The Mutex Watershed: Efficient, Parameter-Free Image Partitioning

Steffen Wolf[1⋆], Constantin Pape[1,2⋆], Alberto Bailoni[1], Nasim Rahaman[1], Anna Kreshuk[1,2], Ullrich Köthe[1], and Fred A. Hamprecht[1]

[1] HCI/IWR, University of Heidelberg, Germany
{firstname.lastname}@iwr.uni-heidelberg.de
[2] EMBL Heidelberg, Germany

Abstract. Image partitioning, or segmentation without semantics, is the task of decomposing an image into distinct segments; or equivalently, the task of detecting closed contours in an image. Most prior work either requires seeds, one per segment; or a threshold; or formulates the task as an NP-hard signed graph partitioning problem. Here, we propose an algorithm with empirically linearithmic complexity. Unlike seeded watershed, the algorithm can accommodate not only attractive but also repulsive cues, allowing it to find a previously *unspecified* number of segments without the need for explicit seeds or a tunable threshold. The algorithm itself, which we dub "Mutex Watershed", is closely related to a minimal spanning tree computation. It is deterministic and easy to implement. When presented with short-range attractive and long-range repulsive cues from a deep neural network, the Mutex Watershed gives results that currently define the state-of-the-art in the competitive ISBI 2012 EM segmentation benchmark. These results are also better than those obtained from other recently proposed clustering strategies operating on the very same network outputs.

1 Introduction

Most image partitioning algorithms are defined over a graph encoding purely attractive interactions. No matter whether a segmentation or clustering is then found agglomeratively (as in single linkage clustering / watershed) or divisively (as in spectral clustering or iterated normalized cuts), the user either needs to specify the desired number of segments or a termination criterion. An even stronger form of supervision is in terms of seeds, where one pixel of each segment needs to be designated as such either by a user or automatically. Unfortunately, clustering with automated seed selection remains a fragile and error-fraught process, because every missed or hallucinated seed causes an under- or oversegmentation error. Although the learning of good edge detectors boosts the quality of classical seed selection strategies (such as finding local minima of the boundary map, or thresholding boundary maps), non-local effects of seed placement along

⋆ Authors contributed equally.

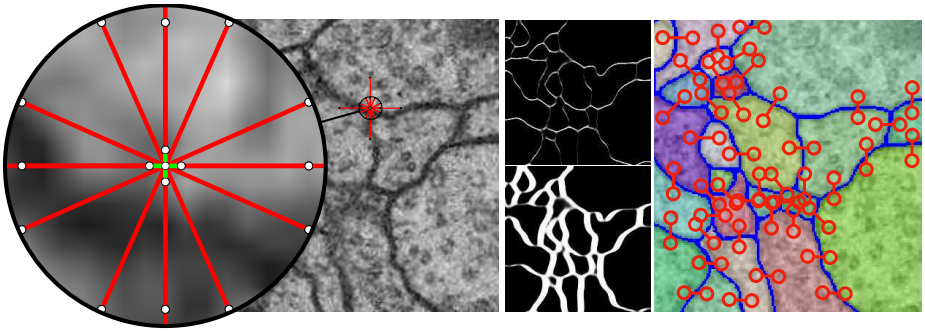


Fig. 1: Left: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. Middle: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. Right: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

with strong variability in region sizes and shapes make it hard for any learned predictor to place *exactly one* seed in every true region.

In contrast to the above class of algorithms, multicut / correlation clustering partitions vertices with both attractive and repulsive interactions encoded into the edges of a graph. Multicut has the great advantage that a "natural" partitioning of a graph can be found, without needing to specify a desired number of clusters, or a termination criterion, or one seed per region. Its great drawback is that its optimization is NP-hard.

The main insight of this paper is that when both attractive and repulsive interactions between pixels are available, then a generalization of the watershed algorithm can be devised that segments an image *without* the need for seeds or stopping criteria or thresholds. It examines all graph edges, attractive and repulsive, sorted by their weight and adds these to an active set iff they are not in conflict with previous, higher-priority, decisions. The attractive subset of the resulting active set is a forest, with one tree representing each segment. However, the active set can have loops involving more than one repulsive edge. See Fig. 1 for a visual abstract.

In summary, our principal contribution, the Mutex Watershed, is a "best of both worlds" algorithm that combines the multicut's desirable lack of hyperparameters with the small computational footprint of Kruskal-type watershed algorithm.

The algorithm is presented in section 3. In section 4 we evaluate the algorithm against very strong baselines. We choose a challenging dataset for neuron segmentation from electron microscopy (EM) image stacks as benchmark. For this task, watershed segmentation is a key component: EM staining only highlights membrane boundaries, discouraging the use of region cues for segmentation. By incorporating long-range repulsions into the watershed procedure, we can obtain

an accurate segmentation from this step already, avoiding costly post-processing for agglomeration. In addition, we present preliminary results on the BSDS500, demonstrating the applicability of the proposed method to natural images. We describe our future plans, including extensions to semantic segmentation, in section 5. Our implementation is available at https://github.com/hci-unihd/mutex-watershed.git.

2 Related Work

In the original watershed algorithm [1], seeds were automatically placed at all local minima of the boundary map. Unfortunately, this leads to severe over-segmentation. Defining better seeds has been a recurring theme of watershed research ever since. The simplest solution is offered by the seeded watershed algorithm [2]: It relies on an oracle (an external algorithm or a human) to provide seeds and assigns each pixel to its nearest seed in terms of minimax path distance. In the absence of an oracle, automatic seed selection is challenging because *exactly one* seed must be placed in every region. Simple methods, e.g. defining seeds by connected regions of low boundary probability, do not work: The segmentation quality is usually insufficient because multiple seeds are in the same region and/or seeds leak through the boundary.

