Learning to Segment Neurons
with Non-local Quality Measures

Thorben Kroeger[1], Shawn Mikula[2], Winfried Denk[2],
Ullrich Koethe[1], and Fred A. Hamprecht[1]

[1] HCI, University of Heidelberg
[2] MPI for Medical Research, Heidelberg

Abstract. Segmentation schemes such as hierarchical region merging or correlation clustering rely on edge weights between adjacent (super-)voxels. The quality of these edge weights directly affects the quality of the resulting segmentations. *Unstructured* learning methods seek to minimize the classification error on *individual* edges. This ignores that a few local mistakes (tiny boundary gaps) can cause catastrophic global segmentation errors. Boundary evidence learning should therefore optimize *structured* quality criteria such as Rand Error or Variation of Information. We present the first structured learning scheme using a structured loss function; and we introduce a new hierarchical scheme that allows to approximately solve the NP hard prediction problem even for huge volume images. The value of these contributions is demonstrated on two challenging neural circuit reconstruction problems in serial sectioning electron microscopic images with billions of voxels. Our contributions lead to a partitioning quality that improves over the current state of the art.

1 Introduction

Connectomics requires extremely accurate circuit reconstruction because minor local mistakes can lead to catastrophic global connectivity errors. Automatic methods that achieve the required accuracy level and scale to huge datasets are still an open problem. When segmentation is based on electron microscopy volume images, one must exclusively rely on boundary evidence, because the desired regions (neurons) cannot be differentiated on the basis of appearance features.

For such partitioning problems, correlation clustering [3,4], or multicut segmentation [5], is a powerful paradigm [6,7,8,9,10,11,1]. An image is represented as a weighted region adjacency graph of (super)-voxels. Positive edge weights indicate that the incident regions should be merged; negative weights that they should be kept separate. An optimal segmentation makes binary decisions for each edge so as to minimize the total cut weight, subject to the constraint of producing a topologically consistent solution [5]. The quality of the resulting segmentation depends critically on the edge weights, which are some function of features computed from the raw data. We focus on learning such weights using a cutting-planes approach. In each iteration, a structured loss is used to compare the segmentations obtained from the current weights to the gold standard.

Ideally, [12,13], the loss function takes the entire segmentation into account (Fig. 1). Unfortunately, such loss functions do not decompose over the binary decisions for individual edges, prohibiting efficient inference. This is why previous work has resorted

a) raw data b) gold standard c) small HE, large VI, RE d) large HE, small VI, RE
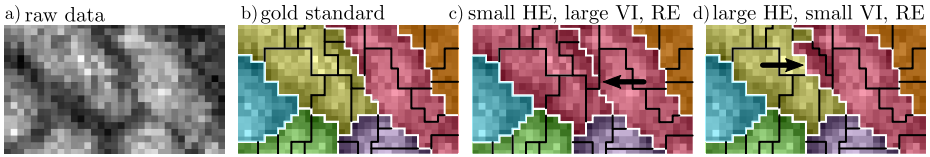
Fig. 1. Segmentation quality is commonly [1,2] measured by Rand Error and Variation of Information which capture (in contrast to the Hamming error HE) not only local segmentation quality, but also some global structural correctness. *(b)* Gold standard for supervoxel segmentation: white edges are correct, black edges incorrect. *(c)* A single missed edge (arrow) has the catastrophic global consequence of merging the two adjacent regions (resulting in the red segment). *(d)* The slightly displaced boundary (arrow) produces a large HE, but small RE or VI.
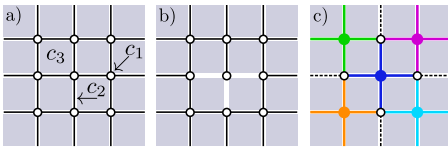
Fig. 2. *(a)* 2D slice through a 3D cell complex representation. *(b)* "dangling" surfaces can be detected locally. *(c)* More efficient enumeration uses the colored lines and their bounded surfaces

to merely counting the number of deviating edge decisions between gold standard and current prediction [7]. Fig. 4 shows that better results can be achieved when a *structured* loss function is used during training. Various attempts have been made to train struct-SVMs with more complex loss functions, but these approaches are tailored to specific applications [14], or use approximation techniques that do not apply to RE and VI [15].

