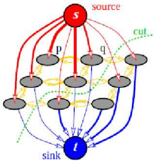
Graph Cuts for Image Segmentation

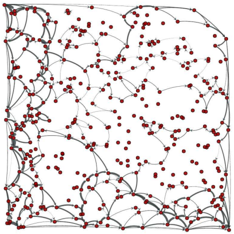
Meghshyam G. Prasad

CSE Department
IIT Bombay
Mumbai.

November 30, 2012

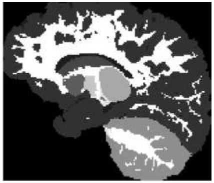
Outline





[1][3]

Image Segmentation



[6]

Goal of Image Segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture.

Applications :- Medical Imaging (Tumor Detection), Face Recognition, Machine Vision etc.

Previous Approaches

- Region Based Segmentation
  - Image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels.
  - Adjacent regions are then merged under some criterion based on homogeneity.
  - Over-stringent criteria create fragmentation; lenient ones overlook blurred boundaries and over-merge.
- Watershed based Image Segmentation
  - Visualizes images in 3-dimensions: x, y, and intensity.
  - Object is distinguished from the background by its up-lifted edges.
  - Give segments with continuous boundaries, also give rise to over-segmentation.

Previous Approaches

- Region Based Segmentation
  - Image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels.
  - Adjacent regions are then merged under some criterion based on homogeneity.
  - Over-stringent criteria create fragmentation; lenient ones overlook blurred boundaries and over-merge.
- Watershed based Image Segmentation
  - Visualizes images in 3-dimensions: x, y, and intensity.
  - Object is distinguished from the background by its up-lifted edges.
  - Give segments with continuous boundaries, also give rise to over-segmentation.

Previous Approaches

- Minimum spanning tree based segmentation [5]
  - Segmentation method based on Kruskal's MST algorithm.
  - Images are modeled as Graph, pixels being nodes.
  - Neighborhood pixels are connected by edges with weight equivalent to similarity between nodes.
  - Edges are considered in increasing order of weight.
  - Edges' endpoint pixels are merged into a region if the pixels are 'similar' to the existing regions' pixels.

Though these approaches gives good accuracy, contextual properties of image grid-graphs are not exploited fully. Such properties can be used by modelling image grid-graph as Markov Random Field (MRF).

Image Segmentation as Energy Minimization

Image Segmentation problem can be modeled as Energy minimization of MRF with Grid Graph containing image pixels. In this framework, one seeks the labeling f that minimizes the energy,

$$E(f) = E_{smooth}(f) + E_{data}(f)$$

$E_{smooth}(f)$:- measures the extent to which f is not piecewise smooth
$E_{data}(f)$:- measures the disagreement between f and the observed data.

$$E_{data}(f) = \sum_{p \in P} D_p(f_p)$$

where D_p measures how well label f_p fits pixel p.
Mostly used model for E_{smooth} is Potts model given as,

$$E_{smooth}(f) = \sum_{\{p,q\} \in N} u_{\{p,q\}}.T(f_p \neq f_q)$$

where T is indicator function i.e. it will output 1 if the input condition is true.

New Moves

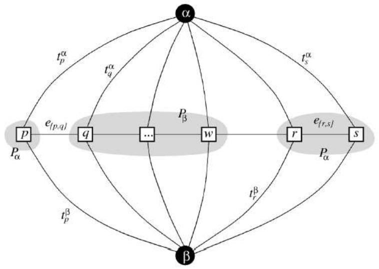
Any labeling f can be uniquely represented by a partition of image pixels $\mathbf{P} = \{P_l | l \in L\}$ where $P_l = \{p \in P | f_p = l\}$ is a subset of pixels assigned label l.

- Standard moves allow only single pixel changing its label at a time, so time consuming.
- So, Boykov et.al[4] proposed new moves which allows many pixels to change their labels at a time:
  - α-β swap
    - Move which changes partition $\mathbf{P}$ to $\mathbf{P}'$ such that some pixels that were labeled α in $\mathbf{P}$ are now labeled as β in $\mathbf{P}'$, and vice-e-versa.
    - Pixels whose labels are other than α,β retain their labels.
  - α-expansion
    - α-expansion move allows any set of image pixels to change their labels to α.
    - But, pixels labeled α retain their labels.

