# A Generalized Framework for Agglomerative Clustering of Signed Graphs applied to Instance Segmentation

**Alberto Bailoni**[1], **Constantin Pape**[1,2], **Steffen Wolf**[1], **Thorsten Beier**[2],
**Anna Kreshuk**[2], **Fred A. Hamprecht**[1]

[1]HCI/IWR, Heidelberg University, Germany
[2]EMBL, Heidelberg, Germany
{alberto.bailoni, steffen.wolf, fred.hamprecht}@iwr.uni-heidelberg.de
{constantin.pape, thorsten.beier, anna.kreshuk}@embl.de

## Abstract

We propose a novel theoretical framework that generalizes algorithms for hierarchical agglomerative clustering to weighted graphs with both attractive and repulsive interactions between the nodes. This framework defines GASP, a Generalized Algorithm for Signed graph Partitioning, and allows us to explore many combinations of different linkage criteria and cannot-link constraints. We prove the equivalence of existing clustering methods to some of those combinations, and introduce new algorithms for combinations which have not been studied. An extensive comparison is performed to evaluate properties of the clustering algorithms in the context of instance segmentation in images, including robustness to noise and efficiency. We show how one of the new algorithms proposed in our framework outperforms all previously known agglomerative methods for signed graphs, both on the competitive CREMI 2016 EM segmentation benchmark and on the CityScapes dataset.

## 1 Introduction

In computer vision, the partitioning of weighted graphs has been successfully applied to such tasks as image segmentation, object tracking and pose estimation. Most graph clustering methods work with positive edge weights only, which can be interpreted as similarities or distances between the nodes. These methods are parameter-based and require users to specify the desired numbers of clusters or a termination criterion (e.g. spectral clustering or iterated normalized cuts) or even to add a seed for each object (e.g. seeded watershed or random walker).

Other graph clustering methods work with so-called *signed graphs*, which include both positive and negative edge weights corresponding to attraction and repulsion between nodes. The advantage of signed graphs over positive-weighted graphs is that balancing attraction and repulsion allows us to perform the clustering without defining additional parameters. This can be done optimally by solving the so-called *multicut optimization* or *correlation clustering* problem [32, 11]. This problem is NP-hard, but approximate solvers have already been proposed [8]. Besides, the general problem of graph partitioning can be solved approximately by greedy agglomerative clustering [36, 49, 81, 33].

Agglomerative clustering algorithms for signed graphs have clear advantages: they are parameter-free and efficient. Despite the fact that there exists a variety of these algorithms, no overarching study has so far been made to compare their robustness and efficiency or to provide guidelines for matching an algorithm to the partitioning problem at hand.

Preprint. Under review.

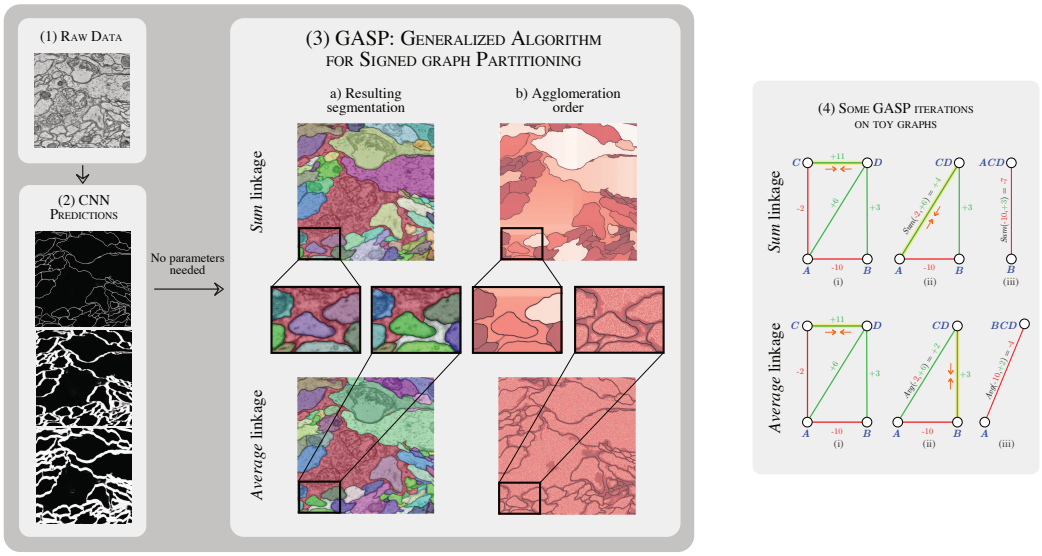



Figure 1: Pipeline description: **(1)** Raw data from the CREMI 2016 neuron-segmentation challenge. **(2)** Some short- and long-range predictions of our CNN model, where white pixels represent boundary evidence. **(3)** Outputs of two agglomerative algorithms included in our proposed generalized clustering framework, with *Sum* and *Average* linkage criteria. The final clustering / instance segmentation is shown in 3a, overlaid with the raw image. The agglomeration order in 3b shows which pairs of neighboring pixels were merged first (white), later on (brown/red), or never (black). (**4**) Some iterations of GASP on toy graph examples with attractive/positive (green) and repulsive/negative (red) interactions. At each iteration, the yellow edge with highest interaction is contracted (orange arrows), until only negative edges are left in the graph.