This problem is typically addressed by biasing seed selection towards over-segmentation (with seeding at all minima being the extreme case). The watershed algorithm then produces superpixels that are merged into final regions by more or less elaborate postprocessing. This works better than using watersheds alone because it exploits the larger context afforded by superpixel adjacency graphs. Many criteria have been proposed to identify the regions to be preserved during merging, e.g. region dynamics [3], the waterfall transform [4], extinction values [5], region saliency [6], and (α, ω)-connected components [7]. A merging process controlled by criteria like these can be iterated to produce a hierarchy of segmentations where important regions survive to the next level. Variants of such hierarchical watersheds are reviewed and evaluated in [8].

These results highlight the close connection of watersheds to hierarchical clustering and minimum spanning trees/forests [9,10], which inspired novel merging strategies and termination criteria. For example, [11] simply terminated hierarchical merging by fixing the number of surviving regions beforehand. [12] incorparate predefined sets of generalized merge constraints into the clustering algorithm. Graph-based segmentation according to [13] defines a measure of quality for the current regions and stops when the merge costs would exceed this measure. Ultrametric contour maps [14] combine the gPb (global probability of boundary) edge detector with an oriented watershed transform. Superpixels are agglomerated until the ultrametric distance between the resulting regions exceeds a learned threshold. An optimization perspective is taken in [15], which introduces h-increasing energy functions and builds the hierarchy incrementally such that merge decisions greedily minimize the energy. The authors prove that

the optimal cut corresponds to a different unique segmentation for every value of a free regularization parameter.

An important line of research is based on the observation that superior partitionings are obtained when the graph has both attractive and repulsive edges. Solutions that optimally balance attraction and repulsion do not require external stopping criteria such as predefined number of regions or seeds. This generalization leads to the NP-hard problem of correlation clustering or (synonymous) multicut (MC) partitioning [16]. Fortunately, modern integer linear programming solvers in combination with incremental constraint generation can solve problem instances of considerable size [17], and good approximations exist for even larger problems [18,19].

Another beneficial extension is the introduction of additional long-range edges. Thanks to their larger field of view, the strength of these edges can often be estimated with greater certainty than is achievable for the local edges used in standard watersheds. This has been used in [20] to represent object size constraints by repulsive long-range edges, which is still an MC-type problem. When long-range edges are also allowed to be attractive, the problem turns into the more complicated lifted multicut (LMC) [21]. Realistic problem sizes can only be solved approximately [22,23], but watershed superpixels followed by LMC postprocessing achieve state-of-the-art results on important benchmarks [24]. Long-range edges are also used in [25], as side losses for the boundary detection CNN; but they are not used explicitly in any downstream inference.

In general, striking progress in watershed-based segmentation has been achieved by learning boundary maps with convolutional neural networks (CNNs). This is nicely illustrated by the evolution of neurosegmentation for connectomics, an important field we also address in the experimental section. CNNs were introduced to this application in [26] and became, in much refined form [27], the winning entry of the ISBI 2012 Neuro-Segmentaion Challenge [28]. Boundary maps and superpixels were further improved by progress in CNN architectures and data augmentation methods, using U-Nets [29], FusionNets [30] or inception modules [24]. Subsequent postprocessing with the GALA algorithm [31,32], conditional random fields [33] or the lifted multicut [24] pushed the envelope of final segmentation quality. MaskExtend [34] applied CNNs to both boundary map prediction and superpixel merging, while flood-filling networks [35] eliminated superpixels all together by training a recurrent neural network to perform region growing one region at a time.

Most networks mentioned so far learn boundary maps on pixels, but learning works equally well for edge-based watersheds, as was demonstrated in [36,37] using CNN-generated edge weights according to [38,39]. Tailoring the learning objective to the needs of the watershed algorithm by penalizing critical edges along minimax paths [39] or end-to-end training of edge weights and region growing [40] improved results yet again.

Outside of connectomics, [41] obtained superior boundary maps from CNNs by learning not just boundary strength, but also its gradient direction. Holistically-nested edge detection [42,43] couples the CNN loss at multiple resolutions using

deep supervision and is successfully used as a basis for watershed segmentation of medical images in [44].

The present paper combines all these concepts (hierarchical clustering, attractive and repulsive interactions, long-range edges, and CNN-based learning) into a novel efficient segmentation framework. It can be interpreted as a generalization of [12], because we also allow for soft constraints (which can be overridden by strong attractive edges), and constraints are generated on the fly by a neural network rather than predefined. Our method is also related to greedy additive edge contraction (GAEC) according to [22], but we handle attractive and repulsive interactions separately and define edge strength between clusters by a maximum instead of an additive rule.

3 The Mutex Watershed Algorithm

3.1 Definitions and notation

We consider the problem of clustering a graph $G(V, E^+ \cup E^-, W^+ \cup W^-)$ with both attractive and repulsive edge attributes. The scalar attribute $w_e^+ \in \mathbb{R}_0^+$ associated with edge $e \in E^+$ is a merge affinity: the higher this number, the higher the inclination of the two incident vertices to be assigned to the same cluster. Similarly, $w_e^- \in \mathbb{R}_0^+$ for $e \in E^-$ is a split tendency: the higher this number, the greater the tendency of the incident vertices to be in different clusters.

In our application, each vertex corresponds to one pixel in the image to be segmented. Two vertices may have no edge connecting them; or an attractive edge $e \in E^+$; or a repulsive edge $e \in E^-$; or two edges at the same time, one attractive and one repulsive. Edges can be either *local/short-range* (when connecting two pixels that are immediately adjacent in the image) or *long-range*.

