

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

Graph-Based Neural Reconstruction from Skeletonized 3D Networks

Anonymous CVPR submission

Paper ID 0446

Abstract

Advancements in electron microscopy image acquisition have created massive connectomics datasets in the terabyte range that make manual reconstruction of neuronal structures infeasible. Current state-of-the-art automatic methods segment neural membranes at the pixel level followed by agglomeration methods to create full neuron reconstructions. However, these approaches widely neglect global geometric properties that are inherent in the graph structure of neural wiring diagrams. In this work, we follow bottom-up pixel-based reconstruction by a top-down graph-based method to more accurately approximate neural pathways. We first generate skeletons in 3D from the membrane labels of the pixel-based segmentation. We then simplify this skeletonized 3D network into a 3D graph with nodes corresponding to labels from the segmentation and edges identifying potential locations of segmentation errors. We use a CNN classifier trained on ground truth data to generate edge weights on the 3D graph corresponding to error probabilities. We then apply a multicut algorithm to generate a partition on the graph that improves the final segmentation. Because the 3D graph is small and encodes top-down information our method is efficient and globally improves the neural reconstruction. We demonstrate the performance of our approach on multiple real-world connectomics datasets with an average variation of information improvement of $X \times$.

1. Introduction

The field of connectomics is concerned with reconstructing the wiring diagram of the brain at nanometer resolutions to enable new insights into the workings of the brain [8, 18]. Recent advancements in image acquisition using multi-beam serial-section electron microscopy (sSEM) have allowed researchers to produce

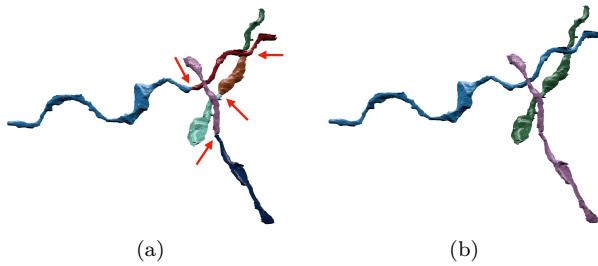


Figure 1: Example improvement of neural reconstruction. (a) We extract 3D skeletons from pixel-based segmentation algorithms to create a 3D graph representation. Edges with high segmentation error probabilities are indicated by the red arrows. (b) We improve the accuracy of segmentation using a graph partitioning algorithm, leveraging both local and global information.

terabytes of image data every hour [12]. It is not feasible for domain experts to manually reconstruct this vast amount of image data [11]. State-of-the-art automatic reconstruction approaches use pixel-based segmentation with convolutional neural networks (CNNs) followed by agglomeration strategies [21, 23, 27, 30, 33, 43]. These bottom-up pixel-based methods produce excellent results but still fall short of acceptable error rates for large volumes.

We present a top-down graph-based method that builds on the outputs of bottom-up pixel-based segmentation approaches. We first extract 3D skeleton networks from the input segmentation and generate a simplified 3D graph (Fig. 1a). We train a CNN classifier on the agglomerated regions in the segmentation data to detect errors. We run the classifier to populate the graph edge weights with error probabilities. We then use a graph optimization algorithm to partition the graph into the final improved reconstruction by enforcing domain-specific global constraints from biology (Fig. 1b).

054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

108 Our approach operates at a level of abstraction
 109 above existing pixel-based methods. This allows us
 110 to leverage both local and global information to pro-
 111 duce more accurate reconstructions. Our method is
 112 independent of image resolution and acquisition pa-
 113 rameters, enabling its application to isotropic and
 114 anisotropic image data without retraining. Using the
 115 3D graph induced by the segmentation allows us to
 116 enforce global biological constraints on the reconstruc-
 117 tion. Our dual approach of assessing local decisions
 118 in a global context yields accuracy improvements over
 119 existing reconstruction methods.
 120

This work makes the following contributions: (1) a novel top-down method using graphs from skeletonized 3D networks for improved neural reconstruction of connectomics data; (2) a region-based CNN classifier to detect errors using the 3D graph as global constraint; (3) an empirical evaluation of our method on several connectomics datasets; (4) our method yields improved performance over a state-of-the-art pixel-based reconstruction approach on average by X percent without drastically increasing the running time.

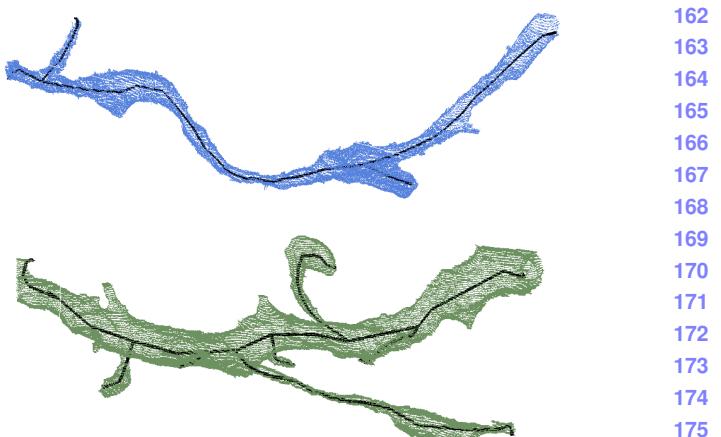
2. Related Work

We review some of the most successful segmentation methods that have been applied to large-scale EM images in connectomics.

Pixel-based methods. A large amount of connectomics research considers the problem of extracting segmentation information at the pixel (i.e., voxel) level from the raw EM images. Some early techniques apply computationally expensive graph partitioning algorithms with a single node per pixel [1]. However, these methods do not scale to terabyte datasets. More recent methods train classifiers to predict membrane probabilities per image slice either using 2D [4, 14, 19, 21, 41] or 3D CNNs [23, 33, 40].

