Image Segmentation by Size-Dependent Single Linkage Clustering of a Watershed Basin Graph

Aleksandar Zlateski[1] and H. Sebastian Seung[2]

[1] Massachusetts Institute of Technology, Cambridge, USA
zlateski@mit.edu
[2] Princeton University, Princeton, USA
sseung@princeton.edu

Abstract. We present a method for hierarchical image segmentation that defines a disaffinity graph on the image, over-segments it into watershed basins, defines a new graph on the basins, and then merges basins with a modified, size-dependent version of single linkage clustering. The efficiency of the method makes is suitable for segmenting large 3D volumes. We illustrate the method on the challenging problem of segmenting 3D electron microscopic brain images.

Keywords: Watershed, image segmentation, hierarchical clustering, electron microscopy

1 Introduction

When segmenting very large volumes an efficient segmentation algorithms are required. We can obtain terabate scale volumes of EM data, we need close to linear time segmentation methods. Watershed is known to be fast but it tends to produce severe over-segmentation, which can be counteracted by pre- and/or post-processing.

Our watershed transform works by finding the basins of attraction of steepest descent dynamics, and has runtime that is linear in the number of graph edges. It yields basins similar to those of watershed cuts [1], except that plateaus are divided between basins consistently and in a more even way. Our post-processing starts by examining the new graph on the basins, in which the edge connecting two basins is assigned the same weight as the minimal edge connecting the basins in the original disaffinity graph. Then single linkage clustering yields a hierarchical segmentation of the original image in which the lowest level consists of the watershed basins. The levels of single linkage clustering yield flat segmentations in which some of the basins are merged. If we only expect to use levels above some minimum value $T_{\min}$, then it turns out to be equivalent and more efficient to preprocess the original disaffinity graph before watershed by setting all edge weights below $T_{\min}$ to a common low value. In another pre-processing step we remove the edges with disaffinity to allow for unsegmented regions.

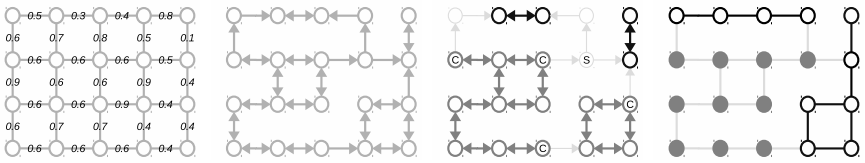
We also show how to modify single linkage clustering by making it depend not only on edge weights but also on cluster size. The modification is useful

when there is prior knowledge about the size of true segments, and is shown to have an efficient implementation because size is a property that is guaranteed to increase with each agglomerative step.

Related work. Linear time watershed transform algorithms on graphs are introduced by Coutsy *at al.* [1, 2]. Alternative close to linear time agglomerative clustering methods are introduced by [3] and [4]. Their methods are also based on satisfying binary predicates of cluster pairs. They start from the set of vertices as individual clusters and produces final segmentation in near linear time for integral and $O(|E| \log |E|)$ for real valued weights of the edges.

2 Watershed Transform

Inspired by the *drop of water principle* [1] we define a steepest descent discrete dynamics on a connected edge-weighted graph $G = (V, E)$ with non-negative weights. A water drop travels from a vertex to a vertex using only *locally minimal* edges. An edge $\{u, v\}$ is *locally minimal* with respect to u if there is no edge in E containing u with lower weight. Starting from a vertex v_0 the evolution of the system can be represented as a *steepest descent walk* $\langle v_0, e_0, v_1, e_1, v_2, \dots \rangle$ where every edge e_i is locally minimal in respect to v_i. A *regional minimum* M is a connected subgraph of G such that every *steepest descent walk* in G starting from a vertex in M will stay within M. A vertex v belongs to the *basin of attraction* of a *regional minimum* M if there exist a *steepest descent walk* from v to any vertex in M. Note that v can belong *basins of attractions* of multiple *regional minima*. In our *watershed transform* we partition V into *basins of attraction* of *the regional minima.* Vertices belonging to more than one *basin of attraction* will be referred to as *border vertices* and will be assigned to one of the *basins* as described below.



