Image Segmentation via Graph-Cuts

Pavel Matula

21. května 2012

Outline

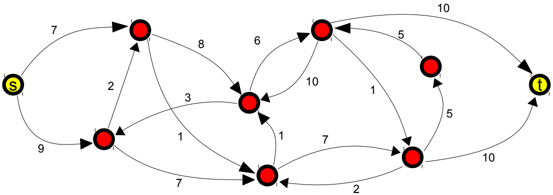
Contents

Contents

Flow Network

Definition

A directed graph $\mathcal{G} = (V, E)$ where V is the set of graph nodes and $E \subseteq V \times V$ is the set of graph edges. Each edge $(u, v) \in E$ has a real-valued capacity $c_{uv} \geq 0$. Further, there are two distinguished nodes in V, called terminal nodes: the source, denoted s and the sink, denoted t.

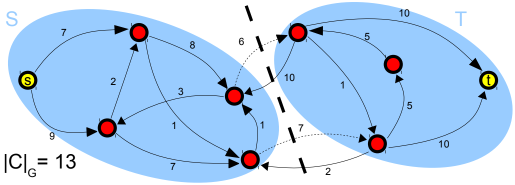


Cut

Definition

An *st*-cut $\mathcal{C}$ is a partition of the set V into two disjoint subsets S and $T = V \setminus S$ such that $s \in S$ and $t \in T$. Cut capacity $|\mathcal{C}|_{\mathcal{G}}$ is the sum of the capacities of the edges going from S to T:

$$|\mathcal{C}|_{\mathcal{G}} = \sum_{(u,v)\in E, u\in S, v\in T} c_{uv}$$



Minimum and Maximum Cuts

- There are $2^{|V|-2}$ possible *st*-cuts
- Minimum cut is a cut with the smallest possible capacity
  - Problem of finding a minimum cut has polynomial time complexity
  - We will discuss algorithms later
- Maximum cut is a cut with the largest possible capacity
  - Problem of finding a maximum cut is NP-hard
- There may be several minimum and maximum cuts

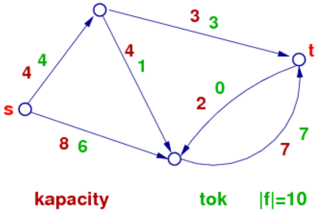
Flow

- Flow in network $G = (V, E)$ is a mapping $f : E \to \mathbb{R}_0^+$ satisfying
  - Capacity constraint: $0 \le f_{uv} \le c_{uv}$ for all $u, v \in V$
  - Continuity condition: for all $u \in V \setminus \{s, t\}$ holds:

$$\sum_{v \in V} (f_{uv} - f_{vu}) = 0.$$

  - The value of the flow outgoing from source s is

$$|f| = \sum_{v \in V} f_{sv}.$$



Maximum Flow and Minimal Cut Duality

- Maximal flow is a flow with the maximal value.
- Ford-Fulkerson:
  - problem of finding minimal cut is equivalent to problem of finding maximal flow.
  - capacity of minimal cut is equal to the value of maximal flow.
  - edges belonging to minimal cut are those which capacity is fully saturated by the flow.

Contents

Ford-Fulkerson

- Obecný iterativní framework pro hledání maximálního toku.
- Sestrojíme reziduální síť.
- Najdeme zlepšující cestu v reziduální síti z uzlu s do uzlu t.
- Zvýšíme tok po dané cestě o hodnotu hrany s nejmenší kapacitou na této cestě.
- Výpočet opakujeme dokud je možné najít další zlepšující cestu.
- V obecném případě složitost $O(E|f|)$, kde E je počet hran a $|f|$ hodnota maximálního toku.
- Nemusí konvergovat pro sítě s neceločíselnými kapacitami.

Edmonds-Karp

- Zvláštní případ metody Ford-Fulkerson.
- Dvě varianty:
  - Zlepšování toku vždy po nejkratší cestě. Složitost $O(VE^2)$.
  - Zlepšování toku vždy po nejširší cestě. Složitost $O(E\log(EU))$, kde U je nejvyšší kapacita vyskytující se v síti.
- Záruka konvergence i pro sítě s iracionálními kapacitami hran.