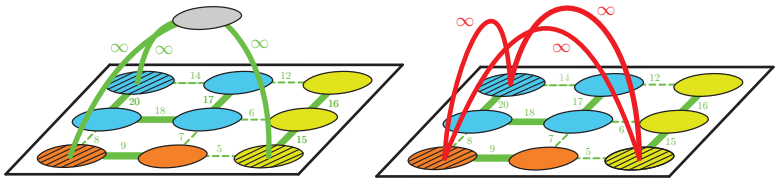
The Mutex Watershed algorithm, defined in subsection 3.3, maintains disjunct active sets $A^+ \subseteq E^+$, $A^- \subseteq E^-$, $A^+ \cap A^- = \emptyset$, that encode merges and mutual exclusion constraints, respectively. Clusters are defined via the "connected" predicate:

$$\begin{aligned} \forall i, j \in V: \qquad \Pi_{i \to j} &= \{\text{path } \pi \text{ from } i \text{ to } j \text{ with } \pi \subseteq E^+\} \\ \text{connected}(i, j) &\Leftrightarrow \exists \text{ path } \pi \in \Pi_{i \to j} \text{ with } \pi \subseteq A^+ \subseteq E^+ \\ \text{cluster}(i) &= \{i\} \cup \{j : \text{connected}(i, j)\} \end{aligned}$$

Conversely, the active subset $A^- \subseteq E^-$ of repulsive edges defines mutual exclusion relations by using the following predicate:

$$\begin{aligned} \text{mutex}(i, j) \Leftrightarrow \exists\, e = (k, l) \in A^- \text{ with} \\ k \in \text{cluster}(i) \text{ and } l \in \text{cluster}(j) \text{ and} \\ \text{cluster}(i) \neq \text{cluster}(j) \end{aligned}$$

Admissible active edge sets A^+ and A^- must be chosen such that the resulting clustering is consistent, i.e. nodes engaged in a mutual exclusion constraint



(a) (b)

Fig. 2: Two equivalent representations of the seeded watershed clustering obtained using (a) a maximum spanning tree computation or (b) Algorithm 1. Both graphs share the weighted attractive (green) edges and seeds (hatched nodes). The infinitely attractive connections to the auxiliary node (gray) in (a) are replaced by infinitely repulsive (red) edges between each pair of seeds in (b). The two final clusterings are defined by the active sets (bold edges) and are identical. Node colors indicate the clustering result, but are arbitrary.

cannot be in the same cluster: $\text{mutex}(i, j) \Rightarrow \text{not connected}(i, j)$. The "connected" and "mutex" predicates can be efficiently evaluated using a union find data structure.

3.2 Seeded watershed from a mutex perspective

One interpretation of the proposed method is in terms of a generalization of the edge-based watershed algorithm [45,46,9] or image foresting transform [47]. This algorithm can only ingest a graph with purely attractive interations, $G(V, E^+, W^+)$. Without further constraints, the algorithm would yield only the trivial result of a single cluster comprising all vertices. To obtain more interesting output, an oracle needs to provide seeds, namely precisely one node per cluster. These seed vertices are all connected to an auxiliary node (see Fig. 2 (a)) by auxiliary edges with infinite merge affinity. A maximum spanning tree (MST) on this augmented graph can be found in linearithmic time; and the maximum spanning tree (or in the case of degeneracy: at least one of the maximum spanning trees) will include the auxiliary edges. When the auxiliary edges are deleted from the MST, a forest results, with each tree representing one cluster [9,45,47].

We now reformulate this well-known algorithm in a way that will later emerge as a special case of the proposed Mutex Watershed: we eliminate the auxiliary node and edges, and replace them by a set of infinitely repulsive edges, one for each pair of seeds (Fig. 2 (b)). Algorithm 1 is a variation of Kruskal's MST algorithm operating on the seed mutex graph just defined, and gives results identical to seeded watershed on the original graph.
This algorithm differs from Kruskal's only by the check for mutual exclusion in the if-statement. Obviously, the modified algorithm has the same effect as the original algorithm, because the final set A^+ is exactly the maximum spanning forest obtained after removing the auxiliary edges from the original solution.

Input: weighted graph $G(V, E^+, W^+)$ and seeds $S \subseteq V$, such that $E^- = \{(s_i, s_j) | i, j \in 1, \ldots, |S|; i \neq j\}$ is the set of infinitely repulsive edges between all pairs of seeds;
Output: clusters defined by activated edges A^+;
Initialization: $A^+ = \emptyset$; $A^- = E^-$;
for $(i, j) = e \in E^+$ in descending order of w^+ **do**
  if not connected(i, j) **and not** mutex(i, j) **then**
    $A^+ \leftarrow A^+ \cup e$;
    ▷ merge i and j and inherit the mutex constraints of the parent clusters
  end
end

Algorithm 1: Mutex version of seeded watershed algorithm.

In the sequel, we generalize this construction by admitting less-than-infinitely repulsive edges. Importantly, these can be dense and are hence much easier to estimate automatically than seeds with their strict requirement of only-one-per-cluster.

3.3 Mutex Watersheds

We now introduce the core contribution: an algorithm that is empirically no more expensive than a MST computation; but that can ingest both attractive and repulsive cues and partition a graph into a number of clusters that does not need to be specified beforehand. There is no requirement of one seed per cluster, and not even of a hyperparameter that would implicitly determine the number of resulting clusters.