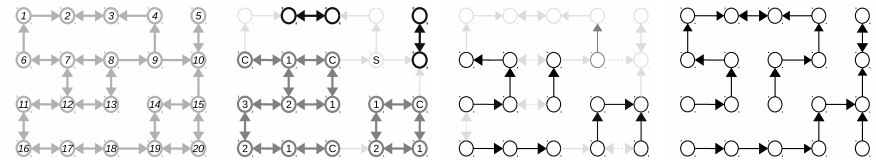
(a) (b) (c) (d)

Fig. 1. (a) An disaffinity graph; (b) derived steepest descent graph; (c) locally minimal plateaus (black), non-minimal plateau (gray), saddle vertex (S), plateau corners (C); (d) the two basins of attractions and border vertices (gray)

Steepest descent graph. The central quantity in the watershed algorithm is the steepest descent graph, defined as follows. Consider an undirected weighted graph G (Fig. 1(a)). Define the directed graph G' in which each undirected

edge of G is replaced by both directed edges between the same vertices. The *steepest descent graph* D (Fig. 1(b)) is a subgraph of G' with the property that D includes every edge of G' with minimal weight of all edges outgoing from the same vertex. A directed path in D is a path of steepest descent in G. The steepest ascent graph can be defined analogously using edges of maximal weight. Either steepest ascent or descent can be used without loss of generality. For simplicity, for a given vertex v we will refer to its edges in D as incoming, outgoing, and bidirectional. A *plateau* is a connected component of the subgraph of D containing only bidirectional edges. A *plateau corner* is a vertex of a *plateau* that has at least one outgoing edge. *Locally minimal plateaus* contain no *plateau corners,* they are equivalent to the regional minima of the original graph. *Non-minimal plateaus* contain one or more *plateau corners.* A *saddle vertex* has more than one outgoing edge. In Fig. 1(c) we show *locally minimal plateaus* (black), *non-minimal plateau* (gray), *plateau corners* (C), and a *saddle vertex* (S).

Assigning border vertices. In Fig. 1(d) we show the *basins of attraction* of the two *regional minima.* The *border* vertices are shown in gray and belong to both *basins of attraction.* Watershed cuts [1, 2] assign *border* vertices with a single constraint that all the *basins of attraction* have to be connected. We introduce additional constrains. The watershed transform has to be uniquely defined and the *non-minimal plateaus* should be divided evenly. More specifically, we want our dynamics to be uniquely defined at *saddle vertices, and* the vertices of the *non-minimal plateaus* to be assigned to the same *basin of attraction* as the nearest *plateau corner* - a *plateau corner* reachable in fewest steps following the rules of our dynamics.



(a) (b) (c) (d)

Fig. 2. (a) Vertex indices; (b) distances to the nearest plateau corner; (c) modifications to the steepest descent graph; (d) final watershed partition of the graph

Watershed transform algorithm. We introduce an ordering function $\alpha : V \to \{1, 2, ..., |V|\}$ such that $\alpha(u) \neq \alpha(v)$ if and only if $u \neq v$. We'll refer to $\alpha(u)$ as the index of u (Fig. 2(a)). In the first part of the algorithm we modify D by removing edges. For all *saddle vertices* we keep only one outgoing edge - the one pointing to a vertex with the lowest index. We then divide the *non-minimal plateaus.* We initialize a global FIFO queue Q. We then mark all the *plateau corner* vertices as visited and insert them into Q in increasing order of their

index. While Q is not empty we remove the vertex v from the top of the queue, we then follow all the bidirectional edges. If the vertex u on the other side of the edge is not visited, we mark it as such, insert it to the back of the queue and change the edge to be incoming. Otherwise, if the vertex was already visited we just remove the edge. The resulting steepest descent graph is shown on Fig. 2(c) - the dotted edges are removed. The connected components of the modified *descent graph* D, now considering all the remaining edges as bidirectional will be our *basins of attraction*.

The algorithm runs in linear time with respect to the number of edges in G and produces an optimal partitioning as defined in [1]. The total number of segments in the partitioning will equal to the total number of *regional minima*. We defer the detailed algorithm listing, the proof of correctness and running time analysis to the supplementary material.