Swap Algorithm

- Start with initial labeling either taken from seeds provided by user or computed from some algorithm.
- For each pair of labels find a labeling which is one α-β swap away from current labeling in a such a way that it will have minimum energy among all labelings which are one α-β swap away from current labeling.
- If this value (of minimum energy) is less than earlier minimum (computed for earlier pairs of labels) we set $success$ flag to 1.
- $success$ Flag indicates that during iteration over pairs of labels we have found another minimum labeling over earlier minimum labeling, so we need to iterate again over all pairs of labels.

Optimal Swap Move

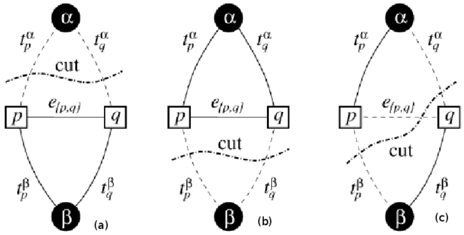


Structure of graph for swap move [4]

edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{q \in N_p, q \notin P_{\alpha\beta}} V(\alpha, f_q)$	$p \in P_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{q \in N_p, q \notin P_{\alpha\beta}} V(\beta, f_q)$	$p \in P_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}, p, q \in P_{\alpha\beta}$

$$f_p^C = \begin{cases} \alpha & \text{if } t_p^\alpha \in \mathcal{C} \text{ for } p \in P_{\alpha\beta} \\ \beta & \text{if } t_p^\beta \in \mathcal{C} \text{ for } p \in P_{\alpha\beta} \\ f_p & p \notin P_{\alpha\beta} \end{cases}$$

Properties of Swap Move



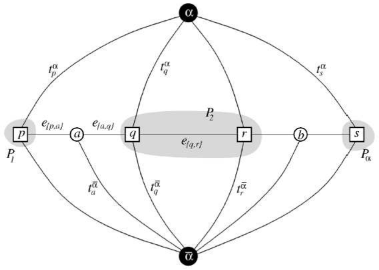
Properties for Swap move [4]

- In (a) and (b) pixels p and q are connected to same terminal, so there is no need to break n-link between those pixels.
- But, in (c), if we don't sever n-link between p and q, there will be path between terminals.

Expansion Algorithm

- Similar to α-β Swap algorithm.
- In each iteration (for each label) we try to find labeling which is one α-expansion from current labeling.
- Some pixels whose label is other than α (the label on which we are iterating over) may change their labels to α.
- But pixels whose label is α do not change their label.
- α-Expansion can be thought of special case of α-β Swap, where β is collection of all labels other than α and movement from only β to α is allowed.

Optimal Expansion Move

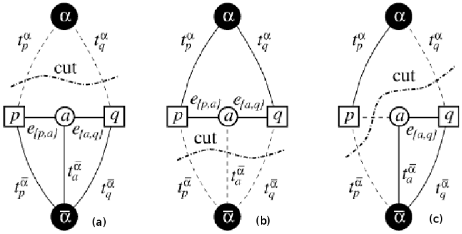


Structure of graph for expansion move [4]

edge	weight	for
$t_p^{\bar\alpha}$	∞	$p \in P_\alpha$
$t_p^{\bar\alpha}$	$D_p(f_p)$	$p \notin P_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in P$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	
$t_a^{\bar\alpha}$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p = f_q$

$$f_p^C = \begin{cases} \alpha & \text{if } t_p^\alpha \in \mathcal{C} \\ f_p & \text{if } t_p^{\bar\alpha} \in \mathcal{C} \end{cases} \quad \forall p \in P$$

Properties of Expansion Move

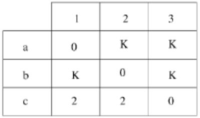


Properties for expansion move [4]

- (a) - If $t_p^\alpha, t_q^\alpha \in \mathcal{C}$, then we need not cut any of the edges from $\epsilon_{\{p,q\}}$.
- (b) - If $t_p^{\bar\alpha}, t_q^{\bar\alpha} \in \mathcal{C}$, we need to cut $t_a^{\bar\alpha}$.
- (c) - If $t_p^{\bar\alpha}, t_q^\alpha \in \mathcal{C}$, we need to cut $e_{\{p,a\}}$ to separate terminals.