The Mutex Watershed, Algorithm 2, proceeds as follows: given a graph with sets of attractive and repulsive edges E^+ and E^-, with edge weights W^+ and W^- respectively, do the following: sort all edges $E^+ \cup E^-$, attractive or repulsive, by their weight in descending order into a priority queue. Iteratively pop all edges from the queue and add them to the active set one by one, provided that a set of conditions are satisfied. More specifically, if the next edge popped from the priority queue is attractive and its incident vertices are not yet in the same tree, then connect the respective trees provided this is not ruled out by a mutual exclusion constraint. If on the other hand the edge popped is repulsive, and if its incident vertices are not yet in the same tree, then add a mutual exclusion constraint between the two trees.
The crucial difference to algorithm 1 is that mutex constraints are no longer pre-defined, but created dynamically whenever a repulsive edge is found. However, new exclusion constraints can never override earlier, high-priority merge decisions. In this case, the repulsive edge in question is simply ignored. Similarly, an attractive edge must never override earlier and thus higher-priority must-not-link decisions.

Input: weighted graph $G(V, E^+ \cup E^-, W^+ \cup W^-)$;
Output: clusters defined by activated edges A^+;
Initialization: $A^+ = \emptyset$; $A^- = \emptyset$;
for $(i, j) = e \in E^+ \cup E^-$ in descending order of $W^+ \cup W^-$ **do**
  if $e \in E^+$ **then**
    if not connected(i, j) **and not** mutex(i, j) **then**
      merge(i, j): $A^+ \leftarrow A^+ \cup e$;
        ▷ merge i and j and inherit the mutex constraints of the parent clusters
    end
  else
    if not connected(i, j) **then**
      addmutex(i, j): $A^- \leftarrow A^- \cup e$;
        ▷ add mutex constraint between i and j
    end
  end
end

Algorithm 2: Mutex Watershed

3.4 Time Complexity Analysis

Before analyzing the time complexity of algorithm 2 we first review the complexity of Kruskal's algorithm. Using a union-find data structure the time complexity of merge(i, j) and connected(i, j) is $\mathcal{O}(\alpha(V))$, where α is the slowly growing inverse Ackerman function, and the total runtime complexity is dominated by the initial sorting of the edges $\mathcal{O}(E \log E)$[48].

To check for mutex constraints efficiently, we maintain a set of all active mutex edges

$$M[C_i] = \{(u,\ v) \in A^- | u \in C_i \vee v \in C_i\}$$

for every C_i = cluster(i) using hash tables, where insertion of new mutex edges (i.e. addmutex) and search have an average complexity of $\mathcal{O}(1)$. Note that every cluster can be efficiently identified by its union-find root node. For mutex(i, j) we check if $M[C_i] \cap M[C_j] = \emptyset$ by searching for all elements of the smaller hash table in the larger hash table. Therefore mutex(i, j) has an average complexity of $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$. Similarly, during merge(i, j), mutex constraints are inherited by merging two hash tables, which also has an average complexity $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$.

In conclusion, the average runtime contribution of attractive edges $\mathcal{O}(|E^+| \cdot \alpha(V) + |E^+| \cdot M)$ (checking mutex constrains and possibly merging) and repulsive edges $\mathcal{O}(|E^-| \cdot \alpha(V) + |E^-|)$ (insertion of one mutex edge) result in a total average runtime complexity of algorithm 2:

$$\mathcal{O}(E \log E + E \cdot \alpha(V) + EM). \tag{1}$$

where M is the expected value of $\min(|M[C_i]|, |M[C_j]|)$. Using $\alpha(V) \in \mathcal{O}(\log V) \in \mathcal{O}(\log E)$ this simplifies to

$$\mathcal{O}(E \log E \ + \ EM). \tag{2}$$

In the worst case $\mathcal{O}(M) = \mathcal{O}(E)$, the Mutex Watershed Algorithm has a runtime complexity of $\mathcal{O}(E^2)$. Empirically, we find that $\mathcal{O}(EM) \approx \mathcal{O}(E \log E)$ by measuring the runtime of Mutex Watershed for different sub-volumes of the ISBI challenge (see Figure 3), leading to a

$$\text{Empirical Mutex Watershed Complexity: } \mathcal{O}(E \log E) \tag{3}$$

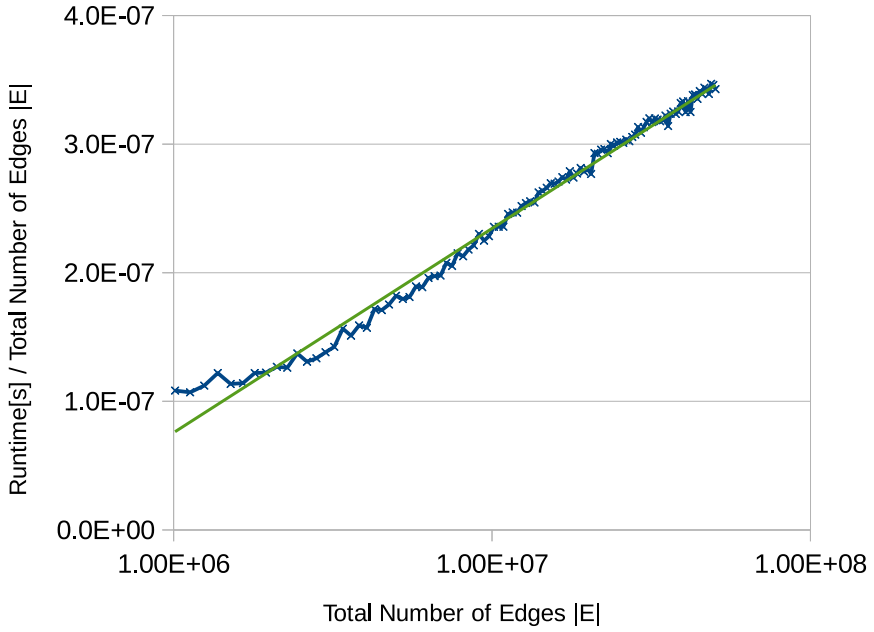


Fig. 3: Runtime T of Mutex Watershed (without sorting of edges) measured on sub-volumes of the ISBI challenge of different sizes (thereby varying the total number of edges E). We plot $\frac{T}{|E|}$ over $|E|$ in a logarithmic plot, which makes $T \sim |E|log(|E|)$ appear as straight line. A logarithmic function (green line) is fitted to the measured $\frac{T}{|E|}$ (blue crosses) with ($R^2 = 0.9896$). The good fit suggests that empirically $T \approx \mathcal{O}(E \log E)$.