In this paper, we propose a novel theoretical framework that generalizes over agglomerative algorithms for signed graphs by linking them to hierarchical agglomerative clustering on positive-weighted graphs [45]. This framework defines an underlying basic algorithm and allows us to explore its combinations with different linkage criteria and *cannot-link constraints*. We then formally prove that some of the combinations correspond to existing clustering algorithms and introduce new algorithms for combinations which have not been explored yet.

We evaluate and compare these algorithms on *instance segmentation* - a computer vision task of assigning each pixel of an image to an object instance. We use a CNN to predict the edge weights of a graph such that each node represents a pixel of the image, similarly to [58, 48, 81], and provide these weights as input to the algorithms in our framework (see Fig. 1).

With our comparison experiments, performed both on 2D urban scenes from the CityScapes dataset and 3D electron microscopy image volumes of neurons, we evaluate the properties of the algorithms in our framework, focusing on their efficiency, robustness and tendency to over- or under-cluster. We show that one of the new algorithms derived from our framework, based on an average linkage criterion, outperforms the previously known agglomeration methods expressed in the framework. It also achieves competitive performance on the challenging CREMI 2016 segmentation benchmark and represents the best-performing proposal-free method on CityScapes. Our code is available at `https://github.com/abailoni/GASP`.

## 2 Related work

**Proposal-based methods** have been highly successful in instance segmentation competitions like MS COCO [52], Pascal VOC2012 [20] and CityScapes [15]. They decompose the instance segmentation task into two steps that consists in generating object proposals and assigning to each bounding box a class and a binary segmentation mask [30, 54, 83, 50, 44, 29, 10, 16, 51]. They commonly rely on Faster-RCNN [68] and can be trained end-to-end using non-maximum suppression. Other methods use instead recurrent models to sequentially generate instances one-by-one [70, 67].

**Proposal-free methods** adopt a bottom-up approach by directly grouping pixels into instances. Recently, there has been a growing interest for such methods that do not involve object detection, since, in certain types of data, object instances cannot be approximated by bounding boxes. For example, the approach proposed in [38] uses a combinatorial framework for instance segmentation; SGN [53] sequentially group pixels into lines and then instances; a watershed transform is learned in [6] by also predicting its gradient direction, whereas the template matching [78] deploys scene depth information. Others use metric learning to predict high-dimensional associative pixel embeddings that map pixels of the same instance close to each other, while mapping pixels belonging to different instances further apart [21, 61, 17, 43]. Final instances are then retrieved by applying a clustering algorithm, like in the end-to-end trainable mean-shift pipeline of [41].

**Edge detection** also experienced recent progress thanks to deep learning, both on natural images [82, 40] and biological data [48, 75, 60, 14]. In neuron segmentation for connectomics, a field of neuroscience we also address in our experiments, boundaries are converted to final instances with subsequent postprocessing and superpixel-merging: some use loopy graphs [34, 42] or trees [60, 57, 55, 24, 79] to represent the region merging hierarchy; the lifted multicut [9] formulates the problem in a combinatorial framework, while flood-filling networks [31] eliminate superpixels by training a recurrent CNN to perform region growing one region at the time. A structured learning approach was also proposed in [26, 77].

**Agglomerative graph clustering** has often been applied to instance segmentation [69, 56, 73], because of its efficiency as compared to other top-down approaches like graph cuts. Novel termination criteria and merging strategies have often been proposed: the agglomeration in [59] deploys fixed sets of merge constraints; ultrametric contour maps [3] combine an oriented watershed transform with an edge detector, so that superpixels are merged until the ultrametric distance exceeds a learned threshold; the popular graph-based method [22] stops the agglomeration when the merge costs exceed a measure of quality for the current clusters. The optimization approach in [37] performs greedy merge decisions that minimize a certain energy, while other pipelines use classical HAC linkage criteria, e.g. average linkage [58, 48], median [26] or a linkage learned by a random forest classifier [62, 39].

**Clustering of signed graphs** has the goal of partitioning a graph with both attractive and repulsive cues. Finding an optimally balanced partitioning has a long history in combinatorial optimization [27, 28, 12]. NP-hardness of the *correlation clustering* problem was shown in [7], while the connection with graph multicuts was made by [18]. Modern integer linear programming solvers can tackle problems of considerable size [2], but accurate approximations [63, 8, 84], greedy agglomerative algorithms [49, 80, 36, 33] and persistence criteria [47, 46] have been proposed for even larger graphs.