Dinitz

- Zlepšování toku po všech nejkratších cestách naráz.
- Pomocí prohledávání do šířky jsou nalezeny všechny nejkratší cesty v reziduální síti.
- Obdržíme podsíť nejkratších cest, která neobsahuje smyčky.
- V získané podsíti najdeme maximální tok pomocí prohledávání do hloubky.
- Lze ukázat, že v další iteraci bude délka nejkratších cest ostře větší.
- Složitost $O(V^2E)$. V praxi výrazně rychlejší, než Edmonds-Karp.

Push-Relabel

- V současnosti pravděpodobně nejrychlejší a nejvyužívanější metody pro obecné grafy.
- Pro každý uzel si udržujeme předpokládanou délku nejkratší cesty k uzlu t.
- Uzly mohou obsahovat přebytky toku (aktivní uzly).
- Optimistické prostrkávání toku do uzlů u kterých předpokládáme, že jsou blíže uzlu t.
- Rozhodování vždy pouze na základě lokálních informací. Vhodné k paralelizaci a lokalizaci výpočtu v paměti.
- Pro rychlý výpočet ovšem třeba pravidelně provádět globální update informací.
- Složitost záleží na strategii volby aktivního uzlu:
  - Naivní implementace $O(V^2E)$
  - First-in First-out strategie $O(V^3)$
  - Nejvzdálenější aktivní uzel $O(V^2\sqrt{E})$

Contents

What energy functions can be minimized via graph cuts?

- [Kolmogorov, Zabih] Let E be a function of n binary variables of type

$$E(x_1, \ldots, x_n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x_i, x_j).$$

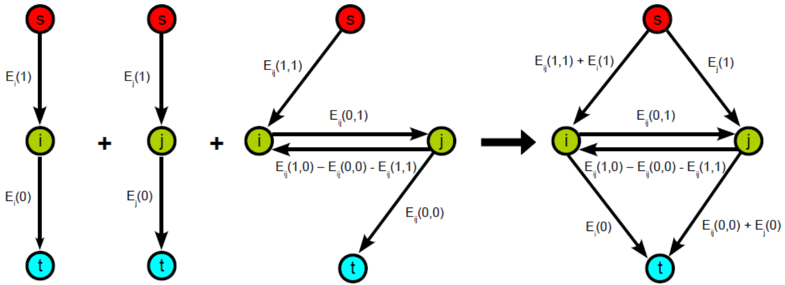
  Then, E is graph-representable if and only if each term $E^{i,j}$ satisfies the inequality

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0).$$

- Necessary and sufficient condition to be able to compute the exact global minimum of E using a single graph cut.

Graph Construction

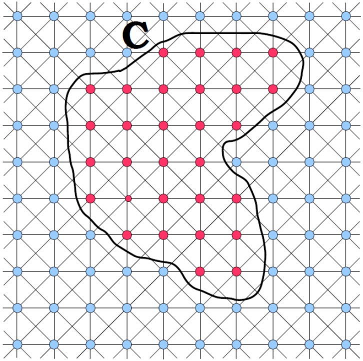
- A graph $G = (V, E)$ is built where V contains the two terminal nodes s and t representing the labels 0 and 1, respectively, and a node for each variable x_i.



Contents

Cut metrics

- Given a regular orthogonal grid and neighbourhood $\mathcal{N}$
- $|\mathcal{C}|_{\mathcal{G}}$ - Cut metric. Sum of weights of edges connecting inner and outer nodes
- For any contour - $|\mathcal{C}|_{\mathcal{G}} \approx |\mathcal{C}|_{\varepsilon}$ and $|\mathcal{C}|_{\mathcal{G}} \rightarrow |\mathcal{C}|_{\varepsilon}$ with increasing grid resolution and neighbourhood density



The Goal

How to set edge weights w_k?

Cauchy-Crofton Formula

- The Cauchy-Crofton formula establishes a connection between Euclidean length $L(\mathcal{C})$ of a curve $\mathcal{C}$ in $\mathbb{R}^2$ and a measure of a set of lines intersecting it:

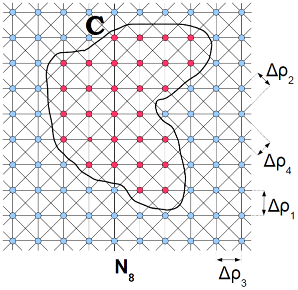
$$L(\mathcal{C}) = \frac{1}{2} \int n_c d\mathcal{L}$$

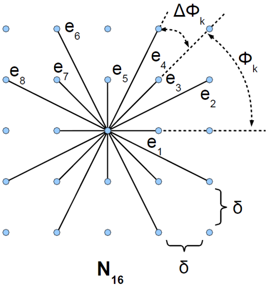
  where n_c is the number of intersections n_c with lines $\mathcal{L}$.