Example

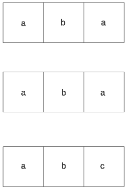
- Assume, an image consisting of just three pixels 1,2 and 3 arranged in one row.
- Three possible labels a, b, c.
- Data Term:- The values for D_p for each pixel/label combination are shown in Figure.
- Coherence Term :- Let $V(a,b) = V(b,c) = K/2$ and $V(a,c) = K$.
- Initially label all pixels as a.



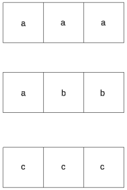
	1	2	3
a	0	K	K
b	K	0	K
c	2	2	0

D_p values [4]

Example



Labeling after Swap moves

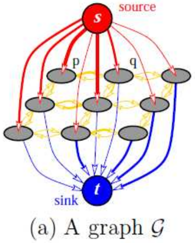


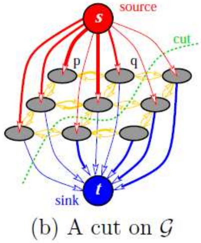
Labeling after Expansion moves

Swap Moves:- We will process pairs of labels in order (a,b), (b,c) and then (a,c) for swap moves. Minimum energy for last configuration is K.

Expansion Moves :- We will process labels in order a, b, c for expansion moves. Minimum energy for last configuration is 4.

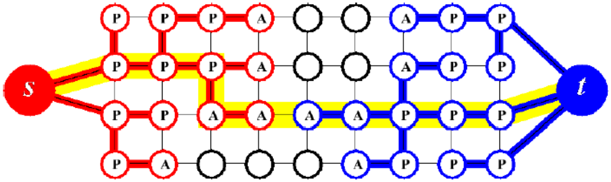
Finding Min-cut in Flow Graphs





Example of directed capacitated graph [3]

Building Search Trees



Example of search trees S(red nodes) and T(blue nodes) [3]

Boykov-Kolmogorov Algorithm I

- “growth” stage: search trees S and T grow until they touch giving an $s \rightarrow t$ path,
  - Active nodes explore adjacent non-saturated edges and acquire new children from set of free nodes.
  - Newly acquired nodes will become active members of corresponding search trees.
  - The growth stage terminates if an active node encounters a neighboring node that belongs to the opposite tree.
- “augmentation” stage: the found path is augmented, search tree(s) break into forest(s),
  - As we are pushing maximum possible flow through augmenting path, some edge(s) will become saturated.
  - When we remove such edge(s), some of the nodes in the trees S and T may become orphans.
  - It effects split of the search trees S and T into forests.

Boykov-Kolmogorov Algorithm II

- The source s and the sink t are still roots of two of the trees while orphans form roots of all other trees.

- “adoption” stage: trees S and T are restored.
  - Most important step of the algorithm.
  - Tries to find valid parent for orphans created in **“augmentation”** stage.
  - A new parent should belong to the same set, S or T, as the orphan.
  - A parent should also be connected through a non-saturated edge.
  - If there is no qualifying parent we remove the orphan from S or T and make it a free node.
- After the adoption stage is completed the algorithm returns to the growth stage. The algorithm terminates when the search trees S and T can not grow.

Drawbacks of Boykov-Kolmogorov Algorithm

- Provable time bound is weaker $O(n^3C)$, where n is the number of nodes and C is the cost of minimum cut.
- Nodes touched in growth/adoption is large as there is little control over augmenting path lengths and the amount of flow pushed in each path.
- Total effort is spread over a very large number of flow augmentation iterations.

Voronoi based Preflow Push Algorithm (VPP)

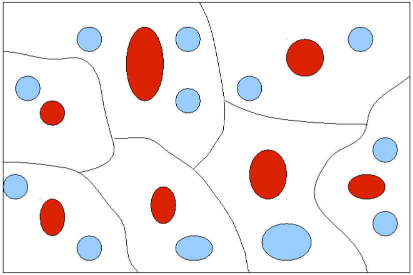
- Algorithm based on push-relabel method.
- Structure of Graph same as in B-K Algorithm.
- Flows in t-links are set equal to their capacities, and in n-links those are set to zero.
- Label the vertices other than s and t by their excesses $(inflow - outflow)$, and delete all t-links.
- Source-Sink max-flow problem is solved in the resultant flow graph by treating nodes with positive excesses as sources and those with negative excesses as sinks.
- All sinks are labeled as 0 and labels for sources are shortest distance to a sink node on a sink cluster boundary.