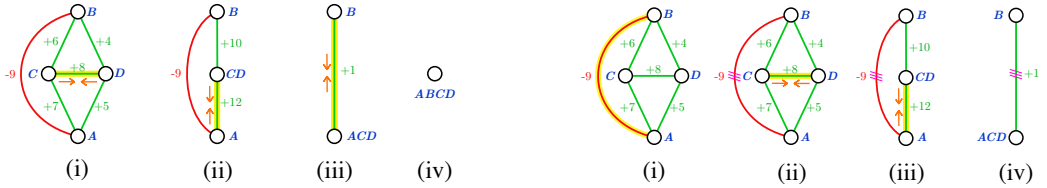
This work reformulates the clustering algorithms of [49, 81, 36] in a generalized framework and adopt ideas from the proposal-free methods [58, 81, 48] to predict long-range relationships between pixels.

## 3 Generalized framework for agglomerative clustering of signed graphs

In this section, we first define notation and then introduce one of our main contributions: a signed graph partitioning algorithm (Sec. 3.2) that can be seen as a generalization of several existing and new clustering algorithms (Sec. 3.3).

### 3.1 Notation and graph formalism

We consider an undirected simple edge-weighted graph $\mathcal{G}(V, E, w^+, w^-)$ with both attractive and repulsive edge attributes. In computer vision applications, the nodes can represent either pixels, superpixels or voxels. We call the set $\Pi$ a *clustering* or *partitioning* with $K$ clusters if $V = \cup_{S\in\Pi} S$, $S \cap S' = \emptyset$ for different clusters $S, S' \in \Pi$ and every cluster $S \in \Pi$ induces a connected subgraph of $\mathcal{G}$. We also denote as $S_u$ the cluster associated with node $u$. The weight function $w^+ : E \to \mathbb{R}^+$ associates to every edge a positive scalar attribute $w_e^+ \in \mathbb{R}^+$ representing a merge affinity or a similarity measure: the higher this number, the higher the inclination of the two incident vertices to




(a) No constraints

(b) With cannot-link constraints

Figure 2: Some iterations of the generalized algorithm (using *Sum* linkage criteria) with and without adding cannot-link constraints. The graph has both attractive (green) and repulsive (red) edges and cannot-link constraints are shown with triple violet bars on the edges. We note that when constraints are enforced, the final clustering is given by two clusters instead of only one.

**Algorithm 1** GASP: generalized algorithm for signed graph partitioning

**Input:** Graph $\mathcal{G}(V, E, w^+, w^-)$; linkage criterion $\mathcal{W}$; boolean `addCannotLinkConstraints`
**Output:** Final clustering $\Pi$

Initialize clustering $\Pi = \{\{v_1\}, \ldots, \{v_{|V|}\}\}$ with each node in its own cluster
Initial interactions between nodes given by $w_e = w_e^+ - w_e^-$
**repeat**
    Select pair of clusters $S_u, S_v \in \Pi$ with highest absolute interaction $|\mathcal{W}(S_u, S_v)|$
    **if** $\left[\mathcal{W}(S_u, S_v) > 0\right]$ **and** $\left[S_u, S_v \text{ are } \textbf{not} \text{ constrained}\right]$ **then**
        Merge cluster $S_u$ with $S_v$: update interactions and cannot-link constraints with all their neighbors
    **else if** $\left[\mathcal{W}(S_u, S_v) \leq 0\right]$ **and** `addCannotLinkConstraints` **then**
        Add CannotLink Constraint between clusters $S_u$ and $S_v$
**until** [all interactions between clusters are repulsive] **or** [all adjacent clusters have cannot-link constraints]
**return** $\Pi$

be assigned to the same cluster[1]. On the other hand, $w^- : E \to \mathbb{R}^+$ associates to each edge a split tendency $w_e^- \in \mathbb{R}^+$: the higher this weight, the more the incident vertices would like to be in different clusters. Graphs of the type $\mathcal{G}(V, E, w^+, w^-)$ are also often defined as *signed graphs* $\mathcal{G}(V, E, w)$, featuring positive and negative edge weights $w_e \in \mathbb{R}$. Following the theoretical considerations in [47], we define these signed weights as $w_e = w_e^+ - w_e^-$. Some approaches directly compute $w_e$, whereas others compute $w_e^+$ and $w_e^-$ separately. In this formalism, graphs with purely attractive interactions are a special case of $\mathcal{G}(V, E, w)$ with $w_e \geq 0$, $\forall e \in E$.