Our first contribution (Sec. 2) is to allow arbitrary structured loss functions, such as Rand Error (RE) or Variation of Information (VI) during struct-SVM training on moderately-sized neighborhoods. This is made possible by a non-redundant and efficient exhaustive enumeration of segmentations. Our second contribution is a hierarchical, blockwise scheme for the structured prediction on large volumes. It produces segmentations that are empirically close to optimal (Sec. 3). Experiments on two different electron microscopy volume images of neural tissue show that the proposed method can improve upon unstructured learning using SVM or random forest classifiers (Sec. 4).

2   Structured Learning for Segmentation

Following [8], we work in a dual representation which specifies a segmentation in terms of binary labels y pertaining to boundaries between supervoxels. However, not every candidate configuration $y \in \{0, 1\}^{|C_2|}$ (where $|C_2|$ is the number of boundaries) represents a valid segmentation: if $y_i = 1$ (boundary is correct) but the adjacent supervoxels belong to the same region (i.e. there exists a path between these supervoxels along which all y_k are labeled $y_k = 0$), y is *inconsistent*, (Figs. 1c and 2b). All consistent y form the set of *multicuts* MC [5]. Region labels are easily determined by connected components. The alternative approach to assign region labels directly (primal representation) leads to a much larger search space, see Tab. 1.

We employ a structured risk minimization formulation in order to learn suitable edge weights from training data. Structural risk minimization aims at finding a regularized predictor that minimizes the empirical loss [16]. Training samples $\boldsymbol{x}^n$ are connected components of surfaces with the gold standard labeling $\boldsymbol{y}^n$. Sample $\boldsymbol{x}$ and any labeling $\boldsymbol{y}$ are described by a joint feature vector $\phi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^M$. The optimal multicut segmentation $\boldsymbol{y}^*$ according to a structured SVM model [16] is then given by

$$\boldsymbol{y}^* = \operatorname*{argmin}_{\boldsymbol{y} \in \mathrm{MC}} \langle \boldsymbol{w}^*, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle \, ; \qquad \boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^D} \left\{ \frac{1}{2} \|\boldsymbol{w}\|^2 + \frac{\lambda}{N} \, l(\boldsymbol{w}) \right\}, \qquad (1)$$

$$l(\boldsymbol{w}) = \sum_{n=1}^{N} \max_{\boldsymbol{y} \in \mathrm{MC}} \left\{ \Delta(\boldsymbol{y}^n, \boldsymbol{y}) - \boldsymbol{w}^T \phi(\boldsymbol{x}^n, \boldsymbol{y}^n) + \boldsymbol{w}^T \phi(\boldsymbol{x}^n, \boldsymbol{y}) \right\}. \qquad (2)$$

The model is trained by finding optimal weights $\boldsymbol{w}^*$. $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$ measures the deviation of $\boldsymbol{y}$ from the gold standard (e.g. by using RE or VI). Hyperparameter λ trades off the regularization and data terms. How should $\phi(\boldsymbol{x}, \boldsymbol{y})$ be chosen? The objective is linear in the edge weights $\boldsymbol{\theta}$ subject to exponentially many constraints restricting $\boldsymbol{y}$ to multicuts:

$$\min_{\boldsymbol{y}} \langle \boldsymbol{\theta}, \boldsymbol{y} \rangle \text{ subject to } \boldsymbol{y} \in \mathrm{MC}. \qquad (3)$$

