Optree: a Learning-Based Adaptive Watershed Algorithm for Neuron Segmentation

Mustafa Gökhan Uzunbaş*, Chao Chen, and Dimitris Metaxsas

CBIM, Rutgers University, Piscataway, NJ, USA

Abstract. We present a new algorithm for automatic and interactive segmentation of neuron structures from electron microscopy (EM) images. Our method selects a collection of nodes from the watershed merging tree as the proposed segmentation. This is achieved by building a conditional random field (CRF) whose underlying graph is the merging tree. The maximum a posteriori (MAP) prediction of the CRF is the output segmentation. Our algorithm outperforms state-of-the-art methods. Both the inference and the training are very efficient as the graph is tree-structured. Furthermore, we develop an interactive segmentation framework which selects uncertain regions for a user to proofread. The uncertainty is measured by the marginals of the graphical model. Based on user corrections, our framework modifies the merging tree and thus improves the segmentation globally.

Keywords: Conditional Random Field, Watershed, EM Segmentation, User Interaction

1 Introduction

The *watershed transform* [9] partitions a given image into *segments* by simulating a water flooding of the landscape of a given scalar function, e.g. the gradient magnitude or the likelihood of each pixel being the boundary (Fig. 1(c)). In order to mitigate the over-segmentation effect, one often merges neighboring segments when the minimal function value along the boundary between them (called the *saliency*) is below certain threshold. Considering all saliency thresholds, a hierarchical merging tree is constructed [9] in which each leaf node is a segment of the original watershed and each non-leaf node is a merged segment. A *height* function can be assigned to each node according to the minimal saliency threshold at which it disappears (is merged with others). The watershed segmentation at any given threshold can be computed by cutting all tree nodes below the threshold and taking all leaf nodes of the remaining tree. In Fig. 1(d) and 1(e), we show the original watershed result and a thresholded one.

The watershed method and its variations have been used on EM images [4,2,8]. However, running watershed using a certain threshold usually leads to accurate segments at certain area yet over/under-segmentation at other areas (see Fig. 1(d) and 1(e)). In this paper, we propose a CRF-based learning algorithm which finds a segmentation of higher quality by selecting different saliency

* This grant was partially supported based on funding from the following grants NIH-R01-HL086578, NIH-R21-HL088354 and NSF-MRI-1229628.

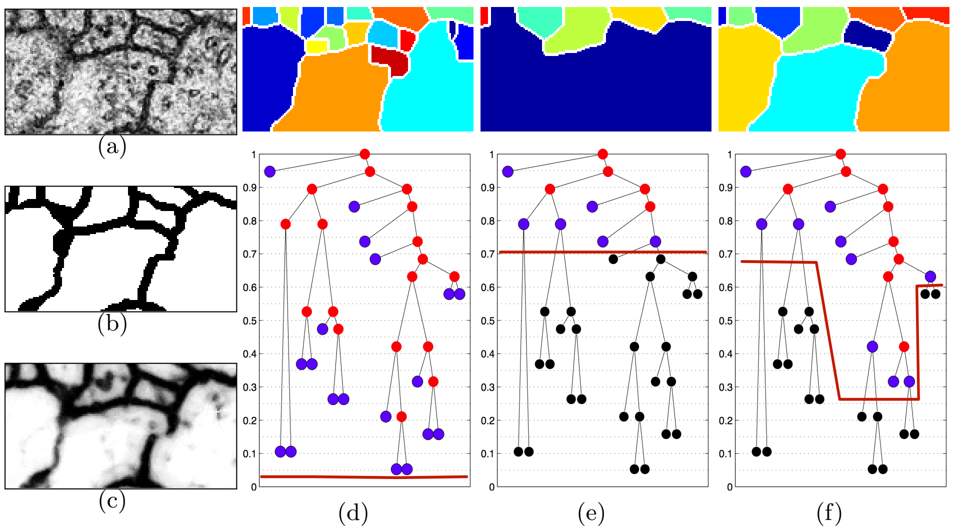


Fig. 1. (a) The EM image patch; (b) the ground truth; (c) the boundary likelihood map (dark pixels have high values); (d) the watershed segmentation and its tree, built using the boundary likelihood map as the landscape function; (e) the watershed segmentation with a higher threshold; (f) the result of our algorithm.

thresholds at different areas of the image. Essentially, our algorithm learns from training data how to cut a hierarchical tree adaptively to achieve a better result. See Figure 1(f) for an example of our result. We train our algorithm to construct a tree-structured graphical model for each image and its hierarchical tree. The (MAP) probability of this graphical model gives a segmentation. Our method outperforms state-of-the-art in automatic segmentation of high resolution 2D (ssTEM) [3] and 3D (FIBSEM) [7] EM images.