Major steps in VPP algorithm

- Initialization : Create Excess Lists (EL) per distance label containing source nodes with given distance label.
- Create Voronoi region graphs around sink clusters.
- In each Voronoi region, first push flow from source cluster to sink cluster boundary and then push flow within sink cluster.
- Rebuild Voronoi region graphs around remaining sink clusters.
- Repeat last two steps until all sink clusters are saturated.

Voronoi Region Graphs

- Collection of nodes in which there is a path between any two nodes passing through only nodes in the collection is called a *cluster*.
- Create Voronoi region graphs around each sink cluster as shown figure.
- Each Voronoi region contains one sink cluster and one or more source clusters which are reachable from given sink cluster.


Example of Voronoi Region Graph

Push Stage

- Push flow happens in two phases.
- In first phase, excess from sources is pushed to the boundaries of sink cluster in topological order.
    - From a node, flow is pushed saturating the out edges till the node has no excess left or all out edges of the node get saturated.
    - The saturated edges are deleted and a node whose all out-edges are deleted is inserted in a list called *Disconnected List(DL)*.
- Second phase moves the excess (which is accumulated at boundary nodes of a sink cluster), inside the sink cluster.
  - It starts from those sink cluster boundary nodes with positive excess on them and pushes the excess inside the cluster in a breadth first manner.

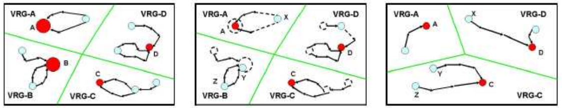
Rebuild Stage

- A node v is put in disconnected list during the Push flow stage because all paths from v to the sink of a Voronoi region graph have been saturated.
- Similarly, nodes in the Voronoi region graphs for which all paths to a sink pass through nodes which are already put in DL are also effectively disconnected.
- Rebuild stage first identifies all such nodes whose distance labels need to be corrected.

Rebuild Stage (cont.)

- An augmenting path from a node v in $DL(d)$ to a sink in the new residual graph will necessarily have to pass through a node which has its path to sink intact.
- If such path exists from node v will have to pass through a neighbor w not in its out-edge list (in earlier push phase).
- For such node w, its label $d(w)$ was greater than or equal to $d(v)$ in the Voronoi region graph in which flow was pushed.
- So, increase label of v to $d(w) + 1$.

Example

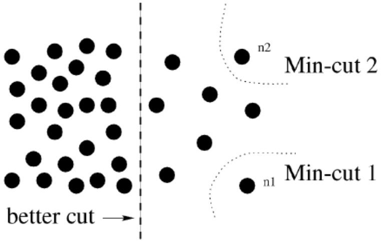


(a) After initialization/rebuild (b) After Push flow (c) After rebuilding Voronoi regions

Flow Graph states [2]

Sinks clusters are circles labeled A,B,C, and D. Rest of the circles are source clusters.
(a) : Four Voronoi regions corresponding to sink clusters A,B,C, and D.
(b) : Thick dashed lines indicate saturated edges while dashed circles show the changes in sources and sink clusters.
(c) : After Rebuild only three Voronoi regions are created corresponding to the remaining sink clusters.

Min-cut vs Normalized cut



Example showing “bad” minimum cuts [7]

Normalized Cut

Normalized Cut is defined as,

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

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$ measures overall association of partition A with whole graph.
Similarly, normalized association within groups for a given partition is:

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

where $assoc(A,A)$ and $assoc(B,B)$ are total weights of edges connecting nodes within A and B, respectively.

$$\Rightarrow Ncut(A,B) = 2 - Nassoc(A,B)$$

Optimal Partition

$\mathbf{x}$: $N = |V|$ dimensional vector containing elements x_i, $x_i = 1$ if i^{th} node $\in A$ and -1 otherwise.
$d(i) = \sum_j w(i,j)$ be sum of weights of all edges from node i to all other nodes.
$\mathbf{D}$: $N \times N$ diagonal matrix with $\mathbf{d}$ on its diagonal
$\mathbf{W}$: $N \times N$ symmetric matrix, $W(i,j) = w_{ij}$
$\mathbf{1}$: $N \times 1$ vector of all ones.
Now, min $Ncut$ is given by,

$$min_{\mathbf{x}} Ncut(\mathbf{x}) = min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \tag{1}$$

with constraint that $y^T\mathbf{D1} = 0$ and $y(i) \in \{1, -b\}$.

where $\mathbf{y} = (1+\mathbf{x}) - b(1-\mathbf{x}), b = \frac{k}{1-k}, k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$.