We seek to optimize the edge weights $\boldsymbol{\theta}$ indirectly by choosing optimal feature weights in the ansatz $\theta_i = <\boldsymbol{w}, \boldsymbol{\alpha}^{(i)}>$ where $\boldsymbol{\alpha}^{(i)}$ is a suitable feature vector associated with each edge[1]. This yields after an exchange of summation order

$$\min_{y \in \mathrm{MC}} \sum_{i=1}^{|C_2|} \left\langle \boldsymbol{w}, \boldsymbol{\alpha}^{(i)} \right\rangle y_i = \min_{y \in \mathrm{MC}} \sum_{i=1}^{|C_2|} \sum_{m=1}^{M} w_m \alpha_m^{(i)} y_i = \min_{y \in \mathrm{MC}} \sum_{m=1}^{M} \boldsymbol{w}_m \sum_{i=1}^{|C_2|} \alpha_m^{(i)} y_i \overset{\text{def}}{=} \min_{y \in \mathrm{MC}} \langle \boldsymbol{w}, \phi \rangle.$$

The joint feature vector $\phi(\boldsymbol{x}, \boldsymbol{y})$ has length M, the number of features for each surface:

$$\phi(\boldsymbol{x}, \boldsymbol{y})^T = \left(\sum_{i=1}^{|C_2|} \alpha_1^{(i)} \cdot y_i, \cdots, \sum_{i=1}^{|C_2|} \alpha_M^{(i)} \cdot y_i \right). \qquad (5)$$

The key operation for determining optimal weights in (1) is the maximization over all feasible segmentations in $l(\boldsymbol{\omega})$, i.e. the identification of the *most violated constraint*. In this paper we investigate how this can be done by *exhaustive search* on as large a subset of the data as possible. We explain our approach to the efficient enumeration of segmentations by means of 2D grids, but the findings likewise apply to 3D supervoxels, which are used in the experiments.

Tab. 1 shows the ratio between the number of true segmentations S and the number of possible configurations for different grid sizes: SP are the number of candidates that would have to be enumerated using the primal representation, SD for the dual representation and SDI for an improved dual enumeration described below. Apparently, S/SDI

---

[1] Note that $\boldsymbol{w}$ is learned from entire configurations, whereas existing methods [8,1,11] learn a probabilistic model $p(y_i|\boldsymbol{\alpha}^{(i)})$ for *individual* edges and then define

$$\theta_i = \log p(y_i = 0 | \boldsymbol{\alpha}^{(i)}) \, / \, \log p(y_i = 1 | \boldsymbol{\alpha}^{(i)}). \qquad (4)$$

Table 1. For a $n \times m$ pixel patch, the ratio between the number of feasible segmentations S and the number of candidate configurations (SP, SD or SDI) depends on the enumeration technique

$n \times m$	$2^{|C_2|}$	S/SP	S/SD	S/SDI
3×3	4'096	0.0219	0.3501	0.6224
4×3	131'072	0.0066	0.2119	0.5024
4×4	16'777'216	0.0016	0.1008	0.4249
5×4	2'147'483'648	0.0004	0.0480	0.2695

achieves the best ratio, i.e. the least work is wasted for solutions that are ultimately rejected. In the dual representation, when exhaustively enumerating all binary vectors $\boldsymbol{y}$, an inconsistent configuration ($\boldsymbol{y} \notin \mathrm{MC}$) can be identified by connected component labeling, which is expensive. Fortunately, many inconsistent configurations (those which contain one or more "dangling" surfaces, Fig. 2b) can be identified more simply.

Efficient and unambiguous enumeration is greatly facilitated by a cell complex data structure [17]. It represents entities of different dimensionality simultaneously along with their bounding relations: supervoxels $c_3 \in C_3$ are bounded by joint-faces between pairs of adjacent supervoxels $c_2 \in C_2$ which in turn are bounded by joint-lines $c_1 \in C_1$ between adjacent faces. Our 2D illustrations should be understood as slices through 3D data: surfaces appear as lines and joint-lines between surfaces appear as points (Fig. 2a). A line c_1 is formed where multiple $c_2 \in \mathrm{bounds}(c_1)$ meet. (green line $\in \mathrm{bounds}$(green dot) in Fig. 2c). We first find a – preferably maximal – set of lines C_1', such that the sets $\mathrm{bounds}(c_1')$, $c_1' \in C_1'$ are mutually disjoint. (Fig. 2c: C_1' consists of all colored dots).