4 Experiments

We evaluate the Mutex Watershed on the challenging task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuro-science that strives to reconstruct neural wiring diagrams spanning complete central nervous systems. The task requires segmentation of neurons from electron microscopy images of neural tissue – a challenging endeavor, since segmentation has to be based only on boundary information (cell membranes) and some of the boundaries are not very pronounced. Besides, cells contain membrane-bound organelles, which have to be suppressed in the segmentation. Some of the neuron protrusions are very thin, but all of those have to be preserved in the segmentation to arrive at the correct connectivity graph. While a lot of progress has been made recently, only manual tracing yields sufficient accuracy for correct circuit reconstruction [49].

We validated the Mutex Watershed algorithm on the most popular neural segmentation challenge: ISBI2012 [28]. We estimate the edge weights using a CNN as described in 4.1 and compare with other entries in the leaderboard as well as with other common post-processing methods for the same network predictions 4.2.

4.1 Estimating edge weights with a CNN

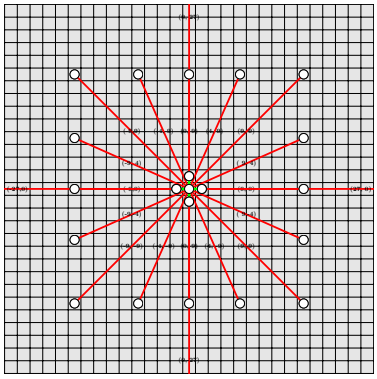
The common approach to EM segmentation is to predict which pixels belong to a cell membrane using a CNN. Different post-processing methods are used on top to obtain a segmentation, see section 2 for an overview of these methods. The CNN can be either trained to predict boundary pixels [27,24] or undirected affinities [25,50] which express how likely it is for a pixel to belong to a different cell than its neighbors in the 6-neighborhood. In this case, the output of the network contains three channels, corresponding to left, down and next imaging plane neighbors in 3d. The affinities do not have to be limited to immediate neighbors - in fact, [25] have shown that introduction of long-range affinities is beneficial for the final segmentation even if they are only used as auxiliary loss during training. Building on the work of [25], we train a CNN to predict short and long-range affinities and then use those directly as weights for the Mutex Watershed algorithm.

We estimate the affinities/edge weights for the neighborhood structure shown in Figure 4. To that end, we define local attractive and long-range repulsive edges. The choice of this structure has to be motivated by the underlying data - we use a different pattern for in-plane and between-plane edges due to the anisotropy of the validation datasets. In more detail, we picked a sparse ring of in-plane repulsive edges and additional longer-range in-plane edges which were necessary to split regions reliably (see Figure 4a). We also added connections to the indirect neighbors in the lower adjacent slice to ensure correct 3D connectivity (see Figure 4b).

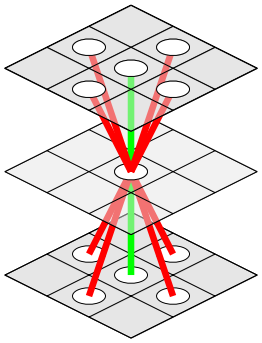
In total, C^+ attractive and C^- repulsive edges are defined for each pixel, resulting in $C^+ + C^-$ output channels in the network. We partition the set of

attractive / repulsive edges into subsets H^+ and H^- that contain all edges at a specific offset, attractive edges: $E^+ = \bigcup_c^{C^+} H_c^+$ and repulsive edges analogously. Each element of the subsets H_c^+ and H_c^- corresponds to a specific channel predicted by the network. We further assume that weights take values in $[0, 1]$ and adopt the same conventions for attraciveness / repulsion as in section 3. For more details on network architecture and training see Supplementary 1.

In our experiments, we pick a subset of repulsive edges, by using strides of 2 in the XY-plane in order to avoid artifacts caused by occasional very thick membranes. Note that the stride is not applied to local (attractive) edges, but only to long-range (repulsive) edges.



(a) XY-plane neighborhood with local attractive edges, a sparse repulsive edges with approximate radius 9 and further long-range connections with distance 27



(b) Due to the high anisotropy of the data we limit the Z-plane edges to a distance of 1. The direct neighbors are attractive; the indirect neighbors are repulsive.

Fig. 4: Local neighborhood structure of attractive (green) and repulsive (red) edges in the Mutex Watershed graph. Due to point symmetry to the origin, we only predict half of the directions with the neural network.

4.2 ISBI Challenge

The ISBI 2012 EM Segmentation Challenge [28] is the neuron segmentation challenge with the largest number of competing entries. The challenge data contains two volumes of dimensions $1.5 \times 2 \times 2$ microns with a resolution of $50 \times 4 \times 4$ nm per pixel. The groundtruth is provided as binary membrane labels, which can easily be converted to a 2D, but not 3D segmentation. To train a 3D model, we follow the procedure described in [24].

