be cast as a graph cut problem.*

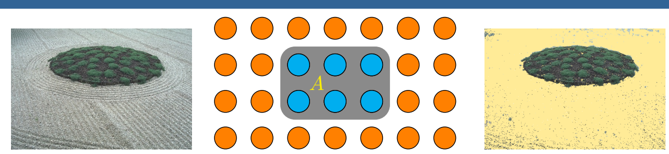
Proof. Taking the second derivatives of E, we see that $C_{ij} \leqslant 0$ has to be satisied for all $i, j \in \Omega$. For these negative C_{ij} we have

$$
C_{ij} x_i x_j = C_{ij} x_i (1 - \overline{x}_j) = C_{ij} x_i - C_{ij} x_i \overline{x}_j = C_{ij} - C_{ij} 1 \overline{x}_i - C_{ij} x_i \overline{x}_j.
$$

In addition, we have for $C_i < 0$ that $C_i x_i = C_i - C_i 1 \overline{x_i}$.
For $C_i > 0$, we obtain $C_i x_i = C_i x_i \overline{0}$.

Image Segmentation

Data Driven Image Segmentation



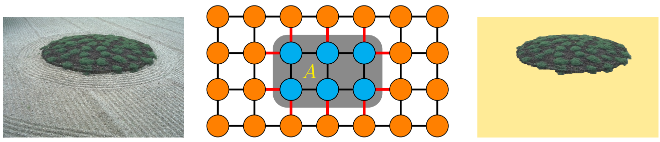
Given Image — Data Term — Global Minimum

Every segment $A \subset \Omega$ induces a binary variable $x \in \mathbb{B}^N$ with $N := |\Omega|$

$$
\begin{aligned}
\operatorname*{argmin}_{x \in \mathbb{B}^N} E(x) &= \operatorname*{argmin}_{x \in \mathbb{B}^N} \sum_{i=1}^{N} \left[f_i^{(1)} x_i + f_i^{(0)} \overline{x}_i \right] \\
&= \operatorname*{argmin}_{x \in \mathbb{B}^N} \sum_{i=1}^{N} f_i^{(0)} + \sum_{i=1}^{N} [\underbrace{f_i^{(1)} - f_i^{(0)}}_{=: f_i}] x_i = \operatorname*{argmin}_{x \in \mathbb{B}^N} \sum_{i=1}^{N} f_i x_i
\end{aligned}
$$

This energy can be minimized in linear time.

Length Regularization



Given Image Data + Length Global Minimum

To every pixel i, we define $\mathcal{N}(i)$ as the set of neighboring pixels of i.

$$\underset{x \in \mathbb{B}^N}{\operatorname{argmin}} E(x) = \underset{x \in \mathbb{B}^N}{\operatorname{argmin}} \sum_{i=1}^{N} f_i x_i + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} c(i,j) x_i \overline{x}_j$$

This energy is submodular and only uses quadratic expressions.
Thus, it can be efficiently optimized via graph cut.

Length Approximation

The quadratic expression $Q(x) = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} c(i,j) x_i \overline{x}_j$ is sometimes referred to as the length of the boundary ∂A. What is actually counts is the amount intersections that straight lines have with ∂A.

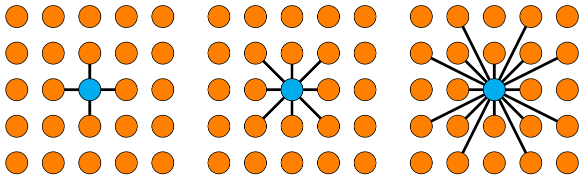
Every line $L \subset \mathbb{R}^2$ in a plane can be parametrized by its normal $\nu = (\cos(\theta), \sin(\theta))$ and the scalar product $\langle p, \nu \rangle = \alpha$ for all $p \in L$. We will write $L(\nu, \alpha)$ for such a line.

Therefore, Q is a discrete approximation of

$$\ell(A) := \int_0^{\pi} \int_{-\infty}^{\infty} \#(L(\nu,\alpha) \cap \partial A) d\alpha d\nu$$

According to the **Cauchy-Crofton formula** we have $\ell(A) = 2 \operatorname{length}(\partial A)$. This justifies calling Q the **length term** of our energy.

Improving Length Approximation



4-neighborhood 8-neighborhood 16-neighborhood

To get a good approximation of the length, we have to weigh each edge (u, v) with the angle distance to its neigboring edges that are also incident to u.

In the case of the **4-neighborhood** and the **8-neighborhood** all edges have to be weighted by $\frac{\pi}{4}$ resp. $\frac{\pi}{8}$.