The surfaces $c_2 \in \mathrm{bounds}(c_1')$ must be assigned a consistent labeling in order for $\boldsymbol{y} \in \mathrm{MC}$ to hold: the configurations (1,0,0,0), (0,1,0,0), (0,0,1,0) and (0,0,0,1) are already *locally inconsistent* (a "dangling" surface). Any $\boldsymbol{y}$ in which these configurations occur for one or more of the sets $\mathrm{bounds}(c_1')$, $c_1' \in C_1'$ can be excluded from the enumeration (improved dual enumeration, DI). Using DI, we only need the expensive connected components check on a tiny fraction of the actual number of segmentations. This way we can handle larger subsets.

In a supervoxel segmentation, due to the voxel grid topology, either three or four surfaces meet to form a line c_1. We first find an approximately maximal set C_1'. Then *candidate* configurations $\boldsymbol{y}$ that are locally consistent are enumerated. Each locally consistent candidate is checked if it is globally consistent by connected component labeling. In our C++ implementation, each segmentation with $|C_2| < 32$ is stored efficiently as a 4-byte integer. The enumeration via C_1' is implemented via fast bitwise operations.

3   Structured Prediction

As (3) is NP-hard [3], in practice a solution cannot be found if the weights make for a "difficult" problem (Fig. 4 left) or the number of variables is too large. Given weights $\boldsymbol{\theta}$ obtained with structured learning, the global optimum of the multicut objective is found using the integer linear programming approach of [1]. In our experiments, only problems with about 10^5 variables could be optimized in reasonable time, while we would like to run structured prediction on problems which are several orders of magnitude larger. We therefore propose a hierarchical blockwise optimization scheme. Given an oversegmentation $\mathcal{C}$ and weights $\boldsymbol{\theta}$ we divide the problem into *subproblems* via blocks.
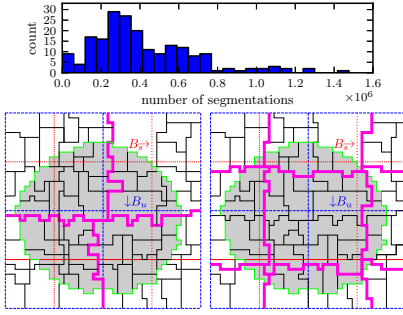
Fig. 3. *Above:* Histogram of the number of (unique) segmentations for the 200 training samples from the mouse dataset. These sizes are still sufficiently small for storage and exhaustive search. *Below:* Dividing a supervoxel segmentation into blocks (*left*: B_u, *right*: B_s). Dashed blue lines indicate the unshifted blocking B_u of the pixels, dashed red lines the shifted blocking B_s. The set ∂C_2 of surfaces separating adjacent blocks is shown with bold magenta lines.

$\mathcal{B}_u$ (*unshifted blocking*) is a partitioning of the volume into rectangular blocks with shape $\boldsymbol{L} = (L_1, L_2, L_3)$. $\mathcal{B}_s$ is a blocking which is *shifted* by $\boldsymbol{L}/2$.

For the chosen blocking $\mathcal{B}$, each supervoxel is uniquely assigned to the block of smallest scan-order index $b \in \mathcal{B}$ with which it intersects (creating a supervoxelized blocking of the volume, Fig. 3 right). Let $C_3(b)$ be the set of supervoxels assigned to block b and $C_2(b)$ the set of all surfaces which bound at least one of these supervoxels. $\partial C_2(b) \subset C_2(b)$ forms b's surface.