Oftentimes these networks produce probabilities for the affinity between two voxels (i.e., the probability that adjacent voxels belong to the same neuron). The MALIS cost function is specifically designed for generating affinities that produce good segmentations [2]. More recently, flood-filling networks produce segmentations by training an end-to-end neural network that goes from EM images directly to label volumes [16]. These networks produce impressive accuracies but at a high computational cost.

Region-based methods. Several pixel-based approaches generate probabilities that neighboring pixels belong to the same neuron. Often a watershed



162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215

Figure 2: Example skeletons (in black) extracted from segments (blue and green) using the TEASER algorithm.

algorithm will then cluster pixels into super-pixels [43]. Many methods build on top of these region-based strategies and train random-forest classifiers to produce the final segmentations [21, 27, 29, 30, 43].

Error-correction methods. Some recent research builds on top of these region-based methods to correct errors in the segmentation either using human proofreading [15, 22, 10, 9] or fully automatically [32, 44]. However, to our knowledge, our method is the first to extract a 3D graph from pixel-based segmentations for a true top-down error correction approach. This allows us to enforce domain-specific biology constraints and efficient graph partitioning algorithms. Many segmentation and clustering algorithms use graph partitioning techniques [1] or normalized cuts for traditional image segmentation [17, 36, 38]. Even though graph partitioning is an NP-Hard problem [5] there are several useful multicut heuristics that provide good approximations with reasonable computational costs [13]. We use the method of Keuper et al. [20] to partition the extracted 3D graph into the final neural reconstruction.

3. Method

There are two types of errors that can occur in connectomics segmentation. The first, called a split error, occurs when there are two segments that should have been merged. The second, called a merge error, happens when one segment should be split into two. Generally, it is much more difficult to correct merge errors than to correct split errors, as the space of possible split proposals grow quickly [28]. Thus, most reconstruction

216 approaches are tuned towards over-segmentation with
 217 many more split than merge errors. Our method takes
 218 as input over-segments of EM image volumes gen-
 219 erated by state-of-the-art connectomics reconstruc-
 220 tion pipelines (Sec. 4.2). Our goal is to identify loca-
 221 tions of split errors and merge the corresponding seg-
 222 ments automatically.
 223

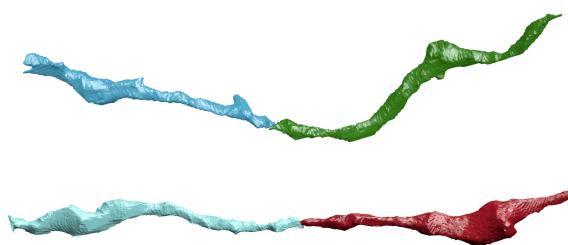
224 From the input segmentation we generate a graph
 225 G with nodes N and edges E with non-negative edge
 226 weights w_e . The nodes correspond to label segments
 227 from the segmentation with edges between segments
 228 considered for merging. Ideally, our graph has edges
 229 corresponding to all of the segments that were erro-
 230 neously split. To compute this graph we generate a
 231 skeleton for every segment in the pixel-based segmen-
 232 tation. The skeleton is a simplified representation of
 233 the overall shape of the neurons. From this skeleton we
 234 identify potential merge locations and produce the cor-
 235 responding edges for the graph. To find actual merges
 236 we run a classification CNN to generate edge weights
 237 that correspond to probabilities of merging. We then
 238 use a multicut heuristic to generate a partition on the
 239 graph where nodes in the same partition are assigned
 240 the same output label in the improved segmentation.
 241 We will now discuss the three major components to our
 242 framework (graph creation, edge weights assignment,
 243 and graph partitioning) in more detail.

3.1. Node Generation

244 The simplest node generation strategy creates one
 245 node for every unique segment label in the input vol-
 246 ume. However, some of the millions of labels in the
 247 volume correspond to very small structures that are
 248 likely the result of segmentation errors. This typically
 249 happens in regions with noisy image data such that
 250 pixel-based methods could not generate the correct
 251 segments. It is difficult to extract useful shape fea-
 252 tures from these segments because of their small, and often
 253 random, shape. We prune these nodes from the graph
 254 by removing all segments with fewer than a threshold
 255 of t_{seg} voxels. We use $t_{seg} = 20,000$ voxels, which
 256 removed on average 56% of the segments in our datasets
 257 (Sec. 4.1). Despite the large number of segments, these
 258 regions only take up 1.6% of the total volume on aver-
 259 age.
 260

3.2. Edge Generation

261 A typical approach to generating edges produces an
 262 edge between all adjacent segments. Two segments l_1
 263 and l_2 are considered adjacent if there is a pair of ad-
 264 jacent voxels where one has label l_1 and the other has
 265 label l_2 . For example, pixel-based agglomeration meth-
 266 ods such as NeuroProof [29] and GALA [27] consider
 267



270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323

Figure 3: Two erroneously split segments.