For the **16-neighborhood** different edges have to be weighted differently.

Interactive Graph Cuts

Boykov and Jolly were the first to show that graph cut based method can be used for interactive image segmentation.

To this end, let $I \colon \Omega \to \mathbb{R}^d$ be an image into a d-dimensional color space.

1. The user provides disjoint subsets $\mathcal{O}, \mathcal{B} \subset \Omega$ that are marked as object ($\mathcal{O}$) and background ($\mathcal{B}$). These subsets are called **seeds**.
2. With respect to the seeds probability distributions p for foreground and q for background are computed.
3. The data term of a pixel i is set to $f_i := \log\left(\frac{q(I(i))}{p(I(i))}\right)$. The length term between two pixels is set to $c(i,j) = \lambda \exp\left(-\frac{[I(i)-I(j)]^2}{2\sigma^2}\right)$.
4. Minimize the energy $E(x) = \sum_{i=1}^{N} f_i x_i + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} c(i,j) x_i \overline{x}_j$ and obtain a cut (S, T).
5. Update p and q with respect to S and T. If p or q changes go to Step 3.
6. S provides for an image segmentation.

Example (Interactive Graph Cut)

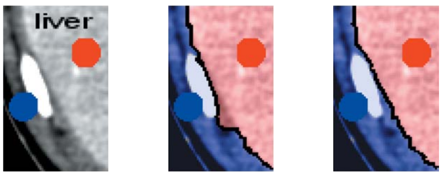


Image with Seeds Undirected Edges Directed Edges

Instead of undirected edges between two neighboring vertices, one can also define the capacity depending on the edge. Boykov and Jolly proposed

$$c(p,q) = \begin{cases} 1 & \text{if } I(p) < I(q) \\ \exp\left(-\frac{[I(p)-I(q)]^2}{2\sigma^2}\right) & \text{otherwise} \end{cases}$$

to favor segmentation edges between bright regions ($\mathcal{O}$) and dark regions ($\mathcal{B}$).

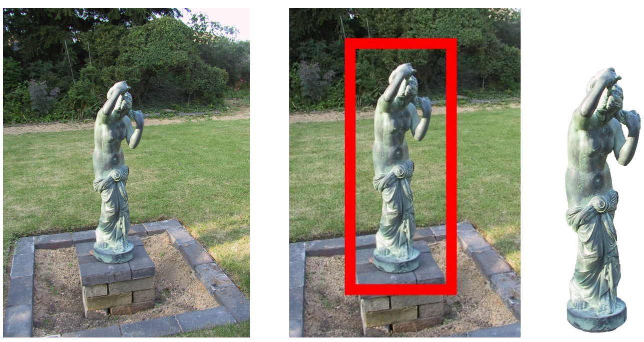
GrabCut Protocol

The interactive Graph Cut was patented by Siemens. A slightly different protocol, called **GrabCut**, was patented by Microsoft:

To this end, let $I \colon \Omega \to \mathbb{R}^d$ be an image into a d-dimensional color space.

1. The user provides a bounding box around the object.
2. With respect to this bounding box, probability models p for foreground and q for background are estimated (using Gaussian mixture models).
3. The data term of a pixel i is set to $f_i := \log\left(\frac{q(I(i))}{p(I(i))}\right)$. The length term between two pixels is set to $c(i,j) = \lambda \exp\left(-\frac{[I(i)-I(j)]^2}{2\sigma^2}\right)$.
4. Minimize the energy $E(x) = \sum_{i=1}^{N} f_i x_i + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} c(i,j) x_i \overline{x}_j$ and obtain a cut (S, T).
5. Update p and q with respect to S and T. If p or q changes go to Step 3.
6. S provides for an image segmentation.

Example (GrabCut)



Original Image User Interaction Segmentation

Literature *

Graph Cut

- Ford and Fulkerson, *Maximal Flow through a Network*, 1956, Canadian J. Math.(8), 399–404.
- Dinitz, *Algorithm for solution of a problem of maximum flow in a network with power estimation*, 1970, Dokl. Akad. nauk SSSR (11), 1277–1280.
- Schrijver, *Combinatorial Optimization*, Chapter 10.

Computer Vision

- Kolmogorov and Zabih, *What Energy Functions can be Minimized via Graph Cuts?*, 2002, ECCV, 65–81.
- Boykov and Jolly, *Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images*, 2001, ICCV, 105–112.
- Rother, Kolmogorov, Blake, *GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts*, 2004, SIGGRAPH, 309–314.