For hierarchy level 1, an initial block shape $\boldsymbol{L}$ is chosen. We start with an unshifted blocking B_u. For each block b, the optimization problem (3) is solved, subject to the additional constraints that surfaces $c_2 \notin C_2(b)$ are assigned zero and $c_2 \in \partial C_2(b)$ are assigned one. Effectively, this reduces the problem size to $|C_2(b)|$ variables. Results from all blocks are combined via binary OR to yield a result $\boldsymbol{y}(B_u)$. As each subproblem yields a consistent solution, and all surfaces separating the blocks have $y_i = 1$, the entire state $\boldsymbol{y}(B_u)$ is consistent. The procedure is repeated with a *shifted blocking* B_s, yielding $\boldsymbol{y}(B_s)$. A vector $\boldsymbol{y}^{(1)}$ for the first hierarchy level is obtained by binary OR of $\boldsymbol{y}(B_u)$ and $\boldsymbol{y}(B_s)$. As an intersection of two segmentations, $\boldsymbol{y}^{(1)} \in$ MC. Combining B_u and B_s considerably reduces boundary artifacts, see Fig. 4, right. All variables $y_i = 0$ are removed from the problem. We then obtain a new cell complex $\mathcal{C}'$ (hierarchy level 2) by a connected component labeling of the 1,2, and 3-cells in $\mathcal{C}$, and a bijection $M(\mathcal{C}) \to \mathcal{C}'$ mapping between the entities of level 1 and 2. $\mathcal{C}'$ consists of fewer, but bigger lines, surfaces and segments. New weights $\theta(c')$ for $c' \in \mathcal{C}'$ are computed by
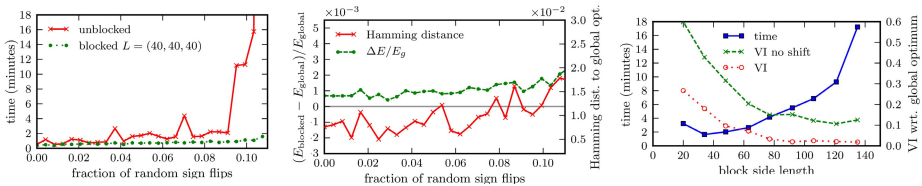
Fig. 4. *Left:* Increasing sign noise on the weights $\boldsymbol{\theta}$ simulates "difficult" weights. With increasing noise, the runtime for obtaining a globally optimal solution explodes. However, if the hierarchical blocked algorithm is used, overall runtime is substantially reduced. *Middle:* the quality of the approximation degrades very slowly, as measured by the energy gap and Hamming distance relative to the optimal solution. *Right:* Effect of blocksize on runtime and accuracy.

Fig. 5. *Left*: Mouse dataset (above) and, drosophila dataset (below), with gold standard segmentations. *Right*: Segmentation result (showing only 900 objects) with structured prediction on 10^6 supervoxels with 10^7 variables, using blockwise hierarchical optimization.
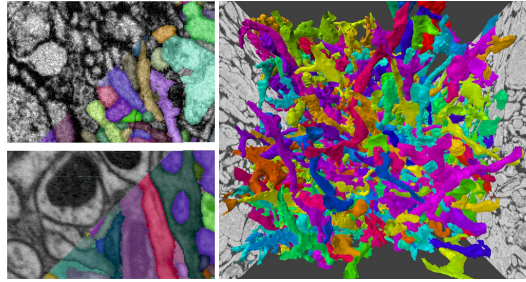
$\theta_{c'} = \sum_{c \in M^{-1}(c')} \theta_c$. Finally, the block size is increased, and the above scheme is applied to $\mathcal{C}'$. In this way, a hierarchy of N levels is created. The final optimization uses no blocking. This algorithm can be parallelized and performs well empirically (Sec. 4).

4   Experiments

The first dataset, (Fig. 5, top left) shows part of adult mouse cerebral cortex at 20nm^3 voxel size (SBEM imaging). The sample was prepared to preserve extracellular space and to suppress intracellular organelle contrast. A $200 \times 300 \times 150$ subset was segmented into 185 segments as gold standard. The second dataset, (Fig. 5, bottom left) shows a part of Drosophila medulla (voxel size 10 nm^3, FIBSEM imaging). Here, organelles such as mitochondria are also stained. 49 blocks of 100^3 have been partially segmented (covering about 2/3 of each volume) as gold standard[2]. Note that both datasets have isotropic voxel size and are therefore amenable to a true 3D approach, as opposed to thick slice data, such as from TEM imaging.