300 all pairs of adjacent segments for merging. However,
 301 this method produces too many edges in the graph
 302 for graph-based optimization approaches. We identify
 303 a smaller number of pairs of segments to consider as
 304 graph edges using the following algorithm.
 305

306 First, we extract a skeleton from each segment us-
 307 ing the TEASER algorithm [34, 42]. Fig. 2 shows an
 308 example of two extracted skeletons (in black) of two
 309 segments from the label volume. These skeletons con-
 310 sist of a sequence of joints, i.e., locations that are a
 311 local maximum distance from the segment boundary,
 312 with line segments connecting successive joints. We
 313 prune the joints that are within $t_{jnt} = 50$ voxels of
 314 each other to reduce unnecessary branching. For the
 315 purposes of our algorithm, joints that have only one
 316 connected neighbor are referred to as endpoints. Many
 317 of the segments that are erroneously split have nearby
 318 endpoints (Fig. 3). We make use of this fact to merge
 319 segments with the following two-pass heuristic.
 320

321 In the first pass, we iterate over all endpoints e be-
 322 longing to a segment S and create a set of segments \mathbb{S}'_e
 323 that includes all labels that are within t_{low} voxels from
 324 e . Elements of \mathbb{S}'_e are candidates for merging. However,
 325 this first pass often leads to too many candidates, re-
 326 quiring an additional pass for further pruning. In the
 327 second pass, we consider all of the segments in \mathbb{S}'_e for
 328 every endpoint e . If a segment $S' \in \mathbb{S}'_e$ has an endpoint
 329 within t_{high} voxels of e , the segment S and S' are
 330 considered for merging. We store the midpoints between
 331 the two endpoints as the center of the potential merges
 332 in the set \mathbb{S}_c .
 333

3.3. Edge Weights Assignment

334 We assign non-negative edge weights w_e to each edge
 335 where the weight corresponds to the probability two
 336 nodes belong to the same neuron. Instead of using
 337 handcrafted features to compute the similarity between
 338 adjacent nodes, we train a 3D CNN classifier to learn
 339 from the manually labeled oversegmentation input vol-
 340 ume (Sec. 4.1).
 341

324 3.3.1 Classifier Input 378

325 We extract a cubic region of interest around each endpoint e in \mathbb{S}_c as input to the CNN. These regions of 379 information provide the local information for the neural 380 network to predict which neighboring segments belong 381 to the same neuron. The CNN receives three input 382 channels for every voxel in the region of interest around 383 segments l_1 and l_2 . The input in all of the channels is 384 in the range $\{-0.5, 0.5\}$. The first channel is 0.5 only if 385 the corresponding voxel has label l_1 . The second channel 386 is 0.5 only if the corresponding voxel has label l_2 . The 387 third channel is 0.5 if the corresponding voxel is 388 either l_1 or l_2 .

389 3.3.2 Network Architecture & Training 338

390 Fig. 4 provides an overview of our CNN architecture. 391 Similar to Chatfield et al. [3] it consists of three layers 392 of double convolutions followed by a max pooling 393 step. The first max pooling layer is anisotropic with 394 pooling only in the x and y dimensions. The output 395 of this final pooling step is flattened into a 1D vector 396 that is input into two fully connected layers. The final 397 layer produces probabilities with a sigmoid activation 398 function [6]. All of the other activation functions are 399 LeakyReLU [24].

400 We use a stochastic gradient descent optimizer with 401 Nesterov’s accelerated gradient [26] for training. We 402 employ dropouts of 0.2 after every pooling layer and 403 the first dense layer, and a dropout of 0.5 after the 404 final dense layer to prevent overfitting. We discuss all 405 other network parameters in Sec. 4.4.

406 3.4. Graph Partitioning 407

408 After constructing the 3D graph we apply graph- 409 based partitioning to compute the final segmentation. 410 Using graph partitioning from the top down allows us 411 to apply biological constraints on the output. Neuro- 412 scientists know that neuronal connectivity graphs in 413 the brain are acyclic (i.e., the graphs have a genus of 414 zero). We enforce this constraint by finding a multicut 415 partition of the graph that generates a forest of nodes. 416 A forest is a partitioning of a graph into a set of trees 417 where no segment has a cycle. To solve this constraint- 418 ing multicut problem we use the method by Keuper et 419 al. [20] that produces a feasible solution by greedy 420 additive edge contraction. Adding additional constraints 421 and improving the graph partition algorithm are areas 422 for future work.

423 4. Experimental Results 424

425 We evaluate our method on two connectomics 426 datasets and compare it to a state-of-the-art pixel- 427

428 based reconstruction method using the variation of 429 information (VI) metric. **HP:** depending on space we 430 can shorten this and put details into the supplemental 431 material

432 4.1. Datasets 433

434 We use the following two datasets for training and 435 experimental evaluation.

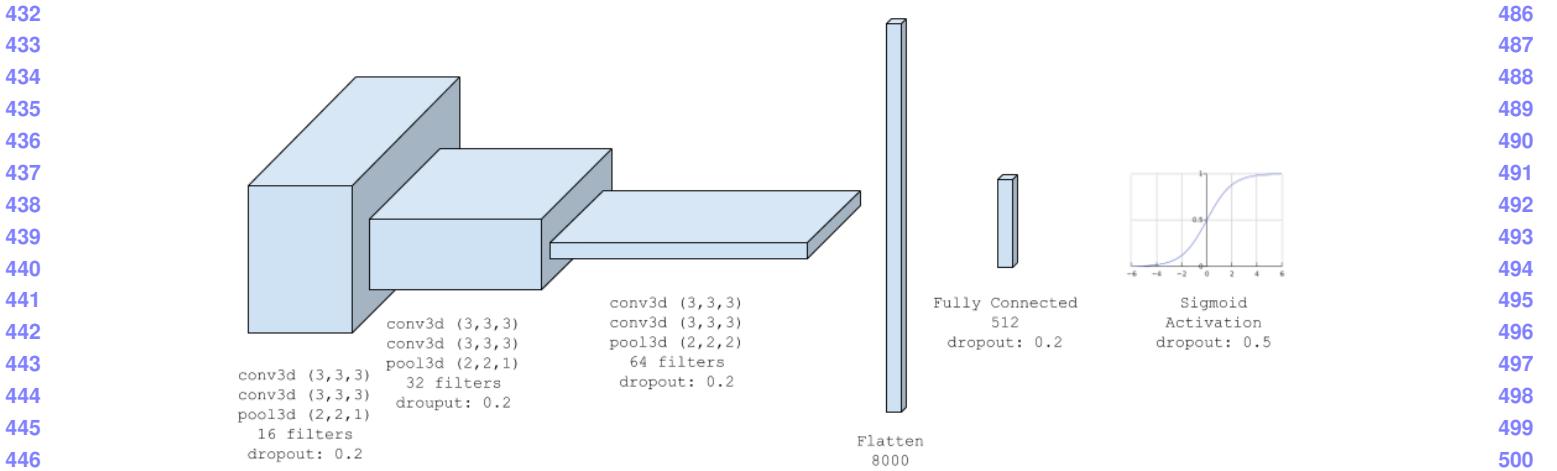
436 Kasthuri. The Kasthuri dataset consists of scanning 437 electron microscope images of the neocortex of a mouse 438 brain [18]. This dataset is $5342 \times 3618 \times 338$ voxels in 439 size. The resolution of the dataset is $3 \times 3 \times 30 \text{ nm}^3$ 440 per voxel. We evaluate our methods using the left 441 cylinder of this 3-cylinder dataset. We downsample the 442 dataset in the x and y dimensions to give a final resolution 443 of $6 \times 6 \times 30 \text{ nm}^3$ per voxel. We divide the dataset 444 into two volumes (Vol. 1 and Vol. 2) along the x dimension 445 where each volume is $8.0 \times 10.9 \times 10.1 \mu\text{m}^3$.