Our CRF model leads to a novel interactive segmentation framework. Connectomics requires extremely accurate partitioning of EM images into distinct neuron cells. In order to achieve a satisfying quality, human experts have to *proofread*, namely, to manually correct the segmentation results [12,4]. It is a difficult and tedious task due to the huge data size (500^3 voxels and more than 1000 cells). Fig. 4(a) shows what a human expert is facing in proofreading. Our interactive interface only highlights a small set of locations for human experts to verify, namely, the locations at which our graphical model has very low confidence. When a user fixes a mistake at one of these locations, our framework modifies the merging tree accordingly and recomputes the segmentation. The improvement to the segmentation is global. Our experiments show that under the new framework, within fifteen user inputs, the segmentation is improved to the optimal quality, much faster than classical user interaction frameworks.

Related work. The closest works to us are [2,8], which also start with an initial watershed segmentation. The problem is formulated as a labeling problem of boundaries under certain topological constraints. The problem is solved using multicut integer linear programming. Another recent work [10] which also uses an initial watershed segmentation and performs agglomeration, learns how to merge watershed segments via active learning methods.

Kiran *et al.* [11] compute cuts of the hierarchical tree optimizing certain predefined energy. However, their energy is not learned in a supervised fashion as we did. Turaga *et al.* [13] learn to construct a hierarchical tree so that the watershed cut at a certain threshold produces a high quality segmentation. There are many other learning based methods for Connectomics segmentation task. See [6] and references therein.

2 Background

Suppose we are given a graph, e.g. a hierarchical merging tree. Denote by $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of nodes and the set of edges respectively. Each node can take a label from a label set $\mathcal{L}$. We call each label configuration of all nodes a *labeling*. Denote by $\mathcal{Y} = \mathcal{L}^{\mathcal{V}}$ the space of all labelings. Given an observation x and a parameter vector w, the conditional probability of a labeling y is $P(y|x,w) = \exp(-E(y|x,w) - \log Z(x,w))$, where $Z(x,w) = \sum_{y \in \mathcal{Y}} \exp(-E(y|x,w))$ is the *partition function*. The energy is often defined as the negative inner product $E(y|x,w) = -\langle w, \phi(x,y) \rangle$, in which $\phi(x,y)$ is the concatenation of features of all nodes and edges. Note that the feature vector also depends on y.

In prediction, one computes the *maximum a posteriori (MAP)*,

$$\operatorname{argmax}_y P(y|x,w) = \operatorname{argmin}_y E(y|x,w). \tag{1}$$

It is also very useful to compute the *marginals*, $P(y_i = \ell|x,w) = \sum_{y:y_i=\ell} P(y|x,w)$. Since our graph is tree-structured, computing MAP and marginals can be solved exactly and efficiently using dynamic programming algorithms. For more general graphs, the inference tasks are NP-hard and have to be solved approximately, e.g. using loopy belief propagation.

We train the parameter vector w from a given set of training data $\{(x^n, y^n)\}$, $n = 1, \ldots, N$. This can be achieved by minimizing the convex loss function $\text{loss}(w) = \|w\|^2 + c \sum_{n=1}^{N} [\log Z(x^n, w) - \langle w, \phi(x^n, y^n) \rangle]$. We can find the optimal w efficiently using gradient descent. The gradient can be computed efficiently as long as the marginals can be computed efficiently. We use the UGM package by Mark Schmidt for the training [1].

Factorization and labeling constraints. Assuming the Markov properties, the energy E can be factorized into the summation of unary potentials and pairwise potentials, $E(y|x,w) = \sum_{(i,i') \in \mathcal{E}} E_{i,i'}(y_i, y_{i'}|x,w) + \sum_{i \in \mathcal{V}} E_i(y_i|x,w)$. For a node/edge, one can forbid a certain label/label combination by forcing the corresponding potential to be infinite. Thus we can compute MAP and marginals only over the set of feasible labelings $\mathcal{Y}' \subseteq \mathcal{Y}$. This makes it possible to do training and inference only over $\mathcal{Y}'$, as we will do in this paper.