A watershed transform yields an oversegmentation. Then, the voxel ground-truth is projected onto these supervoxels to create ground-truth for $c_2 \in C_2$. The feature vector $\boldsymbol{\alpha}^{(i)}$ describes a surface c_2 which separates two adjacent supervoxels c_3^A and c_3^B. Given different voxel features (smoothed data, 1st and 2nd derivative filters), several statistics over the voxels near the surface are computed. Additional features include topological and geometric features such as $\text{size}(c_2)$, $|\text{size}(c_3^A) - \text{size}(c_3^B)|$, ratio between circumference to area of c_2 and number of adjacent surfaces.

The gold standard of each dataset is divided into training and test blocks. We sample connected components of $|C_2| \approx 27$ surfaces and their gold standard labeling to create training samples $(\boldsymbol{x}^n, \boldsymbol{y}^n)$, $n = 1 \ldots 200$. In order to maximize the number of supervoxels involved in each sample, we only consider, for each sample, surfaces that intersect the same axis-aligned plane. To capture the asymmetric distribution of edges $y_i = 0$ versus $y_1 = 1$, the sampling algorithm attempts to obtain samples with a ratio of $p(y_i = 1)/p(y_i = 0)$ estimated from the gold standard segmentation.

Enumerating all (hundreds of thousands, see Fig. 3) segmentations for one sample takes only about a second, thanks to the efficient enumeration from Sec. 2. About 3 seconds are needed to precompute different loss functions (RE, VI, Hamming). Both the list of segmentations (compressed to 4 bytes per segmentation) and the losses are stored

---

[2] Groundtruth for both datasets can be found on the first authors homepage.
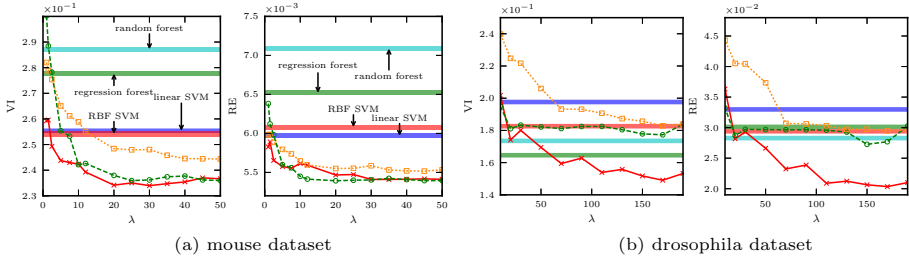
Fig. 6. VI and RE (lower is better) as a function of the regularization parameter λ. Results have been averaged over multiple blocks. For struct-SVM, using the structured loss functions VI "$\times$" or RE "$\circ$" gives substantially better results than the unstructured Hamming loss "$\square$".

on disk to be reused during structured learning. Fig. 3 shows that, for most samples, we have to consider about half a million possible segmentations. Finally the vector $\phi(\boldsymbol{x}, \boldsymbol{y})$ is precomputed for all training samples. When the separation oracle asks for the most violated constraint during cutting-plane training, we only need to compute a dot product of the current weight vector and ϕ for each segmentation and look up the segmentation's loss w.r.t. the gold standard. In addition, it parallelizes easily. Training usually needs about 200 iterations for convergence on 8 CPUs and takes about 20 minutes.

For prediction, the trained model is optimized on several blocks taken from the test portion of each dataset. We compare our results to unstructured methods. Different binary classifiers are trained using a training set which consists of the union of all surfaces involved in any training sample and their gold standard label. For random forest and regression forest classifiers, the probabilistic output is transformed into weights $\boldsymbol{\theta}$ via (4); for linear SVM and RBF SVM classifiers, the weights are taken to be the distance from the margin. Hyperparameters (regression forest: tree depth; linear SVM: regularization strength λ; RBF SVM: γ, λ) are optimized via cross-validation.