- Consider the set of all lines $\mathcal{L}$ given by polar formula $\mathcal{L}(\phi, \rho)$. Then the Cauchy-Crofton formula can be written as

$$L(\mathcal{C}) = \frac{1}{2} \int_{-\infty}^{\infty} \int_{0}^{\pi} n_c(\phi, \rho) d\phi d\rho.$$

Discretization and Boykov's Approach

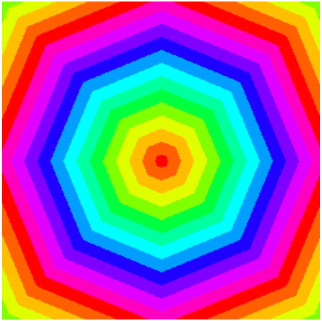




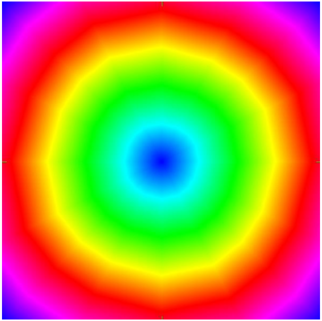
$$|\mathcal{C}|_\varepsilon = \int_0^\pi \int_{-\infty}^{+\infty} \frac{n_c(\phi,\rho)}{2}\, d\rho\, d\phi \approx \sum_{k=1}^{n} \left(\sum_i \frac{n_c(k,i)}{2} \Delta\rho_k \right) \Delta\phi_k \approx \sum_{k=1}^{n} n_c(k) \frac{\Delta\rho_k \Delta\phi_k}{2}$$

Weights (2D isotropic case): $w_k = \dfrac{\Delta\rho_k \Delta\phi_k}{2} \quad \Delta\rho_k = \dfrac{\delta^2}{|e_k|}$

Distance maps



N8



N16

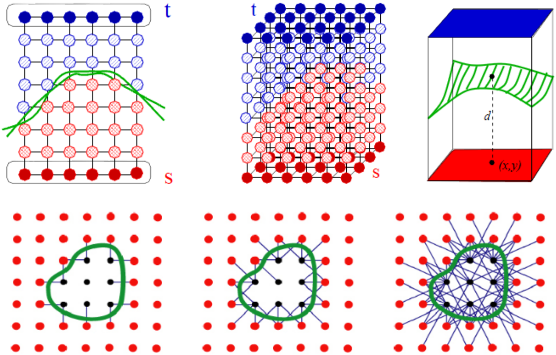
Extension to 3D

Cauchy-Crofton

$$|\mathcal{C}^2|_\varepsilon = \frac{1}{\pi} \int n_c \, d\mathcal{L}$$

Using the same derivative steps

$$w_k = \frac{\Delta\rho_k \Delta\phi_k}{\pi} \qquad \Delta\rho_k = \frac{\delta_x \delta_y \delta_z}{|e_k|}$$

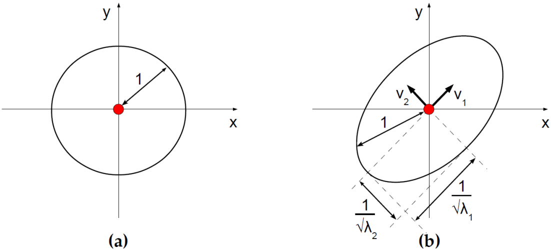


Contents

Riemannian Spaces

- In Riemannian geometry, each point of the space is associated with a metric tensor M that controls how inner product of two vectors is calculated ("space stretching").
- In N-dimensional space, the tensor is a symmetric positive definite N-by-N matrix (a bilinear form) that varies smoothly over the space.
- In case M is constant, the Riemannian norm of a vector u is calculated as:

$$|u|_{\mathcal{R}} = \sqrt{u^T \cdot M \cdot u}.$$



Edge Weights

- Weights approximating a Riemannian metric (Boykov):