The test volume has private groundtruth; results can be submitted to the leaderboard. They are evaluated based on the Adapted Rand Score (Rand-Score) and the Variation of Information Score (VI-Score) [28], separately for each 2D slice.

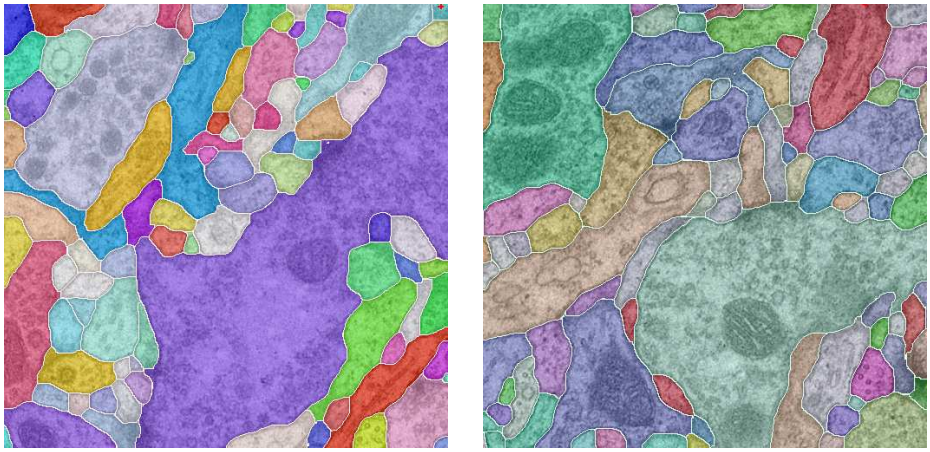


Fig. 5: Mutex Watershed applied on the ISBI Challenge test data. For further images and a detailed comparison to the baseline segmentation methods view Supplementary Section 2.

Our method holds the top entry in the challenge's leader board[3] at the time of submission, see Table 1a. This is especially remarkable, because it is simpler than the methods holding the other top entries. Similar to us, they rely on a CNN to predict boundary locations, but postprocess its output with the complex pipeline described in [24], that involves a NP-hard partitioning step.

In addition, we compare to baseline post-processing methods starting from our network predictions: thresholding (THRESH), two watershed variants (WS, WSDT), and one multicut variant (MC-LOCAL) only take into account short-range predictions. Lifed multicut (LMC) and another multicut variant (MC-FULL) also use long-range predictions. For these baseline methods we have only produced 2D segmentations for the individual slices, either because the 3D results were inferior (THRESH, WS, WSDT) or infeasible to obtain (MC, LMC). In contrast, the Mutex Watershed benefited from 3D segmentation. See table 1b for the evaluation results and see Supplementary 2 for further details on the baseline methods and a qualitative comparison.

The three methods that use short- and long-range connectivty perform significantly better than the other methods. Somewhat surprisingly, MWS performs better than MC-FULL and LMC, which are based on a NP-hard partition problem. This might be explained by the lack of 3D information in the two latter two approaches (solving the 3D model was infeasible).

4.3 Study on natural image segmentation

We conducted preliminary experiments on the Berkeley segmentation dataset BSD500 [53] to study the Mutex Watersheds applicability to natural images.

[3] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new

Method	Rand-Score	VI-Score
UNet + MWS	**0.98792**	**0.99183**
M2FCN + LMC [51]	0.98788	0.99072
SCN + LMC [52]	0.98680	0.99144
FusionNet + LMC [30]	0.98365	0.99130
ICv1 + LMC [24]	0.98262	0.98945

(a) Top five entries at time of submission. Our Mutex Watershed (MWS) is state-of-the-art without relying on complex lifted multicut postprocessing used by all other top entries.

Method	Rand-Score	VI-Score	Time [s]
MWS	**0.98792**	**0.99183**	43.32
MC-FULL	0.98029	0.99044	9415.8
LMC	0.97990	0.99007	966.0
THRESH	0.91435	0.96961	0.2
WSDT	0.88336	0.96312	4.4
MC-LOCAL	0.70990	0.86874	1410.7
WS	0.63958	0.89237	4.9

(b) Comparison to other segmentation strategies, all of which are based on our CNN.

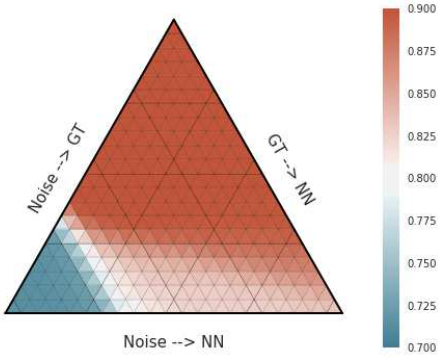
Table 1: Results on the ISBI 2012 EM Segmentation Challenge.

Training a state-of-the-art edge detection network on this small dataset requires a set of dataset specific optimization tricks such as training with external data, multi resolution architectures and auxiliary losses [43]. In this preliminary study we train a 2D version of the network used for the ISBI experiments to predict the 2D connectivity pattern depicted in Figure 4a. To alleviate the small size of the training set, we present this network with predictions from [42] as additional input channel.

In order to isolate the influence of the quality of the underlying affinities, we run ablation experiments where we interpolate (via weighted average) between (a) affinities as predicted by our neural network, (b) those obtained from the ground-truth and (c) uniform noise. We obtain Mutex Watershed segmentations from the interpolated affinities for the BSD testset, size-filter them (as the only post-processing step) and evaluate with the Rand Index. The "phase transition diagram" resulting from these experiments is shown in Figure 6a; Table 6b shows Rand Index and Variation of Information obtained for several points on this diagram.