446 FlyEM. The FlyEM dataset comes from the mushroom 447 body of a 5-day old adult male Drosophila fly imaged by a focused ion-beam milling scanning electron 448 microscopy [37]. The mushroom body in this species 449 is the major site of associative learning. The original 450 dataset contains a $40 \times 50 \times 120 \mu\text{m}^3$ volume of which 451 we use two cubes of size $10 \times 10 \times 10 \mu\text{m}^3$. The resolution 452 of this dataset is $10 \times 10 \times 10 \text{ nm}^3$ or 1000 voxels 453 in each dimension in each of the two volumes.

454 4.2. Pixel-Based Segmentations 455

456 The segmentation on the Kasthuri dataset was com- 457 puted by agglomerating 3D supervoxels produced by 458 the z-watershed algorithm from 3D affinity predictions 459 [43]. A recent study by Funke et al. [35] demon- 460 strated superior performance of such methods over ex- 461 isting ones on anisotropic data. We learn 3D affinities 462 using MALIS loss with a U-net [39, 33]. We apply 463 the z-watershed algorithm with suitable parameters to 464 compute a 3D oversegmentation of the volume. The re- 465 sulting 3D oversegmentation is then agglomerated us- 466 ing the technique of context-aware delayed agglomera- 467 tion to generate the final segmentation [29]. We use 468 this segmentation as our baseline for comparisons with 469 our method.

470 For the FlyEM data we collected two $1000 \times 1000 \times$ 471 1000 voxel ($10 \times 10 \times 10 \mu\text{m}^3$) volumes from the 472 authors [37]. Based on the authors’ suggestion, we 473 applied a context-aware delayed agglomeration algo- 474 rithm [29] that shows improved performance on this 475 dataset over the pipeline used in the original pub- 476 lication. This segmentation framework learns voxel



and supervoxel classifiers with an emphasis to minimize under-segmentation error. At the same time this framework produces lower oversegmentation than standard algorithms. The algorithm first computes multi-channel 3-D predictions for membranes, cell interiors, and mitochondria, among other cell features. The membrane prediction channel is used to produce an over-segmented volume using 3D watershed, which is then agglomerated hierarchically up to a certain confidence threshold. We used exactly the same parameters as the publicly available code for this algorithm.

4.3. Graph Generation Parameters

There are two essential parameters to the graph generation algorithm described in Section 3.1: t_{low} and t_{high} . Ideally, the merge candidates output by this algorithm will contain all possible positive examples with a very limited number of negative examples. After considering various thresholds, we find that $t_{low} = 240\text{ nm}$ and $t_{high} = 600\text{ nm}$ produce the best results considering this objective.

In our implementation we use nanometers for these thresholds and not voxels. Connectomics datasets often have lower sample resolutions in z . Using nanometers allows us to have uniform units across all of these datasets and calculate the thresholds in voxels at runtime. For example, the thresholds in voxels are $t_{low} = (40, 40, 8)$ and $t_{high} = (100, 100, 20)$ for the anisotropic Kasthuri dataset and $t_{low} = (24, 24, 24)$ and $t_{high} = (60, 60, 60)$ for the isotropic FlyEM dataset.

4.4. Classifier Training

We use the left cylinder of the Kasthuri dataset for training and validation. We train on 80% of the po-

tential merge candidates for this volume. We validate the CNN classifier on the remaining 20% of candidates. We apply data augmentation to the generated examples to increase the size of the training datasets. We consider all rotations of 90 degrees along the xy -plane in addition to mirroring along the x and z axes. This produces 16 times more training data.

We consider networks with varying input sizes, optimizers, loss functions, filter sizes, learning rates, and activation functions. The supplemental material includes information on the experiments that determined these final parameters. Table 1 provides the parameters of the final network. There are 7,294,705 learnable parameters in our final architecture. All the parameters are randomly initialized following the Xavier uniform distribution [7]. Training concluded after 34 epochs.

Parameters	Values
Loss Function	Mean Squared Error
Optimizer	SGD with Nesterov Momentum
Momentum	0.9
Initial Learning Rate	0.01
Decay Rate	$5 * 10^{-8}$
Activation	LeakyReLU ($\alpha = 0.001$)
Kernel Sizes	$3 \times 3 \times 3$
Filter Sizes	$16 \rightarrow 32 \rightarrow 64$

Table 1: Training parameters.