Reducing over-segmentation. Often the edge weights of G represent *disaffinities* - confidence that the two vertices don't belong to the same segment. The *disaffinity* between two vertices not connected by an edge is considered to be infinite. The *saliency* of two adjacent segments is the value of the minimal *disaffinity* between the vertices of the two segments. Noisy values of *disaffinities* can produce severe over-segmentation (Fig. 3(c)). In order to reduce the over-segmentation we often merge adjacent segments with the *saliency* below some given threshold $T_{\min}$ [5]. We are confident that *disaffinities* below $T_{\min}$ should yield connected vertices. An equivalent segmentation can be obtained by replacing the weights of all edges in G with the weight smaller than $T_{\min}$ to a common low value (e.g. 0) before applying the watershed transform. We prove this claim in the supplementary material. To show confidence about high values of *disaffinities*, and in order to prevent undesired mergers, we introduce a threshold $T_{\max}$ by erasing all the edges from G with the weight higher than $T_{\max}$, and essentially setting them to ∞. The $T_{\max}$ threshold can produce singleton vertices in G. The singleton vertices are not assigned to any *basin of attraction* and are considered background, which is often a desired result.

3 Hierarchical Clustering of the Watershed Basin Graph

A hierarchical clustering of an undirected weighted graph treats each vertex as a singleton cluster and successively merges clusters connected by an edge in the graph. A cluster is always a connected subset of the graph's vertices. Each merge operations creates a new level of the hierarchy - a flat segmentation where each cluster represents a segment. In *single linkage* clustering, each step we merges two clusters connected by an edge with the lowest weight in the original graph. *Single linkage* clustering is equivalent to finding the minimum spanning tree of the graph [6].

In this section we propose a size-dependent single linkage clustering. The method can be applied to any edge weighted graph, however we find it superior when used on the *watershed basin graph* defined as follows. Let $V_W = \{B_1, B_2, \dots\}$ be the set of *watershed basins* obtained by the watershed trans-

form of a graph $G = (V, E)$. We define the watershed basin graph of G as $G_W = (V_W, E_W)$ where an edge $\{B_i, B_j\}$ exists in E_W for all neighboring basins B_i and B_j, and has the weight $w(\{B_i, B_j\})$ equal to the *saliency* of the two *basins*. We will refer to the vertices of the *watershed basin graph* as *basins* and to the edge weights as *saliencies*.

In our size-dependent *single linkage* clustering method, in each step we merge clusters with the lowest *saliency* that don't satisfy a given predicate. *Saliency* of two clusters is defined as the minimal *saliency* of any two members:

$$d_{C_1,C_2} = \min_{B_i \in C_1, B_j \in C_2, \{B_i,B_j\} \in E_W} w(\{B_i, B_j\}) \tag{1}$$

At the last level of the hierarchy all pairs of clusters will satisfy the predicate.

Size-dependent comparison predicate. We define a predicate Λ, for evaluating whether two clusters should be merged. The predicate is based on the sizes of the two clusters. Let $S(C)$ represent the size of C (e.g. number of *basins* in the cluster or the sum of the *basin* sizes). The size can be any measure of a cluster with the property that $S(C_1 \cup C_2) \geq S(C_1)$. We first define a non-increasing threshold function of a cluster size $\tau(s)$. The value of $\tau(s)$ represents the maximal *saliency* allowed between a cluster of size s and any adjacent cluster. Our predicate is then defined as:

$$\Lambda(C_1, C_2) = \begin{cases} \text{true} & \text{if } d_{C_1,C_2} \geq \tau\left(\min\left\{S(C_1), S(C_2)\right\}\right) \\ \text{false} & \text{otherwise} \end{cases} \tag{2}$$

The intuition behind the predicate is to apply prior knowledge about the sizes of the true segments. With the threshold function we control the confidence required to grow a cluster of a certain size.

With a slight modification of the predicate we could allow for an arbitrary threshold function (changing the condition to $d_{C_1,C_2} \geq \min\left\{\tau(S(C_1)), \tau(S(C_2))\right\}$). However, restricting the function to be non-decreasing allows us to design a more efficient algorithm. It is also more intuitive to allow higher *saliency* for merging small clusters and require lower *saliency* as the sizes of the clusters grow. As τ is required to be non-decreasing, we can rewrite the condition in (2) as $\tau^{-1}(d_{C_1,C_2}) \geq \min\left\{S(C_1), S(C_2)\right\}$ allowing us to specify either τ or τ^{-1}. When τ^{-1} is constant the algorithm will tend to aggresively merge segments smaller than the given constant.

Algorithm 1 In our clustering algorithm we visit all the edges of the watershed basin graph in non-decreasing order and merge the corresponding clusters based on the introduced predicate.