Optimal Partition

If $\mathbf{y}$ is relaxed to take on real values, we can minimize (1) by solving the generalized eigenvalue system,

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y} \tag{2}$$

- Second smallest eigenvector of the generalized eigensystem (2) is the real valued solution to normalized cut problem.
- Similarly the eigenvector with the third smallest eigenvalue is the real valued solution that optimally sub-partitions the first two parts.
- We can extended this argument to show that one can subdivide the existing graphs, each time using the eigenvector with the next smallest eigenvalue.

Grouping Algorithm I

- Construct a weighted graph $G = (V, E)$ by taking each pixel as node and connecting each pair of pixels by edge.
- Weight of edge connecting two nodes i and j can be defined as:

$$w_{ij} = e^{\frac{-\|F_{(i)} - F_{(j)}\|^2}{\sigma_I^2}} * \begin{cases} e^{\frac{-\|X_{(i)} - X_{(j)}\|^2}{\sigma_X^2}} & \text{if } \|X(i) - X(j)\| < r \\ 0 & \text{otherwise} \end{cases}$$

$X(i)$: Spatial location of node i, $F(i)$: feature vector (intensity, color, or texture information) at that node.

- Solve for the eigenvectors with the smallest eigenvalues of the system (2).
- Use the eigenvector with the second smallest eigenvalue to bipartition the graph.

Grouping Algorithm II

- Decide if the current partition should be subdivided by checking the stability of the cut.

Summary I

- Graph based methods are preferred over other approaches for solving Image Segmentation problem.
- In [4] new moves namely, α-β swap and α expansion were proposed.
- These moves allow large number of pixels to change their label as opposed to single pixel changing its label in standard moves.
- Min-cut problem is mainly solved by two methods: augmenting paths or push-relabel.
    - Boykov-Kolmogorov [3] Algorithm has given an efficient way to find augmenting path as we don't have to start from scratch while finding new augmenting path but use existing structure itself with minor arrangements.
      - Experimental Comparison done in [3] shows that this algorithm outperforms standard algorithms (in terms of time taken).
    - C. Arora et.al. [2] has proposed an algorithm namely "Voronoi based Push Preflow"(VPP) based on Push-relabel method.

Summary II

- VPP algorithm exploits structural properties inherent in image grid-graphs.
- It pushes flow in phases while relabeling is done only in global manner.
- It is observed[2] that number of nodes “touched” in each iteration reduces drastically which improves its time complexity significantly over BK algorithm.

- In [7], new approach called as “Normalized cuts” is proposed.
  - It focuses on extracting the global impression of an image rather than focusing on local features and their consistencies in the image data.
  - *Normalized cut* criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups.
  - An efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion.

References I

[1] graph-tool.
https://projects.skewed.de/graph-tool/doc/dev/flow.html.

[2] Chetan Arora, Subhashis Banerjee, Prem Kalra, and S. Maheshwari.
An efficient graph cut algorithm for computer vision problems.
In *Computer Vision ECCV 2010*, volume 6313 of *Lecture Notes in Computer Science*, pages 552–565. Springer Berlin / Heidelberg, 2010.

[3] Yuri Boykov and Vladimir Kolmogorov.
An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 26:1124–1138, 2004.

References II

[4] Yuri Boykov, Olga Veksler, and Ramin Zabih.
Fast approximate energy minimization via graph cuts.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 23:1222–1238, 2001.

[5] Pedro F. Felzenszwalb and Daniel P. Huttenlocher.
Efficient graph-based image segmentation.
International Journal of Computer Vision, 59(2):167–182, 2004.

[6] Maria Murgasova, Leigh Dyet, David Edwards, Mary Rutherford, Jo Hajnal, and Daniel Rueckert.
Segmentation of brain mri in young children.
Academic Radiology, 14(11):1350 – 1366, 2007.

References III

[7] Jianbo Shi and Jitendra Malik.
Normalized cuts and image segmentation.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:888–905, 2000.