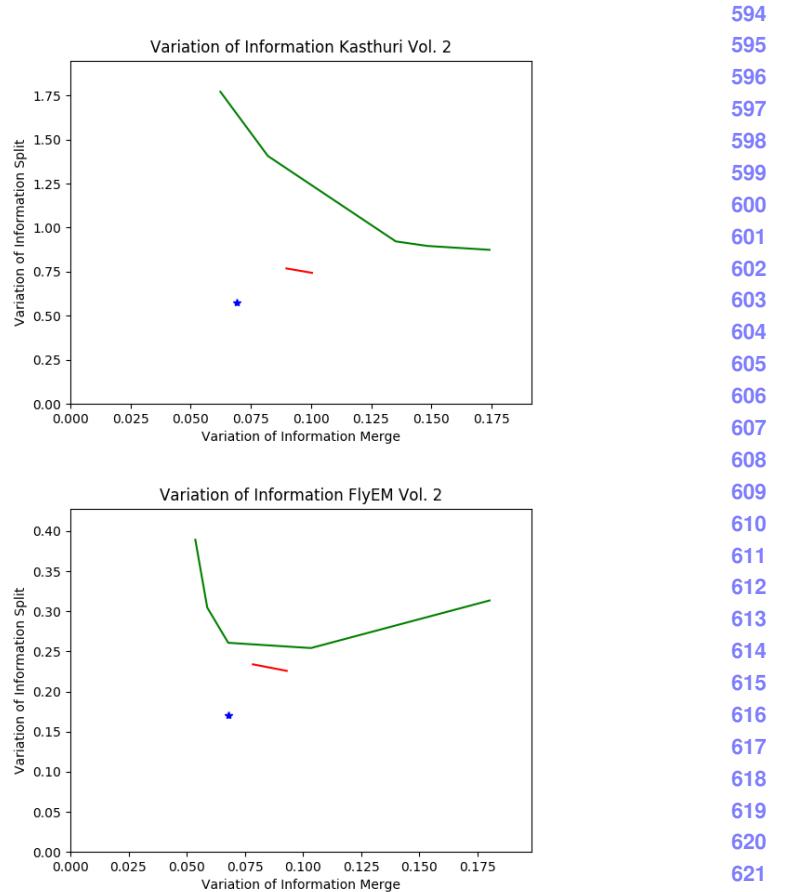
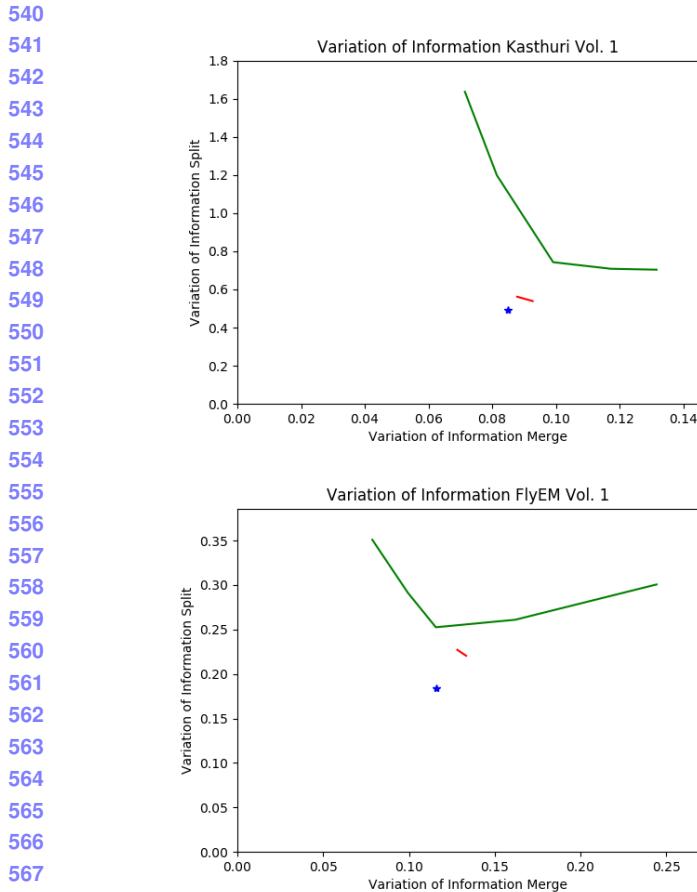


Figure 5: VI scores of our method (red) compared to the baseline segmentation (green) and an oracle (blue) that optimally partitions the graph based on ground truth.

5. Results

5.1. Error Metric

We evaluate the performance of the different methods using the split version of variance of information (VI) [25]. Given a ground truth labeling GT and our automatically reconstructed segmentation SG , over and undersegmentation are quantified by the conditional entropy $H(GT|SG)$ and $H(SG|GT)$, respectively. Since we are measuring the entropy between two clusterings, better VI scores are closer to the origin.

5.2. Variation of Information Improvement

In Fig. 5, we show the VI results of NeuroProof on the Kasthuri and FlyEM data at varying thresholds of agglomeration (green). The green circle represents the variation of information for our input segmentation (a threshold of 0.3 for all datasets). Our results are in red. We show the comparison to an oracle (blue) that correctly partitions the graph from our algorithm based

on ground truth.

Our algorithm improves the accuracy of the input segmentation on every dataset, reducing the VI split score on average by X% and only increasing the VI merge score by X%. Scores closer to the origin are better for this metric, and in every instance we are below the green curve. We see significant improvements on the Kasthuri datasets (VI split reduction of X% and X% on the training and testing datasets respectively) and slightly more modest improvements on the FlyEM datasets (reduction of X% and X%). However, our baseline, NeuroProof performs much better on the FlyEM datasets reducing the potential improvement. Isotropic datasets are easier to segment using state-of-the-art region-based methods than anisotropic ones [31]. Thus there is less room for improvement on these datasets.