Observe that the vertices corresponding to (a) and (c) can be interpreted as structured and unstructured noise on the ground-truth affinities (respectively). Hence, the results of our experiments show that the Mutex Watershed is fairly robust against both types of noise; when mixing the GT with noise, the quality of the segmentations is unaffected up to 60 % noise. When mixing GT with NN predictions, it is unaffected to an even higher degree.

In addition, we compare to the result of [22], who use an approach similar to ours and solve a Lifted Multicut based on long range potentials extracted from a pre-computed probability map. In Supplementary 3, we show the segmentations resulting at different stages of interpolation between GT, NN predictions and noise.



(a) BSD500 segmentation quality of MWS algorithm, given affinities from ground truth (top corner), from a neural network (right corner) or pure noise (left corner); plus hundreds of experiments on weighted combinations of the above. MWS segmentation quality (evaluated with Rand index) degrades only once a large amount of noise is added to the affinities.

NN	GT	Noise	RI	VI
100%	0%	0%	0.826	1.722
0%	100%	0%	0.901	0.927
0%	38%	62%	0.897	0.976
0%	33%	66%	0.820	1.912
80%	20%	0%	0.878	1.247
43%	0%	57%	0.813	2.127
43%	14%	43%	0.838	1.636
Keuper et al. [22]			0.82	1.75

(b) BSD500 scores at various interpolations between the neural network predictions (NN), ground-truth (GT) and noise. See Supplementary Section 3 for example images of the interpolated affinities. We include [22] as a reference point, because they also use long range potentials in their segmentation method.

5 Conclusion

We have presented a fast algorithm for the clustering of graphs with both attractive and repulsive edges. The ability to consider both obviates the need for the kind of stopping criterion or even seeds that all popular algorithms except for correlation clustering need. The proposed method has low computational complexity in imitation of its close relative, Kruskal's algorithm.

Finally, we have found that the proposed algorithm, when presented with informative edge costs from a good neural network, outperforms all known methods on a competitive bioimage partitioning benchmark, including methods that operate on the very same network predictions.

In future work we want to generalize our algorithm to semantic instance segmentation commonly found in natural image segmentation challenges [54,55,56].

6 Acknowledgements

The authors acknowledge partial support by DFG HA 4364/8-1 and DFG SFB 1129.

References

1. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Trans. Pattern Analysis Machine Intelligence (6) (1991) 583–598
2. Beucher, S., Meyer, F.: The morphological approach to segmentation: the watershed transformation. Optical Engineering **34** (1992) 433–433
3. Grimaud, M.: New measure of contrast: the dynamics. In P. D. Gader, E. R. Dougherty, & J. C. Serra, ed.: Proc. Image Algebra and Morphological Processing. Volume 1769 of SPIE Conf. Series. (1992) 292–305
4. Beucher, S.: Watershed, hierarchical segmentation and waterfall algorithm. In: Proc. ISMM'94. Volume 94. (1994) 69–76
5. Vachier, C., Meyer, F.: Extinction value: a new measurement of persistence. In: Worksh. Nonlinear Signal and Image Processing. Volume 1. (1995) 254–257
6. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence **18**(12) (1996) 1163–1173
7. Soille, P.: Constrained connectivity for hierarchical image decomposition and simplification. IEEE Trans. Patt. Anal. Mach. Intell. **30**(7) (2008) 1132–1145
8. Perret, B., Cousty, J., Guimaraes, S.J., Maia, D.S.: Evaluation of hierarchical watersheds. (2017) HAL preprint 01430865.
9. Meyer, F.: Morphological multiscale and interactive segmentation. In: WS on Nonlinear Signal and Image Processing. (1999) 369–377
10. Najman, L.: On the equivalence between hierarchical segmentations and ultrametric watersheds. J. of Mathematical Imaging and Vision **40**(3) (2011) 231–247
11. Salembier, P., Garrido, L.: Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. IEEE Trans. Image Proc. **9** (2000) 561–576
12. Malmberg, F., Strand, R., Nyström, I.: Generalized hard constraints for graph segmentation. In: Scandinavian Conference on Image Analysis, Springer (2011) 36–47
13. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. Int. J. Comput. Vision **59**(2) (2004) 167–181
14. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Patt. Anal. Mach. Intell. **33**(5) (2011) 898–916
15. Kiran, B.R., Serra, J.: Globallocal optimizations by hierarchical cuts and climbing energies. Pattern Recognition **47**(1) (2014) 12–24
16. Andres, B., Kappes, J.H., Beier, T., Köthe, U., Hamprecht, F.A.: Probabilistic image segmentation with closedness constraints. In: Proc. ICCV'11. (2011) 2611 – 2618 1.
17. Andres, B., Kröger, T., Briggmann, K.L., Denk, W., Norogod, N., Knott, G., Köthe, U., Hamprecht, F.A.: Globally optimal closed-surface segmentation for connectomics. In: Proc. ECCV'12, part 2. Number 7574 (2012) 778–791
18. Yarkony, J., Ihler, A., Fowlkes, C.C.: Fast planar correlation clustering for image segmentation. In: Proc. ECCV'12. (2012) 568–581
19. Pape, C., Beier, T., Li, P., Jain, V., Bock, D.D., Kreshuk, A.: Solving large multicut problems for connectomics via domain decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 1–10
20. Zhang, C., Yarkony, J., Hamprecht, F.A.: Cell detection and segmentation using correlation clustering. In: Proc. MICCAI'14. (2014) 9–16