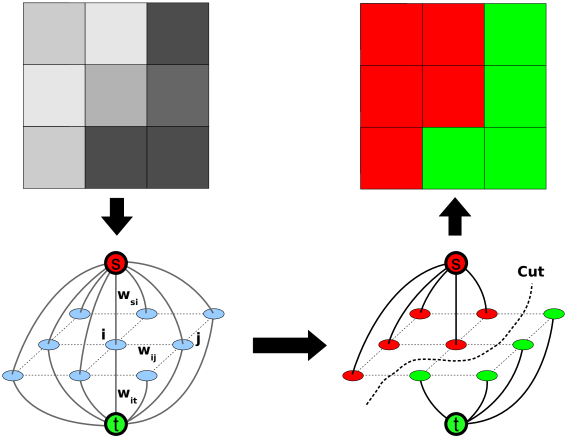
$$w_k^{\mathcal{R}} = w_k^{\mathcal{E}} \cdot \frac{\det M}{(u_k^T \cdot M \cdot u_k)^p}$$

  where u_k is a unit vector in the direction of e_k, $w_k^{\mathcal{E}}$ is the weight for the Euclidean metric approximation and p equals to $3/2$ and 2 in 2D and 3D, respectively.
- Euclidean metric a special case where $M_{\text{const}} = I$

Contents

Contents

Graph Cut Segmentation Framework



Contents

How to set the weights?

- Geodesic Active Contours energy:

$$E_{GAC}(\mathcal{C}) = \int_0^1 g(|\nabla G_\sigma * I(C(q))|)|C'(q)|dq$$

- t-links: 0 or large value (hard constraints)
- n-links - Riemannian tensors

Contents

How to set the weights?

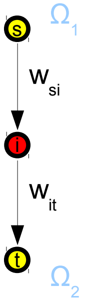
Chan-Vese Functional

$$E_{CV}(\mathcal{C}, c_1, c_2) = \mu L(\mathcal{C}) + \lambda_1 \int_{\Omega_1} (f(x) - c_1)^2 dx + \lambda_2 \int_{\Omega_2} (f(x) - c_2)^2 dx$$

T-link weights:

$$w_{si} = \lambda_2 (f(i) - c_2)^2$$

$$w_{it} = \lambda_1 (f(i) - c_1)^2$$



N-link weights: $w_{ij} = \mu w_k$ - see Page 21

Graph Cut Based Chan-Vese Minimization Overview

Key observation

It is possible to setup w_{ij}, w_{si} and w_{it} such that capacity of any cut approximates the CV energy of the corresponding segmentation for fixed c_1 and c_2.

Alternating minimization scheme:

1. Obtain an initial estimate of c_1 and c_2
2. Construct graph and find globally minimal segmentation with respect to the fixed mean values
3. Update c_1 and c_2
4. Repeat from 2 until reaching a steady state

How to initialize c_1 nad c_2?

Algorithm idea

Minimize the Chan-Vese functional with a relaxed regularization term:

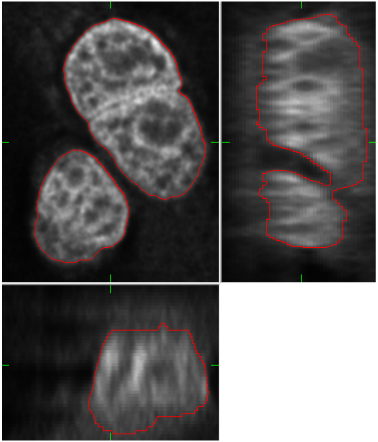
$$E(\mathcal{C}, c_1, c_2) = \lambda_1 \int_{\Omega_1} (f(x) - c_1)^2 dx + \lambda_2 \int_{\Omega_2} (f(x) - c_2)^2 dx$$