3 Computing Tree-Derived Segmentation

For any given image, we construct a watershed merging tree by running on a boundary likelihood map, namely, the likelihood of whether a pixel belonging to the boundaries between neuron cells (Figure 1(c)). Next, we construct a tree-structured graphical model whose underlying graph is the same as the

[1] `http://www.di.ens.fr/~mschmidt/Software/UGM.html`

hierarchical tree. The problem of computing the optimal segmentation is transformed into the problem of computing the optimal labeling of this graphical model (Equation (1)). In order to achieve the transformation, we need to build a correspondence between segmentations and labelings.

The correspondence between segmentations and labelings. Denote by $\mathcal{P} = \{p_1, \ldots, p_M\}$ the set of all regions corresponding to the leaf nodes of the tree. We call these regions the *superpixels*. A *segmentation* is a decomposition of the set of superpixels into a disjoint set of *segments*. Formally, a segmentation $S = \{s_1, \ldots, s_m \mid s_i \subseteq \mathcal{P}, \forall i; \dot{\bigcup}_{i=1}^{m} s_i = \mathcal{P}\}$.

Denote by $\mathcal{S}$ the space of all possible segmentations. For a given image I, we would like to find the most probable segmentation $\mathrm{argmax}_{S \in \mathcal{S}} P(S|I)$, with a suitably defined posterior probability $P(S|I)$. Since $\mathcal{S}$ is too large to search through, we restrict the solution space to a smaller subset. Given a hierarchical tree, T, a segmentation S is *T-derived* if and only if each segment $s_i \in S$ is a node of T. We call such segmentations *tree-derived* and our algorithm search through the space of all T-derived segmentations, denoted by $\mathcal{S}_T \subseteq \mathcal{S}$. The correspondence between tree-derived segmentations and the labelings of T is provided as follows. Let the label set $\mathcal{L} = \{-1, 0, 1\}$, represent whether each tree node is an under-segment, a segment or an over-segment. An under-segment node is the ascendant of a set of segment nodes. An over-segment node is the descendant of a segment node. For each labeling y, we take the set of zero-labeled nodes as the corresponding segmentation. In Figure 1(f), the tree represents a labeling. Red, blue and dark colors correspond to -1, 0 and $+1$ labels respectively. The corresponding segmentation is shown in the top row.

We are only interested in labelings that derive legit segmentations. Therefore we enforce certain restrictions on the labelings. For a labeling y and a node v, let $\Gamma_v(y)$ be the sequence of labels along the path from v to the root.

Theorem 1 *There is an one-to-one correspondence between $\mathcal{S}_T$ and the set of labelings $\mathcal{Y}_T$, such that for any leaf node v, (1) $\Gamma_v(y)$ is monotonically non-increasing; (2) The zero label appears exactly once in $\Gamma_v(y)$; (3) The first label (label of v) cannot be* -1 *and the last label (label of the root node) cannot be* $+1$.

We defer the proof to the supplemental material.

We call labelings within $\mathcal{Y}_T$ *segmentation labelings.* Conditions in Theorem 1 can be translated into restrictions on labels of nodes and edges. In particular, a labeling y is a segmentation labeling if and only if (1) the root has label $y_{root} \in \{-1, 0\}$; (2) any leaf node, v, has label $y_v \in \{0, 1\}$; (3) for any child-parent pair, (c, p), $y_c \geq y_p \geq y_c - 1$; (4) if $y_p = y_c$, $y_c \neq 0$.

Prediction and training. For any given image and hierarchical tree, we construct a graphical model. [2] Let the posterior probability of a tree-derived segmentation $S \in \mathcal{S}_T$ be $P(S|I) = P(y^S|I, w)$, where y^S is the corresponding segmentation labeling of S. Recall that we can enforce all aforementioned label constraints by setting certain potentials to infinity. Therefore, we can restrict the set of feasible solutions of the graphical model to be $\mathcal{Y}_T$ and compute the MAP, $\mathrm{argmax}_{y \in \mathcal{Y}_T} P(y|I, w)$. The corresponding segmentation is the predicted

[2] See the supplemental material for details of extracting features $\phi(x, y)$ in Eq. 1.