21. Horňáková, A., Lange, J.H., Andres, B.: Analysis and optimization of graph decompositions by lifted multicuts. In: International Conference on Machine Learning. (2017) 1539–1548
22. Keuper, M., Levinkov, E., Bonneel, N., Lavoué, G., Brox, T., Andres, B.: Efficient decomposition of image and mesh graphs by lifted multicuts. In: Proc. ICCV'15. (2015) 1751–1759
23. Beier, T., Andres, B., Köthe, U., Hamprecht, F.A.: An efficient fusion move algorithm for the minimum cost lifted multicut problem. In: European Conference on Computer Vision, Springer (2016) 715–730
24. Beier, T., Pape, C., Rahaman, N., Prange, T.e.a.: Multicut brings automated neurite segmentation closer to human performance. Nature Methods **14**(2) (2017) 101–102
25. Lee, K., Zung, J., Li, P., Jain, V., Seung, H.S.: Superhuman accuracy on the snemi3d connectomics challenge. arXiv preprint arXiv:1706.00120 (2017)
26. Jain, V., Murray, J.F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K.L., Helmstaedter, M.N., Denk, W., Seung, H.S.: Supervised learning of image restoration with convolutional networks. Proc. ICCV'07 (2007) 1–8
27. Ciresan, D.C., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. Proc. NIPS'12 (2012)
28. Arganda-Carreras, I., Turaga, S., Berger, D., et al.: Crowdsourcing the creation of image segmentation algorithms for connectomics. Front. Neuroanatomy **9** (2015) 142
29. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. Proc. MICCAI'15 (2015) 234–241
30. Quan, T.M., Hilderbrand, D.G., Jeong, W.K.: FusionNet: a deep fully residual convolutional neural network for image segmentation in connectomics. arXiv:1612.05360 (2016)
31. Nunez-Iglesias, J., Kennedy, R., Parag, T., Shi, J., Chklovskii, D.: Machine learning of hierarchical clustering to segment 2D and 3D images. PLoS one **8** (2013) e71715
32. Knowles-Barley, S., Kaynig, V., Jones, T.R., Wilson, A., Morgan, J., Lee, D., Berger, D., Kasthuri, N., Lichtman, J.W., Pfister, H.: RhoanaNet pipeline: Dense automatic neural annotation. arXiv:1611.06973 (2016)
33. Uzunbaş, M.G., Chen, C., Metaxas, D.: Optree: a learning-based adaptive watershed algorithm for neuron segmentation. In: Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'14). (2014) 97–105
34. Meirovitch, Y., Matveev, A., Saribekyan, H., Budden, D., Rolnick, D., Odor, G., Jones, S.K.B.T.R., Pfister, H., Lichtman, J.W., Shavit, N.: A multi-pass approach to large-scale connectomics. arXiv preprint:1612.02120 (2016)
35. Januszewski, M., Maitin-Shepard, J., Li, P., Kornfeld, J., Denk, W., Jain, V.: Flood-filling networks. arXiv:1611.00421 (2016)
36. Zlateski, A., Seung, H.S.: Image segmentation by size-dependent single linkage clustering of a watershed basin graph. arXiv:1505.00249 (2015)
37. Parag, T., Tschopp, F., Grisaitis, W., Turaga, S.C., Zhang, X., Matejek, B., Kamentsky, L., Lichtman, J.W., Pfister, H.: Anisotropic em segmentation by 3d affinity learning and agglomeration. arXiv preprint 1707.08935 (2017)
38. Turaga, S.C., Murray, J.F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., Seung, H.S.: Convolutional networks can learn to generate affinity graphs for image segmentation. Neural Computation **22**(2) (2010) 511–538
39. Turaga, S.C., Briggman, K.L., Helmstaedter, M., Denk, W., Seung, H.S.: Maximin affinity learning of image segmentation. arXiv:0911.5372 (2009)

40. Wolf, S., Schott, L., Köthe, U., Hamprecht, F.: Learned watershed: End-to-end learning of seeded segmentation. Proc. ICCV'17 (2017)
41. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. arXiv:1611.08303 (2016)
42. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proc. ICCV'15. (2015) 1395–1403
43. Kokkinos, I.: Pushing the boundaries of boundary detection using deep learning. arXiv:1511.07386 (2015)
44. Cai, J., Lu, L., Zhang, Z., Xing, F., Yang, L., Yin, Q.: Pancreas segmentation in MRI using graph-based decision fusion on convolutional neural networks. In: Proc. MICCAI. (2016)
45. Meyer, F.: Topographic distance and watershed lines. Signal processing **38**(1) (1994) 113–125
46. Meyer, F.: Minimum spanning forests for morphological segmentation. In: Mathematical morphology and its applications to image processing. (1994) 77–84
47. Falcão, A.X., Stolfi, J., de Alencar Lotufo, R.: The image foresting transform: Theory, algorithms, and applications. IEEE Trans. Patt. Anal. Mach. Intell. **26**(1) (2004) 19–29
48. Cormen, T.H.: Introduction to algorithms. MIT press (2009)
49. Schlegel, P., Costa, M., Jefferis, G.S.: Learning from connectomics on the fly. Current opinion in insect science (2017)
50. Funke, J., Tschopp, F.D., Grisaitis, W., Sheridan, A., Singh, C., Saalfeld, S., Turaga, S.C.: Large scale image segmentation with structured loss based deep learning for connectome reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
51. Shen, W., Wang, B., Jiang, Y., Wang, Y., Yuille, A.: Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection. arXiv preprint arXiv:1703.08493 (2017)
52. Weiler, M., Hamprecht, F.A., Storath, M.: Learning steerable filters for rotation equivariant cnns. arXiv preprint arXiv:1711.07289 (2017)
53. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision. Volume 2. (July 2001) 416–423
54. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
55. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755
56. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014)