1. Sort E_W into $\pi(e_1, \ldots, e_n)$, by non-decreasing edge weight.
2. Start with basins as singleton clusters $S^0 = \{C_1 = \{B_1\}, C_2 = \{B_2\}, \ldots\}$
3. Repeat step 4 for $k = 1, \ldots, n$
4. Construct S^k from S^{k-1}. Let $e_k = \{B_i, B_j\}$ be the k-th edge in the ordering. Let C_i^{k-1} and C_j^{k-1} be components of S^{k-1} containing B_i and B_j. If $C_i^{k-1} \neq C_j^{k-1}$ and $\Lambda(C_i^{k-1}, C_j^{k-1})$ then S^k is created from S^{k-1} by merging C_i^{k-1} and C_j^{k-1}, otherwise $S^k = S^{k-1}$.

5. Return the hierarchical segmentation $(S^0, \ldots, S^n)$

Theorem 1 *The highest level of the hierarchical segmentation produced by algorithm (1) will have the predicate Λ satisfied for all pairs of the clusters. The complexity of the algorithm is $|E_W| \log |E_W|$. The algorithm can be modified to consider only the edges of the minimum cost spanning tree of G_W.*

We defer the proof the supplementary material.

The steps 2-5 of the algorithm have near linear complexity. Once we have a sorted list of the edges we can re-run the algorithm for different threshold functions more efficiently.

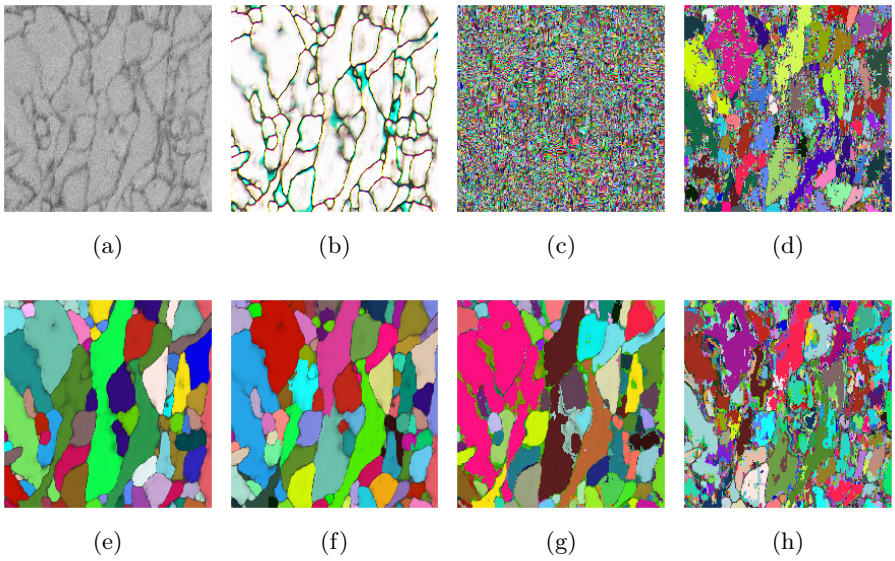
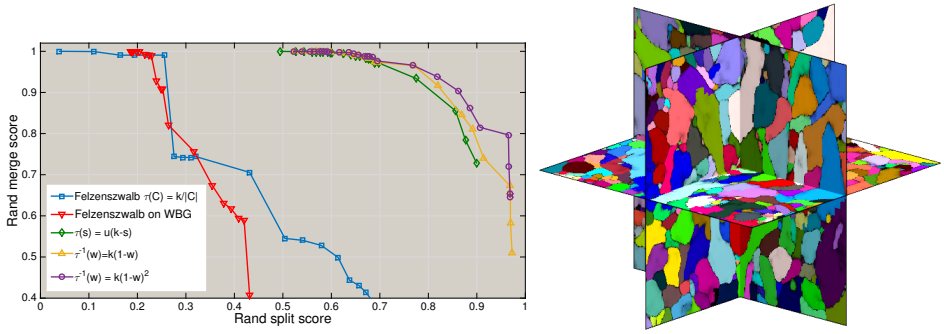


Fig. 3. (a) EM image; (b) volumetric disaffinity graph, the 3 directions of disaffinities are represented with RGB; (c) watershed transform output; (d) watershed transform output with $T_{\min} = 0.01$ and $T_{\max} = 0.9$; (e) ground truth; (f) our method with $\tau^1(w) = 3000\,(1 - w)$; (g), (h) segmentation by *[3]* with $k = 0.5$ and $k = 10$ respectivelly.