**Inter-cluster interaction** We call two clusters $S_u, S_v$ *adjacent* if there exists at least one edge $e_{ts} \in E$ connecting a node $t \in S_u$ to a node $s \in S_v$. In hierarchical agglomerative clustering, the interaction $\mathcal{W}(S_u, S_v)$ between the two clusters is usually defined as a function $\mathcal{W} : \Pi \times \Pi \to \mathbb{R}$, named *linkage criterion*, depending on the weights of *all* edges connecting clusters $S_u$ and $S_v$, i.e. $(S_u \times S_v) \cap E$. All the linkage criteria tested in this article are listed and defined in Table 1.

### 3.2 GASP: generalized algorithm for signed graph partitioning

In Algorithm 1, we provide simplified pseudo-code for the proposed GASP algorithm. GASP implements a bottom-up approach that starts by assigning each node to its own cluster and then iteratively merges pairs of adjacent clusters. The algorithm has two variants. The first one, with `addCannotLinkConstraints=False`, starts by merging clusters with the strongest attractive interaction and stops once the remaining clusters share only mutual repulsive interactions (see iterations on toy graphs in block 4 of Fig. 1). After each merging iteration, the interaction between the merged cluster and its neighbors is updated according to one of the linkage criteria $\mathcal{W}(S_u, S_v)$ listed in Table 1.

In the second variant, when `addCannotLinkConstraints=True`, Algorithm 1 also introduces *cannot-link constraints*, which represent mutual exclusion relationships between pairs of nodes that

[1]Note that other formalisms for positively weighted graphs associate distances to the edges, thus, the *lower* the edge weight, the higher the attraction between the two linked nodes, contrary to our definition of $w^+$.

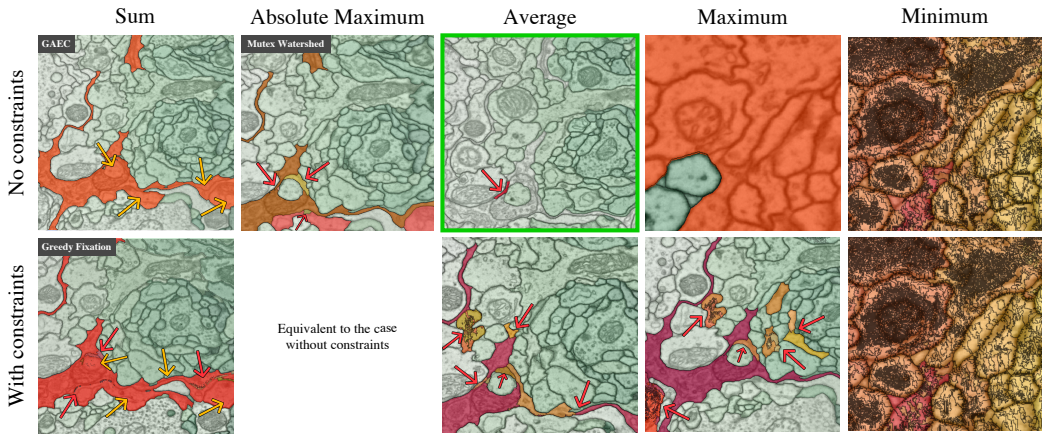



Figure 3: Failure cases of GASP with different linkage criteria highlighted on some difficult parts of the CREMI Challenge data. The main *wrongly* segmented regions are highlighted in different warm colors. Note that the data is 3D, hence the same color could be assigned to parts of segments that appear disconnected in 2D. Red arrows point to wrongly split regions. Yellow arrows point out merge errors. The *Average* linkage without cannot-link constraints returned the best segmentation.

cannot be associated with the same cluster in the final clustering. This variant selects the pair of clusters with the highest absolute interaction $|\mathcal{W}(S_u, S_v)|$, so that the most attractive and the most repulsive pairs are analyzed first (see example in Fig. 2(b)). If the interaction is repulsive, then the two clusters are constrained and its members can never merge in subsequent steps. If the interaction is attractive, then the clusters are merged, provided that they were not previously constrained. The algorithm terminates when all the remaining clusters are constrained.

In Appendix 7.1, we comment on the algorithm's computational complexity $\mathcal{O}(N^2 \log N)$ and present our implementation given by the edge contraction Algorithm 2 based on a priority queue.