Fig. 4 shows VI and RE, averaged over multiple test blocks as a function of the regularization parameter λ with respect to the performance of unstructured methods. For the mouse dataset our approach is able to outperform both an unstructured learning of $\boldsymbol{\theta}$ as well as structured learning with decomposable Hamming loss, while being insensitive to the exact choice of hyperparameter λ. Choosing either VI and RE loss during learning yields similar performance (as measured by VI or RE) with respect to the gold standard. On the drosophila dataset, using VI for learning improves over unstructured methods; interestingly the RE loss does no better than unstructured learning. Note also that the relative performance of unstructured SVM and Random Forest is inverted between both datasets, emphasizing the distinctness of the two problems.

5   Conclusion

This paper addresses the problem of learning a supervised segmentation algorithm with arbitrary loss functions. A structured support vector machine has been used to learn weights for correlation clustering on an edge-weighted region adjacency graph.

Our first contribution is an efficient exhaustive enumeration of segmentations for small subsets of the training data for loss-augmented prediction. This allows to train on the same structured loss functions (RE or VI) as are used for evalutation of the segmentation quality. We find that, for a neighborhood of $|C_2| \approx 27$ and the linear struct-SVM classifier, structured learning with a structured loss function can beat more complex, but unstructured classifiers in two different microscopic modalities, but still fails to match the quality of a human expert. Our second contribution, a hierarchical blockwise scheme for structured prediction, enables us to analyze a 1000^3 dataset involving over 10 million variables, which was broken up initially into 3,000 blocks. The complete hierarchical blockwise segmentation takes about a day, but can be easily parallelized over the independent subproblems. Fig. 5, right, shows 900 objects from the final partitioning.

References

1. 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: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 778–791. Springer, Heidelberg (2012)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI 33(5), 898–916 (2011)
3. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1), 89–113 (2004)
4. Finley, T., Joachims, T.: Supervised clustering with support vector machines. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 217–224. ACM (2005)
5. Chopra, S., Rao, M.: The partition problem. Mathematical Programming 59(1), 87–115 (1993)
6. Vitaladevuni, S., Basri, R.: Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction. In: CVPR (2010)
7. Kim, S., Nowozin, S., Kohli, P.: Task-specific image partitioning. IEEE TIP 22(2), 488–500 (2013)
8. Andres, B., Kappes, J.H., Beier, T., Köthe, U., Hamprecht, F.A.: Probabilistic image segmentation with closedness constraints. In: ICCV (2011)
9. Kappes, J.H., Speth, M., Andres, B., Reinelt, G., Schn, C.: Globally optimal image partitioning by multicuts. In: Boykov, Y., Kahl, F., Lempitsky, V., Schmidt, F.R. (eds.) EMMCVPR 2011. LNCS, vol. 6819, pp. 31–44. Springer, Heidelberg (2011)
10. Bagon, S., Galun, M.: A multiscale framework for challenging discrete optimization. In: NIPS Workshop on Optimization for Machine Learning (2012)
11. Yarkony, J., Ihler, A., Fowlkes, C.C.: Fast planar correlation clustering for image segmentation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VI. LNCS, vol. 7577, pp. 568–581. Springer, Heidelberg (2012)
12. Turaga, S., Briggman, K., Helmstaedter, M., Denk, W., Seung, H.: Maximin affinity learning of image segmentation. NIPS (2009)

13. Jain, V., Turaga, S.C., Briggman, K.L., Helmstaedter, M.N., Denk, W., Seung, H.S.: Learning to agglomerate superpixel hierarchies. NIPS 2(5) (2011)
14. Tarlow, D., Zemel, R.S.: Structured output learning with high order loss functions. In: AISTATS (2012)
15. Ranjbar, M., Mori, G., Wang, Y.: Optimizing complex loss functions in structured prediction. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 580–593. Springer, Heidelberg (2010)
16. Nowozin, S., Lampert, C.: Structured learning and prediction in computer vision. Now Publishers Inc. (2011)
17. Hatcher, A.: Algebraic Topology. Cambridge Univ. Press (2002)