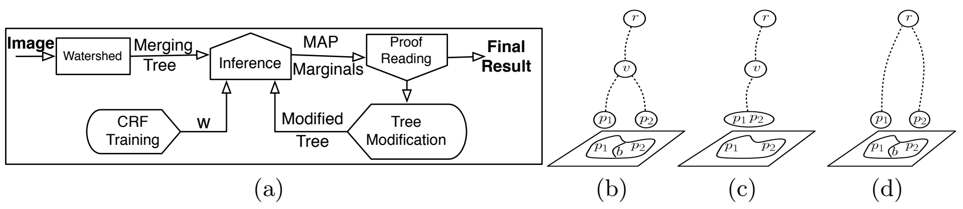


Fig. 2. (a)The flowchart of the interactive framework; (b) a tree (nodes on the curved paths are not shown for clarity); (c) b is labeled merged; (d) b is labeled split.

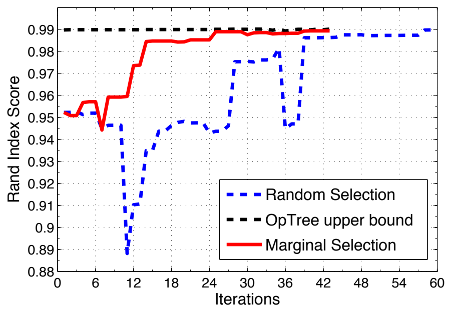
segmentation. At the training stage, for each data, we construct the graphical model similarly so that the marginals are computed within $\mathcal{Y}_T$. It remains to show how to compute the ground truth labeling for each training data.

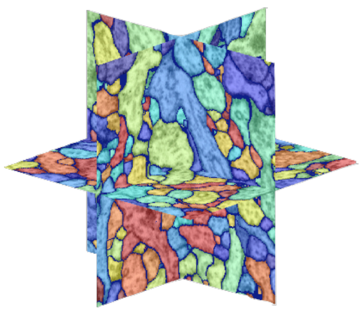
Computing the optimal tree-derived segmentation. For each training data, we are given a ground truth segmentation $\widehat{S}$, which may not be tree-derived. In order to find a ground truth labeling of this data, we find the tree-derived segmentation that best approximates $\widehat{S}$, $S^* = \text{argmax}_{S \in \mathcal{S}_T} \text{score}(S, \widehat{S})$. For EM images, there are two popular score functions, the random index and the variation of information. Notice that both functions can be decomposed into a summation of scores of elements of S, $\text{score}(S) = \sum_{s_i \in S} f(s_i)$. Based on this observation, we provide a polynomial algorithm to compute S^* using standard dynamic programing techniques. See the supplemental material for the pseudocode.

4 Interactive Segmentation

Our interactive algorithm works as follows. For a given test image, we compute an automatic segmentation using the algorithm presented in the previous section. Based on marginals of the graphical model, we suggest the user a few locations to proofread. When a user finds a mistake in the segmentation, he/she clicks and corrects it. The boundaries that have been corrected would not be highlighted during the remaining iterations. We modify the merging tree accordingly and recompute the segmentation on the modified tree. This process is repeated until the user is satisfied. Recall that to construct a graphical model, we need a parameter vector w. Throughout the user interaction, we use the same w which is learned in the training stage. See Figure 2(a) for the flowchart.

Boundaries and their marginals. The basic elements for a user to handle are boundaries between superpixels. Recall superpixels form a bottom layer watershed segmentation. We collect the set of all boundaries (curves for 2D images and surface patches for 3D images). For a given segmentation, we say a boundary is labeled *merged* if the two adjacent superpixels belong to a same segment. Otherwise, it is labeled *split*. In other words, the boundary will appear (not appear) in the segmentation if it is labeled split (merged). The task of segmentation is equivalent to finding an optimal split/merged labeling to all the boundaries. Our CRF generates marginals for nodes of the tree. However, we can easily translate node marginals into marginals of whether each boundary being split or merged. For a given boundary b and its two adjacent superpixels, p_1, p_2, we find the least common ancestor of the leaf nodes containing p_1 and p_2. We call this node, v, the *containing node* of b, because b is in the interior of v and all its ancestors





(a) (b)

Fig. 3. (a) Interaction Simulation; (b) 3D Result.