### 3.3 GASP with different linkage criteria: new and existing algorithms

Our main contribution is the generalized algorithm for signed graph partitioning, short GASP, that encompasses several known and novel agglomerative algorithms on display in Table 1. In our framework, individual algorithms are differentiated by the linkage criterion employed. We review them in the following paragraphs.

| **GASP linkage criteria** $\mathcal{W}(S_u, S_v)$ | | **Unsigned Graphs** | **Signed Graphs** | |
|---|---|---|---|---|
| | | | No Constraints | With Constraints |
| Sum: | $\sum_{e \in E_{uv}} w_e$ | Sum Linkage Hier. Aggl. Clust. | GAEC [36] | Greedy Fixation [49] |
| Absolute Max: | $w_e$ with $e = \arg\max_{t \in E_{uv}} \lvert w_t \rvert$ | Single Linkage Hier. Aggl. Clust. | Mutex Watershed [81] | Mutex Watershed [81] |
| Average: | $\sum_{e \in E_{uv}} w_e \Big/ \lvert E_{uv} \rvert$ | Average Linkage Hier. Aggl. Clust. | **NEW** | **NEW** |
| Max: | $\max_{e \in E_{uv}} w_e$ | Single Linkage Hier. Aggl. Clust. | **NEW** | **NEW** |
| Min: | $\min_{e \in E_{uv}} w_e$ | Complete Linkage Hier. Aggl. Clust. | **NEW** | **NEW** |

Table 1: Existing and new clustering algorithms that can be reformulated as special cases of the proposed generalized algorithm for signed graph partitioning GASP, given a linkage criterium, a type of graph (signed or unsigned) and the optional use of cannot-link constraints. The set $E_{uv}$ is defined as the set of all edges connecting cluster $S_u$ to cluster $S_v$, i.e. $E_{uv} = (S_u \times S_{v \neq u}) \cap E$.

In the special case of an unsigned graph with only positive interactions, i.e. $w_e^- = 0$ and $w_e \geq 0$ $\forall e \in E$, the algorithm performs a standard agglomerative hierarchical clustering by returning only a single cluster and a hierarchy of clusters defined by the order in which the clusters are merged (see Table 1, unsigned graphs).

Given a graph with both attractive and repulsive cues, an edge contraction algorithm with a sum update rule was pioneered in [49, 36] (Table 1, *Sum* linkage). The authors present both a version with cannot-link constraints and one without, and then compare them with other greedy local-search algorithms approximating the multicut optimization problem. The Mutex Watershed [81] is another signed graph partitioning algorithm that introduces dynamical cannot-link constraints. In Proposition 7.1 (see Appendix 7.2) we prove that, surprisingly, it can also be seen as an efficient implementation of GASP with *Absolute maximum* linkage (def. in Table 1). Moreover, in Proposition 7.2 we also prove that GASP with *Abs Max* linkage returns the same clustering with or without enforcing cannot-link constraints. On the other hand, to our knowledge, *Average*, *Max* or *Min* linkage criteria have never been used for signed graph agglomerative algorithms or been combined with cannot-link constraints.

Apart from the linkage criteria defined in Table 1, additional ones were proposed in the literature: [62] for example uses a learned approach where a random forest classifier updates the cluster interactions depending on predefined edge and node features; other approaches introduce a weight regularization depending on the size of the clusters [22, 33], whereas [26] uses a *quantile* linkage criteria by populating a histogram for each inter-cluster interaction. In our experiments, we decided to focus on the linkage criteria listed in Table 1, since they represent the most common options.

## 4 Experiments on neuron segmentation

We first evaluate and compare the agglomerative clustering algorithms described in the generalized framework on the task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuro-science with the goal of reconstructing neural wiring diagrams spanning complete central nervous systems. Currently, only proof-reading or manual tracing yields sufficient accuracy for correct circuit reconstruction [74], thus further progress is required in automated reconstruction methods.

EM segmentation is commonly performed by first predicting boundary pixels [9, 14] or undirected affinities [81, 48, 26], which represent how likely it is for a pair of pixels to belong to the same neuron segment. The affinities do not have to be limited to immediately adjacent pixels. Thus, similarly to [48], we train a CNN to predict both short- and long-range affinities and use them as edge weights of a 3D grid graph, where each node represents a pixel/voxel of the volume image.

### 4.1 Data: CREMI challenge

We evaluate all algorithms in the proposed framework on the competitive CREMI 2016 EM Segmentation Challenge [25] that is currently the neuron segmentation challenge with the largest amount of training data available. The dataset comes from serial section EM of *Drosophila* fruit-fly tissue and consists of 6 volumes of 1250x1250x125 voxels at resolution 4x4x40nm, three of which present publicly available training ground truth. The results submitted to the leaderboard are evaluated using the CREMI score (`https://cremi.org/leaderboard/`), based on the Adapted Rand-Score (Rand-Score) and the Variation of Information Score [4]. In Appendix 7.4, we provide more details about the training of our CNN model, inspired by work of [48, 26].