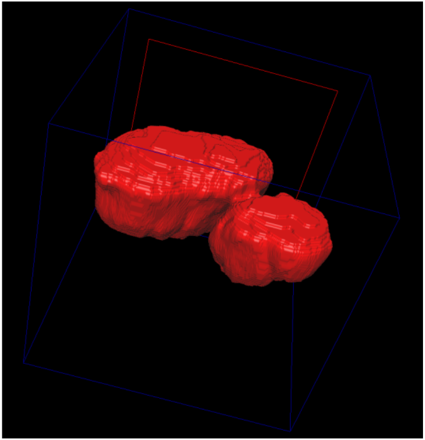
- A significantly simpler problem
- Weighted KMeans clustering
  - Only data terms are compared in each pixel
    - Corresponds to finding a minimum cut with zero N-link weights

Properties

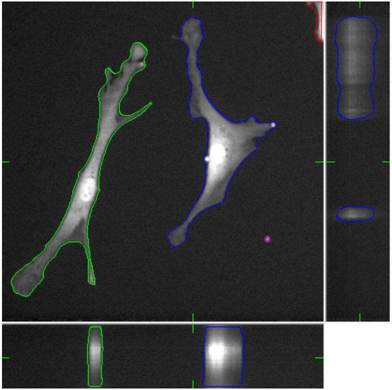
- Advantages:
  - Simple and fast (and “automatic”)
  - Reflects λ_1 and λ_2
  - Very good estimate for small μ
    - Requires less iterations of the main algorithm
- Disadvantages:
  - Only an approximation, initialization dependent
  - Does not guarantee reaching a global minimum

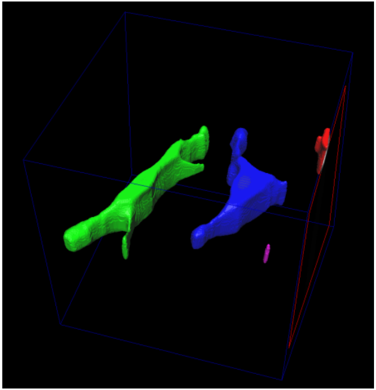
Chan-Vese Segmentation - Examples (1)





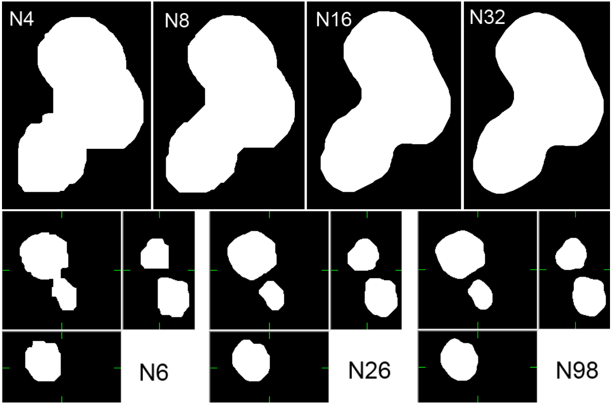
Chan-Vese Segmentation - Examples (2)





Boundary Smoothness

- Depends strongly on the neighbourhood size:



- Large neighbourhoods are computationally expensive

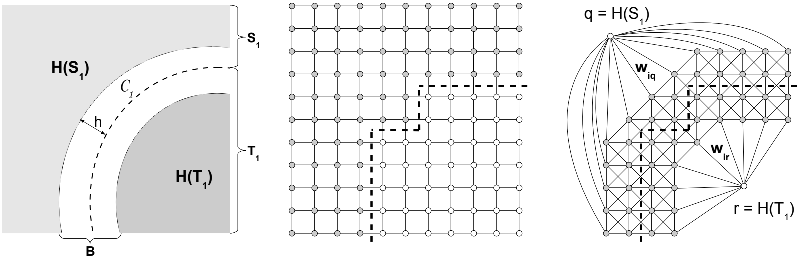
Two-Stage Algorithm

Key observation

Segmentations are different but very close to each other.

Two-stage algorithm:

1. Coarse segmentation using a small neighbourhood
2. Refinement of the segmentation in a narrow band around the boundary using a large neighbourhood



Contents

Conclusion

- Graph-cut framework is powerful tool for discrete function minimization.
- Finding of minimal cut is a polynomial problem and we obtain global optimum.
- GC allows interactive segmenation.
- Geodesic segmentation can easily be implemented using graph-cuts.
- We have presented an iterative algorithm for Chan-Vese minimization.
- The neighbourhood system of the grid is related to boundary smoothness. Two stage-minimization reduces memory demands.
- http://cbia.fi.muni.cz/projects/graph-cut-library.html