(Fig. 2(b)). The boundary b is split if and only if v is an under-segment, i.e. has label -1. Therefore, the probability of b being split is equal to the marginal $P(y_v = -1|x, w)$. At each iteration of the segmentation, we show the predicted segmentation and highlight the boundaries that we have low confidence of being either merged or split. This gives a user a small number of options to proofread (see Fig. 4(a)). After the user corrected a boundary, the system will update the tree accordingly and present the low marginal boundaries again.

Modifying trees. We conclude this section by explaining how to update the tree according to user inputs. When a user specifies a boundary between two superpixels to be merged, we merge the paths from the two leaf nodes containing the two superpixels to their common ancestor node into a single path. When user specifies a boundary to be split, we split the path from v to the root into two paths, nodes along the original path are assigned to either of the two new paths, depending on the situation. Also we enforce an extra constraint that the raised containing node could only have label -1. These operations will ensure the boundary is merged/split in any tree-derived segmentation of the modified tree. In Fig. 2, we illustrate the two operations on a same tree.

5 Experiments

2D experiments. We applied our method to neuron EM images from the ISBI 2012 EM image segmentation challenge [3]. The data contains 30 2D sections of ssTEM images, each of which has 512×512 pixels. We used the boundary likelihood map from [5]. We submitted our result on test data to the challenge website and our method achieved the 2nd place overall in the competition, with an adjusted Rand Index Error of 0.023, Warping Error 0.0008 and Pixel Error 0.11. [1]. Note that our group name is "optree-idsia". The training took 167 seconds and the MAP computation during testing per image was 0.05 seconds.

To demonstrate the necessity of all structural constraints that we enforce in our algorithm, we ran a holistic experiment. We compared our method with three baseline approaches: watersheds (WS); node classifier (NC); unconstrained CRF (UNC). WS is the the classical watershed using the best threshold. In NC we trained a random forest classifier to predict node labels. In UNC, we used CRF on trees without constraints defined in Theorem 1. We used post-processing to ensure the final results of these methods are legit segmentations. Note that NC and UNC are both supervised training methods, like ours.

	WS	NC	UNC	Optree	Optimal
ARI	0.05	0.14	0.10	0.023	0.015

In the table, Optree outperforms all baselines in the Adjusted Rand Index (ARI). We also present the optimal tree-derived segmentation result (Optimal) for reference, which is achieved when the ground truth is known. This is the theoretical upper-bound of our tree-derived segmentations. We observe that UNC outperforms NC, justifying the significance of the tree structure in the model.

3D experiments. We applied our method to a $500 \times 500 \times 500$ 3D EM image from [7]. We used the boundary likelihood map from "Ilastik" [12]. To reduce the watershed tree size, we removed all nodes with height less than 0.05. We divided the initial volume into 8 250x250x250 sub-volumes. We run experiments for 8 times. Each time we use one sub-volume for training and the rest for testing. Training took 465 seconds and MAP computation took 0.9 seconds on average. The average Rand Index score (one minus the Rand Index Error) is 0.9837. Our method performs better than state-of-the-art [8,2]. The optimal score of tree-derived segmentations (Optimal) is 0.9923. There is still room for improvement over the optree segmentation result. In the next section, we show the result can be improved to the optimal via user interactions, as explained in Section 4.

Interactive experiments. Our interactive system suggests a few boundaries for users to proofread at each iteration. Users judge by observation whether a boundary is mislabeled and correct it. We run experiments to show how this method could improve the efficiency of proofreading. We simulate a user interaction process on a particular 250^3 sub-volume on which our automatic method has the worst score. We start from the automatic algorithm result and correct mislabeled boundaries iteratively. At each iteration, the simulated user (robot) selects one mislabeled boundary based on certain strategy. The merging tree is modified accordingly. We implemented two strategies (1) always select the mislabeled boundary with the least confidence (least marginal); (2) randomly select a mislabeled boundary. In Figure 3(a), correcting low-marginal boundaries (red curve) clearly improves the results much faster than correcting randomly selected boundaries (blue curve). The former takes about 14 iterations to reach the accuracy > 98.7 while the latter takes 39 iterations to reach the same accuracy. We also show the optimal result (black curve) of tree-derived segmentation as a theoretical upper bound.