### 4.2 Results and discussion

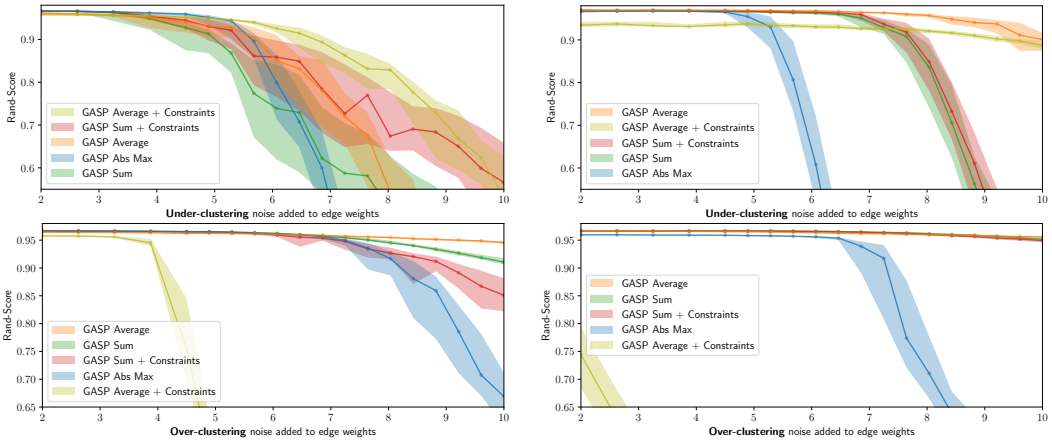
**Comparison of linkage criteria** Table 2 shows how the agglomerative algorithms derived from our framework compare to each other. For a simple baseline, we also include a segmentation produced by thresholding the affinity predictions (THRESH). GASP with *Average* linkage, representing one of the new algorithms derived from our generalized framework, significantly outperformed all other previously proposed agglomerative methods like GAEC (GASP Sum) [36], Greedy Fixation (GASP Sum + Constraints) [49] or Mutex Watershed (GASP Abs. Max.) [81]. The competitive performance of this simple parameter-free algorithm is also reported in Table 3, showing the current leader-board of the challenge: all entries, apart from GASP, employ superpixel-based post-processing pipelines, several of which rely on the lifted multicut formulation of [9] that uses several random forests to

| | CREMI-Score (lower is better) |
|---|---|
| **GASP Average** | **0.226** |
| GASP Sum + Constraints [49] | 0.282 |
| GASP Abs. Max. [81] | 0.322 |
| GASP Max. + Constraints | 0.324 |
| GASP Sum [36] | 0.334 |
| GASP Average + Constraints | 0.563 |
| THRESH | 1.521 |

Table 2: CREMI-Scores achieved by different linkage criteria and thresholding. All methods use the affinity predictions from our CNN as input. Scores are averages over the three CREMI training datasets.

| | CREMI-Score (lower is better) |
|---|---|
| Our CNN + DTWS + LMC | 0.221 |
| PNI CNN [48] | 0.228 |
| **Our CNN + GASP Average** | **0.241** |
| MALA CNN + MC [26] | 0.276 |
| CRU-Net [85] | 0.566 |
| LFC [64] | 0.616 |

Table 3: Current leading entries in the CREMI challenge leaderboard [25] (May 2019). The scores are averages of the three test datasets.




(a) No long-range predictions: $p_{\text{long}} = 0$

(b) With long-range predictions: $p_{\text{long}} = 0.1$

Figure 4: GASP sensitivity to noise: *Average* linkage proved to be the most robust. Performances are given by Rand-Score (higher is better) depending on the amount of noise added to the CNN predictions. Solid lines represent median values over 30 experiments. Values between the 25th and the 75th percentile are shown in shaded areas. The two sets of experiments using under- and over-clustering noise are summarized in the plots at the top and at the bottom, respectively (see Appendix 7.6 for more details). For each experiment, some of the long-range CNN predictions were randomly selected with probability $p_{\text{long}}$ and added as long-range edges to the pixel grid-graph. Experiments are performed on a crop of CREMI training sample B.

predict graph edge weights, relying not only on information derived from affinity maps but also raw data and shape information. Note that the test volumes contain several imaging artifacts that make segmentation particularly challenging and might profit from more robust edge statistics of super-pixel based approaches. On the other hand, the fact that our algorithm can operate on pixels directly removes the parameter tuning necessary to obtain good super-pixels and can also avoid errors that result from wrong superpixels that cannot be fixed during later agglomeration. In Appendix 7.5, we provide more details about how we scaled up GASP to the full datasets. Appendix Table 5 lists the performances and the run-times for all tested GASP linkage.