Fig. 6 shows successful merges on the Kasthuri Vol. 2 dataset. Several of these examples combine multiple consecutive segments that span the volume. In the third example we correct the oversegmentation of a

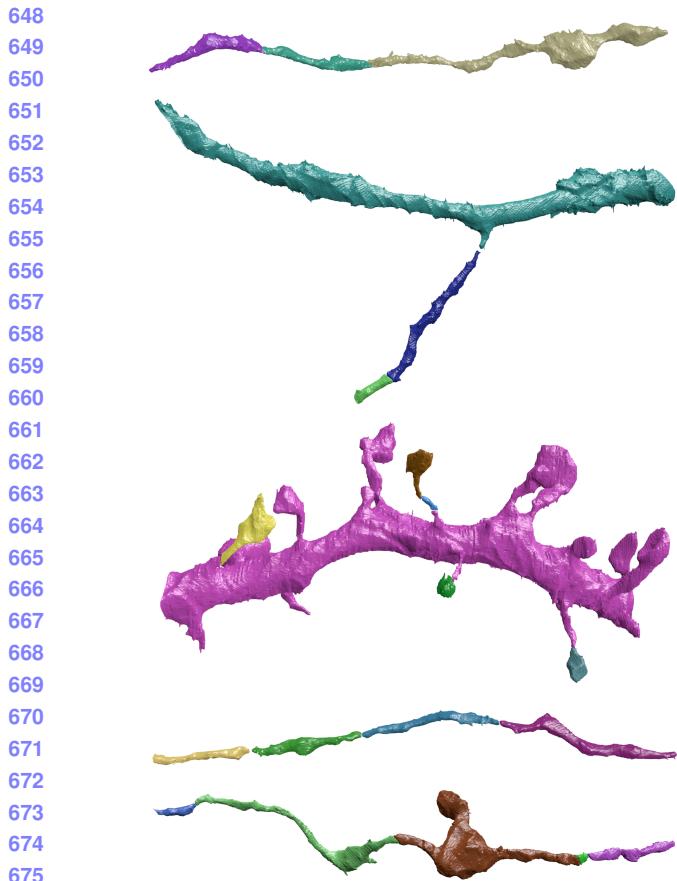


Figure 6: Correctly segmented neurons from our method.



Figure 7: Errors made by our method.

Dataset	Baseline	After Pruning
Kasthuri Vol. 1	763 / 21242 (3.47%)	753 / 3459 (17.88%)
Kasthuri Vol. 2	1010 / 26073 (3.73%)	904 / 4327 (17.28%)
FlyEM Vol. 1	269 / 14875 (1.78%)	262 / 946 (21.69%)
FlyEM Vol. 2	270 / 16808 (1.58%)	285 / 768 (27.07%)

Table 2: The results of our skeleton graph pruning heuristic compared to the baseline segmentation.

dendrite. Fig. 7 shows some failure cases. In two of these examples the algorithm correctly predicted several merges but made one error. In the third example a merge error in the initial segmentation propagated to our output. We now analyze how each major component of our method contributes to this final result.

5.3. Graph Creation

Table 2 shows the results of pruning the skeleton graph using the heuristic discussed in Sec. 3.1. The baseline algorithm considers all adjacent regions for merging. Our method removes a significant portion

of these candidates while maintaining a large number of the true merge locations. This edge pruning is essential for the graph partitioning algorithm, which has a computational complexity dependence on the number of edges. Our pruning heuristic removes at least $6 \times$ the number of edges between correctly split segments on all datasets, achieving a maximum removal ratio of $20 \times$.

Equally important is the number of split errors that remain after pruning. These are the locations that we want to merge to create a more accurate reconstruction. For every dataset, the number of true split errors remains constant before and after pruning. However, since our heuristic does not enforce an adjacency constraint of two regions when constructing edges in the graph, the difference does not indicate the number of examples excluded by pruning. In fact, our method finds a number of examples that are non-adjacent. Figure 8 shows two example segments with split errors, one that our algorithm missed (top) and one that it identified (bottom), despite the fact that the split segments are not adjacent. Of the successful examples in Figure 6, the second and fourth groupings contain pairs of non-adjacent segments that merge with our algorithm.

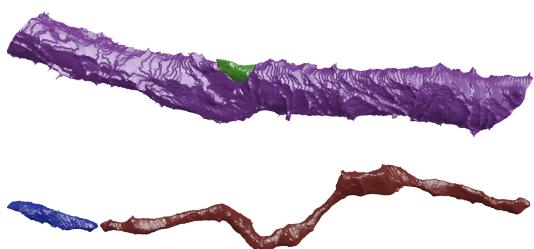


Figure 8: Example merge candidates.

5.4. Classification Performance

Figure 9 shows the receiver operating characteristic (ROC) curve for all of the datasets. We train our CNN using one of the Kasthuri volumes and test using the other three datasets. Since our CNN only takes as input a region of the label volume we can train on an anisotropic dataset and infer on an isotropic one. This provides a major benefit given the time-intensive task of manually generating ground truth data at various resolutions.

As evidenced by the ROC curve, the test results on the Kasthuri data are better than the results for FlyEM. We believe this is in part because of the differences in the datasets (i.e., isotropy and xy resolution). To test this hypothesis, we also evaluate the performance of the FlyEM datasets when the network trains on FlyEM Vol. 1 and infers on FlyEM Vol. 2.¹ The dotted lines in the figure represent these tests. There is a slight performance increase for FlyEM Vol. 2. However, the FlyEM datasets have reasonable results when the network is trained by the Kasthuri data, and the results outside of this section follow from that setup.