We illustrate how one user input can improve the segmentation globally. In Figure 4(a), a user has to verify all colored boundaries. However, in our system, she only needs to pay attention to boundaries with low marginals (yellow and cyan). Boundaries of yellow (resp. cyan) color are labeled split (resp. merged). After a selected boundary (yellow arrow) is corrected, many other boundaries are automatically corrected (Figure 4(b), 4(c) and 4(d)). Our interactive segmentation framework is very efficient. In average, each iteration takes around 1.9 second. This includes modifying the tree and recomputing MAP and marginals.

6 Conclusion

This paper presents a CRF-based algorithm for neuron segmentation. The tree-structured graphical model allows us to compute accurate segmentation of 500^3 dataset within a second. Furthermore, we develop an interactive segmentation

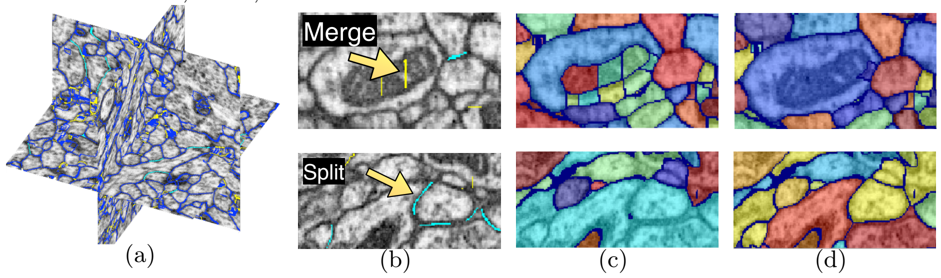


Fig. 4. (a): boundaries a user has to proofread; (b): our system only shows boundaries with low marginals (yellow if labeled split, cyan if labeled merged); (c): before user input; (b): after user input, many boundaries are fixed.

framework that takes advantage of the marginals of the graphical model. The new framework improves the segmentation to the optimal quality within a small number of user inputs.

References

1. ISBI Challenge, `http://brainiac2.mit.edu/isbi_challenge/leaders-board`
2. Andres, B., Kroeger, T., Briggman, K.L., Denk, W., Korogod, N., Knott, G., Koethe, U., Hamprecht, F.A.: Globally optimal closed-surface segmentation for connectomics. In: Computer Vision–ECCV 2012, pp. 778–791. Springer (2012)
3. Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., Hartenstein, V.: An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. PLoS biology 8(10) (2010)
4. Chklovskii, D.B., Vitaladevuni, S., Scheffer, L.K.: Semi-automated reconstruction of neural circuits using electron microscopy. Current opinion in neurobiology 20(5), 667–675 (2010)
5. Ciresan, D., Giusti, A., Schmidhuber, J., et al.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in Neural Information Processing Systems 25. pp. 2852–2860 (2012)
6. Jain, V., Seung, H.S., Turaga, S.C.: Machines that learn to segment images: a crucial technology for connectomics. Current Opinion in Neurobiology 20(5), 653 – 666 (2010)
7. Knott, G., Marchman, H., Wall, D., Lich, B.: Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. The Journal of Neuroscience 28(12), 2959–2964 (2008)
8. Kroeger, T., Mikula, S., Denk, W., Koethe, U., Hamprecht, F.: Learning to segment neurons with non-local quality measures. In: Medical Image Computing and Computer-Assisted Intervention MICCAI 2013, pp. 419–427. Springer (2013)
9. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. PAMI 18(12), 1163–1173 (1996)
10. Nunez-Iglesias, J., Kennedy, R., Parag, T., Shi, J., Chklovskii, D.B.: Machine learning of hierarchical clustering to segment 2d and 3d images. PLoS ONE 8 (2013)
11. Ravi Kiran, B., Serra, J.: Global–local optimizations by hierarchical cuts and climbing energies. Pattern Recognition 47(1), 12–24 (2014)
12. Sommer, C., Straehle, C., Koethe, U., Hamprecht, F.A.: "ilastik: Interactive learning and segmentation toolkit". In: 8th IEEE Int. Symposium(ISBI) (2011)
13. Turaga, S.C., Briggman, K.L., Helmstaedter, M., Denk, W., Seung, H.S.: Maximin affinity learning of image segmentation. In: NIPS. pp. 1865–1873 (2009)