**Noise experiments** Additionally, we conduct a set of experiments where the CNN predictions are perturbed by structured noise, in order to highlight the properties of each GASP variant and perform an in-depth comparison that is as quantitative as possible. Appendix 7.6 introduces the type of spatially correlated noise that allowed us to perturb the CNN outputs by introducing simulated additional artifacts like missing or false positive boundary evidence. Fig. 4 summarizes our 12000 noise experiments: we focus on the best performing linkage criteria, i.e. *Average*, *Sum* and *Abs Max*, and test them with different amount of noise. In these experiments, we also want to assess how beneficial it is to use long-range CNN predictions in the agglomeration. Thus, we perform a set

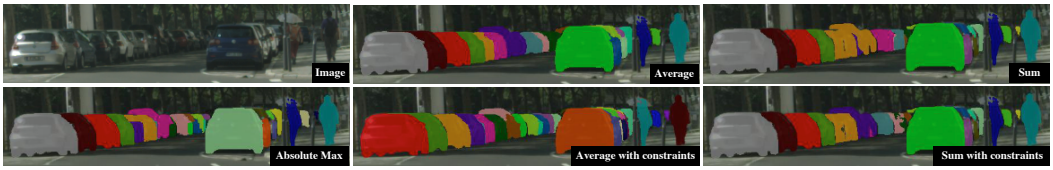



Figure 5: Visual results given by different GASP linkage criteria on a crop of a CityScapes image

of simulations without adding long-range connections to the grid-graph and another set where we introduce them with a 10% probability[2].

**Average and Abs Max linkage** Our findings confirm that GASP with *Average* linkage criterion represents the most robust algorithm tested and the one that benefits the most from using the long-range CNN predictions. On the other hand, it is not a surprise that the *Abs Max* statistic proposed by [81] is less robust to noise than the *Average* linkage, but, as we show in the Appendix Table 5, *Abs Max* represents a valid and considerably faster option. Adding long-range connections to the graph is generally helpful, but when many of them carry repulsive weights, then GASP with cannot-link constraints shows a clear tendency to over-cluster.

**Sum linkage** All our experiments show that GASP with *Sum* linkage is the algorithm with the highest tendency to under-cluster and incorrectly merge segments (see Fig. 3 for an example). This property is related to the empirical observation that a *Sum* statistic tends to grow clusters one after the other, as shown in Fig. 1 by the quite unique agglomeration order of the *Sum* statistic. An intuitive explanation of this fact is the following: initially, most of the intra-cluster nodes present similar attractive interactions between each others; when the two nodes sharing the most attractive interaction are merged, there is a high chance that they both share an attractive interaction with a common neighboring node, so the new interaction with this common neighbor will be immediately assigned to a high priority in the agglomeration, given by the sum of two high weights; this usually starts a "chain reaction", where only a single cluster is agglomerated at the beginning. On the other hand, as we also see in Fig. 1, other linkage criteria like *Average* or *Abs Max* grow clusters of similar sizes in parallel and accumulate in this way much more reliable inter-cluster statistics.

## 5 Experiments on CityScapes

We also evaluate the performances of GASP on the CityScapes dataset [15], which consists of 5000 street-scene images: 2975 for training, 500 for validation and 1525 for testing. We used the pipeline proposed in GMIS [58], representing the proposal-free method performing best on the dataset. The pipeline consists of two neural networks with similar structures, one predicting pixel level semantic scores and the other predicting pixel affinities between instances. The code and the model are publicly available, so we provided its output affinities as input to GASP. In Appendix 7.7 we present how we fine-tuned the model by using a *Søresen-Dice* loss, similarly to [81].

Results are summarized in Table 4 and Fig. 5: the best scores are achieved by PANet [54], which is a proposal-based method strongly related to Mask R-CNN. GASP with *Average* linkage achieves competitive results and outperforms all previously proposed proposal-free methods. Similarly to the experiments on neuron segmentation, other linkage criteria tend to over-cluster, like *Abs Max*, or under-cluster and merge instances, like *Sum*. The graph-merging algorithm proposed by [58] (MultiStepHAC) requires the user to tune several threshold parameters and it was probably tailored to the output affinities of the original model: the graph-merging method did not generalize well to our fine-tuned model and, on the validation set, it achieved an AP score of 33.0, which is lower than the original value 34.1 reported in [58]. Table 6 in Appendix includes the scores of all other tested GASP algorithms.

[2]We also performed experiments adding all the long-range predictions given by the CNN model, but we did not note major differences when using only 10% of them. Adding this fraction is usually sufficient to improve the scores.

| Method | Agglomeration type | AP |
|---|---|---|
| PANet [54] | - | **36.5** |
| Mask R-CNN [30] | - | 31.5 |
| GMIS Model [58] | GASP Average | 34.3 |
| | GASP Average + Constraints | 33.9 |
| | MultiStepHAC [58] | 33.0 |
| | GASP Abs. Max. [81] | 32.1 |
| | GASP Sum + Constraints [49] | 31.9 |
| | GASP Sum [36] | 31.3 |

(a) CityScapes *validation* set

| Method | AP | AP 50% |
|---|---|---|
| PANet [54] | **31.8** | **57.1** |
| **Ours: GMIS Model + GASP Average** | 28.3 | 47.0 |
| GMIS [58] | 27.3 | 45.6 |
| Mask R-CNN [30] | 26.2 | 49.9 |
| SGN [53] | 25.0 | 44.9 |
| DIN [5] | 20.0 | 38.8 |
| DWT [6] | 19.4 | 35.3 |
| InstanceCut [38] | 13.0 | 27.9 |

(b) CityScapes *test* set

Table 4: Average Precision scores (higher is better) on the CityScapes dataset. GASP with *Average* linkage combined with the GMIS Model [58] represents the proposal-free method achieving the best results (May 2019). In order to have a fair comparison, we only compare methods that did not use external data (e.g. COCO [52]) for training.