5.5. Graph Based Strategies

The graph-based optimization strategy increases our correction accuracy over using just the CNN. In particular, the precision increases on each dataset, although the recall decreases on all but one of the datasets. Since it is more difficult to correct merge errors than split errors, it is often desirable to sacrifice recall for precision. Table 3 shows the changes in precision, recall, and accuracy for all four datasets compared to the CNN. Over the three testing datasets, applying a graph-based partitioning strategy reduced the number of merge errors by X, Y, and Z.

¹Since the FlyEM datasets have significantly fewer examples, we initialize the network with the weights from the Kasthuri training and have an initial learning rate of 10^{-4} .

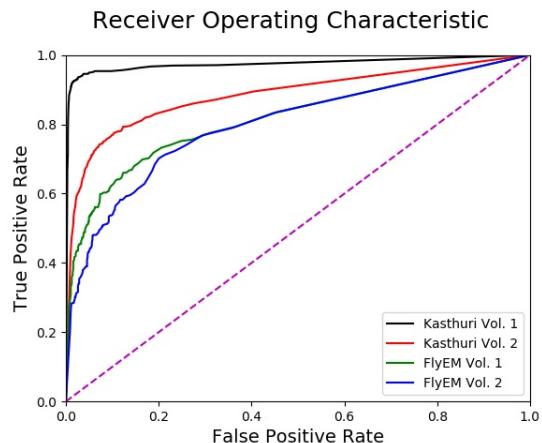


Figure 9: The receiver operating characteristic for all four datasets.

Dataset	Δ Precision	Δ Recall	Δ Accuracy
Kasthuri Training	+3.60%	-0.01%	+0.60%
Kasthuri Testing	+7.59%	-1.77%	+1.38%
FlyEM Vol. 1	+2.68%	+0.76%	+0.66%
FlyEM Vol. 2	+2.22%	-1.05%	+0.29%

Table 3: Precision and recall for the training and three test datasets.

6. Conclusions

We present a novel method for reconstructing neuronal processes in connectomics datasets. We extend existing region-based reconstruction strategies using a morphological 3D skeleton graph and show significant accuracy improvement on two datasets from two different species. Our approach enables enforcing domain-constraints on the resulting segmentation. We believe this approach extends beyond connectomics to other domains with different constraints.

There is significant room for additional research related to this topic. We can still extract a lot more information from the skeletons of each segment. Currently these skeletons identify potential merge locations. However, these representations can be used to identify improperly merged segments. Additionally, we can further improve our results by tweaking the partitioning cost functions to more closely match the underlying biology. One such extension might take the outputs of a synaptic detector to enforce the constraint that a given segment only has post- or pre-synaptic connections.

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

References

- [1] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In European Conference on Computer Vision, pages 778–791. Springer, 2012. [2](#)
- [2] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga. Maximin affinity learning of image segmentation. In Advances in Neural Information Processing Systems, pages 1865–1873, 2009. [2](#)
- [3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531, 2014. [4](#)
- [4] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In Advances in neural information processing systems, pages 2843–2851, 2012. [2](#)
- [5] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. Theoretical Computer Science, 361(2-3):172–187, 2006. [2](#)
- [6] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. Neural networks, 2(3):183–192, 1989. [4](#)
- [7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pages 249–256, 2010. [5](#)
- [8] D. Haehn, J. Hoffer, B. Matejek, A. Suissa-Peleg, A. K. Al-Awami, L. Kamensky, F. Gonda, E. Meng, W. Zhang, R. Schalek, et al. Scalable interactive visualization for connectomics. In Informatics, volume 4, page 29. Multidisciplinary Digital Publishing Institute, 2017. [1](#)
- [9] D. Haehn, V. Kaynig, J. Tompkin, J. W. Lichtman, and H. Pfister. Guided proofreading of automatic segmentations for connectomics. arXiv preprint arXiv:1704.00848, 2017. [2](#)
- [10] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis 2014), 20(12):2466–2475, 2014. [2](#)
- [11] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. W. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. IEEE Transactions on Visualization and Computer Graphics, 20(12):2466–2475, 2014. [1](#)
- [12] D. G. C. Hildebrand, M. Cicconet, R. M. Torres, W. Choi, T. M. Quan, J. Moon, A. W. Wetzel, A. S. Champion, B. J. Graham, O. Randlett, et al. Whole-brain serial-section electron microscopy in larval zebrafish. Nature, 545(7654):345–349, 2017. [1](#)
- [13] A. Horňáková, J.-H. Lange, and B. Andres. Analysis and optimization of graph decompositions by lifted multicut. In International Conference on Machine Learning, pages 1539–1548, 2017. [2](#)
- [14] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstädt, K. Briggman, W. Denk, J. Bowden, J. Mendenhall, W. Abraham, K. Harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and S. Seung. Boundary learning by optimization with topological constraints. In Proc. IEEE CVPR 2010, pages 2488–2495, 2010. [2](#)
- [15] Janelia Farm. Raveler. <https://openwiki.janelia.org/wiki/display/flyem/Raveler>, 2014. Accessed on 11/01/2016. [2](#)
- [16] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain. Flood-filling networks. arXiv preprint arXiv:1611.00421, 2016. [2](#)
- [17] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr. Higher-order segmentation via multicut. Computer Vision and Image Understanding, 143:104–119, 2016. [2](#)
- [18] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. Saturated reconstruction of a volume of neocortex. Cell, 162(3):648–661, 2015. [1, 4](#)
- [19] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. Medical image analysis, 22(1):77–88, 2015. [2](#)
- [20] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicut. In Proceedings of the IEEE International Conference on Computer Vision, pages 1751–1759, 2015. [2, 4](#)
- [21] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister. Rhoanet pipeline: Dense automatic neural annotation. arXiv preprint arXiv:1611.06973, 2016. [1, 2](#)
- [22] S. Knowles-Barley, M. Roberts, N. Kasthuri, D. Lee, H. Pfister, and J. W. Lichtman. Mojo 2.0: Connectome annotation tool. Frontiers in Neuroinformatics, (60), 2013. [2](#)
- [23] K. Lee, A. Zlateski, V. Ashwin, and H. S. Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary prediction. In Advances in Neural Information Processing Systems, pages 3573–3581, 2015. [1, 2](#)
- [24] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML, volume 30, 2013. [4](#)
- [25] M. Meila. Comparing clusterings by the variation of information. In Colt, volume 3, pages 173–187. Springer, 2003. [6](#)