4 Results

We have applied our method to 3D electron microscopic brain images obtained by [cite e2198] (Fig. 3(a)). We obtained disaffinity graphs using convolutional neural networks described by Turaga *et al.* [7, 8] (Fig. 3(b)). We define the size of a segment to be the number of 3D pixels contained inside the segment. In Fig 3. we

show the results of the watershed transform (Fig. 3(c)), the watershed transform on the pre-processed graph with thresholds $T_{\min} = 0.01$ and $T_{\max} = 0.9$ (Fig. 3(d)). We have tested our method on three intuitive, paramatrized, threshold functions. The first one enforces all the segments to be at least some size, second and the third one require the minimal size of the segment to be proportional to the affinity (or the square of affinity).



(a) (b)

Fig. 4. (a) Performance of our method and the method described in [3]; (b) Segmentation obtained by our method with $\tau^{-1}(w) = 3000(1-w)$

Measuring the quality of the segmentations. A good metric for evaluating a segmentation should ideally tell us how good the segmentation is for using in practice, how much manual corrections are needed to acheve the correct segmentation. Often foreground-restricted Rand Index [9] between the proposed segmentation and the ground truth is used. However it has problems when the number of segments is large as the dynamics range gets smaller. A better metric is introduced by introduced by Arganda at al, separating the rand indices for splits and mergers. Let p_{ij} is the probability that a randomly chosen pixel belongs to segiment i in the proposed segmentation and segment j in the ground truth, let s_i and t_j be the probability of randombly chosen pixel beloning to the component i in the proposed segmentation and component j in the ground truth respectivelly. The Rand split score and Rand merge score are defined as follows, with the higher value representing better segmentation:

$$V_{\text{split}}^{\text{Rand}} = \frac{\sum_{ij} p_{ij}^2}{\sum_k t_k^2} \text{ and } V_{\text{merge}}^{\text{Rand}} = \frac{\sum_{ij} p_{ij}^2}{\sum_k s_k^2} \tag{3}$$

In Fig. 4(a) we compare our method with various threshold functions to the method described in[3] applied both to the original graph and the *watershed basin graph*. Our method shows superior results, a sample segmentation is shown on Fig. 4(b).

Our method greatly outperforms [3]. A sentence explaining the results (why would we expect this?)

5 Conclusions

Our method is superior for segmenting images where we have prior knowledge about the size of the true segments, which makes it ideal for segmenting 3D EM images. For conservative threshold functions our method can greatly reduce the number of segments without introducing mergers, which makes it ideal as pre processing method for some other methods. Final segmentation can be obtained by merging our segments. It would be a good input to a lot of other methods like [10]prior knowledge about segments assumes all have the same size properties. Useful when proofreading done by non-experts (just by merging segments) eyewire ref,

References

[1] Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Minimum spanning forests and the drop of water principle. IEEE Transactions on Pattern Analysis and Machine Intelligence **31** (2009) 1362–1374

[2] Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest path forests, and topological watersheds. IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010) 925–939

[3] Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59** (2004) 167–181

[4] Guimarães, S.J.F., Cousty, J., Kenmochi, Y., Najman, L.: A Hierarchical Image Segmentation Algorithm Based on an Observation Scale. (2012) 116–125

[5] Najman, L., Schmitt, M.: Geodesic Saliency of Watershed Contours and Hierarchical Segmentation. Analysis **18** (1996) 1163–1173

[6] Gower, J.C., Ross, G.J.S.: Minimum Spanning Trees and Single Linkage Cluster Analysis. Journal of the Royal Statistical Society. Series C (Applied Statistics) **18** (1969) 54–64

[7] Turaga, S., Briggman, K.: Maximin affinity learning of image segmentation. arXiv preprint arXiv: ... (2009) 1–9

[8] Turaga, S.C., Murray, J.F., Seung, H.S.: Convolutional Networks Can Learn to Generate Affinity. **538** (2010) 511–538

[9] Rand, W.M..: Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association **66** (1971) 846–850

[10] Mustafa, G.: Optree : a Learning-Based Adaptive Watershed Algorithm for Neuron Segmentation. (**1**) 1–8