- 972 [26] Y. Nesterov. A method of solving a convex program-
973 ming problem with convergence rate $O(1/k^2)$. 4
974 [27] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza,
975 A. Chakraborty, and W. T. Katz. Graph-based active
976 learning of agglomeration (gala): a python library
977 to segment 2d and 3d neuroimages. *Frontiers in neu-*
978 *roinformatics*, 8, 2014. 1, 2, 3
979 [28] T. Parag. What properties are desirable from an
980 electron microscopy segmentation algorithm. arXiv
981 preprint arXiv:1503.05430, 2015. 2
982 [29] T. Parag, A. Chakraborty, S. Plaza, and L. Scheff-
983 er. A context-aware delayed agglomeration framework
984 for electron microscopy segmentation. *PLOS ONE*,
985 10(5):1–19, 05 2015. 2, 3, 4
986 [30] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga,
987 X. Zhang, B. Matejek, L. Kamentsky, J. W. Licht-
988 man, and H. Pfister. Anisotropic em segmentation by
989 3d affinity learning and agglomeration. arXiv preprint
990 arXiv:1707.08935, 2017. 1, 2
991 [31] S. M. Plaza, T. Parag, G. B. Huang, D. J. Olbris,
992 M. A. Saunders, and P. K. Rivlin. Annotating synapses
993 in large em datasets. arXiv preprint arXiv:1409.1801,
994 2014. 6
995 [32] D. Rolnick, Y. Meirovitch, T. Parag, H. Pfister,
996 V. Jain, J. W. Lichtman, E. S. Boyden, and N. Shavit.
997 Morphological error detection in 3d segmentations.
998 arXiv preprint arXiv:1705.10882, 2017. 2
999 [33] O. Ronneberger, P. Fischer, and T. Brox. U-net:
1000 Convolutional networks for biomedical image segmen-
1001 tation. In *International Conference on Medical Im-
1002 age Computing and Computer-Assisted Intervention*,
1003 pages 234–241. Springer, 2015. 1, 2, 4
1004 [34] M. Sato, I. Bitter, M. A. Bender, A. E. Kaufman, and
1005 M. Nakajima. Teasar: Tree-structure extraction algo-
1006 rithm for accurate and robust skeletons. In *Computer
1007 Graphics and Applications, 2000. Proceedings. The
1008 Eighth Pacific Conference on*, pages 281–449. IEEE,
1009 2000. 3
1010 [35] P. Schlegel, M. Costa, and G. S. Jefferis. Learning from
1011 connectomics on the fly. *Current Opinion in Insect
1012 Science*, 2017. 4
1013 [36] J. Shi and J. Malik. Normalized cuts and image seg-
1014 mentation. *IEEE Transactions on pattern analysis and
1015 machine intelligence*, 22(8):888–905, 2000. 2
1016 [37] S.-y. Takemura, Y. Aso, T. Hige, A. M. Wong, Z. Lu,
1017 C. S. Xu, P. K. Rivlin, H. F. Hess, T. Zhao, T. Parag,
1018 S. Berg, G. Huang, W. T. Katz, D. J. Olbris, S. M.
1019 Plaza, L. A. Umayam, R. Aniceto, L.-A. Chang,
1020 S. Lauchie, and et al. A connectome of a learning and
1021 memory center in the adult drosophila brain. *eLife*,
1022 6:e26975, 2017 Jul 18 2017. 4
1023 [38] S. Tatiraju and A. Mehta. Image segmentation using
1024 k-means clustering, em and normalized cuts. *Depart-
1025 ment of EECS*, 1:1–7, 2008. 2
1026 [39] S. Turaga, K. Briggman, M. Helmstaedter, W. Denk,
1027 and S. Seung. Maximin affinity learning of image seg-
1028 mentation. In *Advances in Neural Information Pro-
1029 cessing Systems* 22. 2009. 4
1030 [40] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helm-
1031 staedter, K. Briggman, W. Denk, and H. S. Seung.
1032 Convolutional networks can learn to generate affinity
1033 graphs for image segmentation. *Neural computation*,
1034 22(2):511–538, 2010. 2
1035 [41] A. Vázquez-Reina, M. Gelbart, D. Huang, J. Licht-
1036 man, E. Miller, and H. Pfister. Segmentation fusion
1037 for connectomics. In *Proc. IEEE ICCV*, pages 177–
1038 184, Nov 2011. 2
1039 [42] T. Zhao and S. M. Plaza. Automatic neuron type
1040 identification by neurite localization in the drosophila
1041 medulla. arXiv preprint arXiv:1409.1892, 2014. 3
1042 [43] A. Zlateski and H. S. Seung. Image segmentation by
1043 size-dependent single linkage clustering of a watershed
1044 basin graph. arXiv preprint arXiv:1505.00249, 2015.
1045 1, 2, 4
1046 [44] J. Zung, I. Tartavull, and H. S. Seung. An error
1047 detection and correction framework for connectomics.
1048 *CoRR*, abs/1708.02599, 2017. 2
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079