## 6 Conclusion

We have presented a novel unifying framework for agglomerative clustering of graphs with both positive and negative edge weights and we have shown that several existing clustering algorithms, e.g. the Mutex Watershed [81], can be reformulated as special cases of one underlying agglomerative algorithm. This framework also allowed us to introduce new algorithms, one of which, based on an *Average* linkage criterion, outperformed all the others: It proved to be a simple and remarkably robust approach to process short- and long-range predictions of a CNN applied to an instance segmentation task. On biological images, this simple average agglomeration algorithm can represent a valuable choice for user who is not willing to spend much time tuning complex task-dependent pipelines based on superpixels. In future work we plan to extend the comparison to other types of non-image graphs and explore common theoretical properties of the algorithms included in the framework.

## References


[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23, 2008.

[2] 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.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[4] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015.

[5] A. Arnab and P. H. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 441–450, 2017.

[6] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[7] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

[8] T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht. An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *European Conference on Computer Vision*, pages 715–730. Springer, 2016.

[9] T. Beier, C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101, 2017.

[10] Y.-T. Chen, X. Liu, and M.-H. Yang. Multi-instance object segmentation with occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3470–3478, 2015.

[11] S. Chopra and M. R. Rao. On the multiway cut polyhedron. *Networks*, 21(1):51–89, 1991.

[12] S. Chopra and M. R. Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993.

[13] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[14] 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.

[15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[16] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[17] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[18] 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.

[19] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[20] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[21] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[22] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[23] J. R. Finkel and C. D. Manning. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 45–48. Association for Computational Linguistics, 2008.

[24] J. Funke, F. A. Hamprecht, and C. Zhang. Learning to segment: training hierarchical segmentation under a topological loss. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 268–275. Springer, 2015.

[25] J. Funke, S. Saalfeld, D. Bock, S. Turaga, and E. Perlman. Cremi challenge. `https://cremi.org.`, 2016. Accessed: 2019-05-15.

[26] J. Funke, F. D. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[27] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96, 1989.

[28] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1-3):367–387, 1990.

[29] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[30] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[31] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605, 2018.

[32] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn. Globally optimal image partitioning by multicuts. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44. Springer, 2011.

[33] A. Kardoost and M. Keuper. Solving minimum cost lifted multicut problems by node agglomeration. In *ACCV 2018, 14th Asian Conference on Computer Vision*, Perth, Australia, 2018.

[34] 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.

[35] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[36] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015.

[37] B. R. Kiran and J. Serra. Global–local optimizations by hierarchical cuts and climbing energies. *Pattern Recognition*, 47(1):12–24, 2014.

[38] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017.

[39] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister. Rhoananet pipeline: Dense automatic neural annotation. *arXiv preprint arXiv:1611.06973*, 2016.

[40] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*, 2015.

[41] S. Kong and C. C. Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018.

[42] N. Krasowski, T. Beier, G. W. Knott, U. Koethe, F. A. Hamprecht, and A. Kreshuk. Improving 3d em data segmentation by joint optimization over boundary evidence and biological priors. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 536–539. IEEE, 2015.

[43] V. Kulikov, V. Yurchenko, and V. Lempitsky. Instance segmentation by deep coloring. *arXiv preprint arXiv:1807.10007*, 2018.

[44] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr. What, where and how many? combining object detectors and crfs. In *European conference on computer vision*, pages 424–437. Springer, 2010.

[45] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The computer journal*, 9(4):373–380, 1967.

[46] J.-H. Lange, B. Andres, and P. Swoboda. Combinatorial persistency criteria for multicut and max-cut. *arXiv preprint arXiv:1812.01426*, 2018.

[47] J.-H. Lange, A. Karrenbauer, and B. Andres. Partial optimality and fast lower bounds for weighted correlation clustering. In *International Conference on Machine Learning*, pages 2898–2907, 2018.

[48] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[49] E. Levinkov, A. Kirillov, and B. Andres. A comparative study of local search algorithms for correlation clustering. In *German Conference on Pattern Recognition*, pages 103–114. Springer, 2017.

[50] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[51] X. Liang, Y. Wei, X. Shen, Z. Jie, J. Feng, L. Lin, and S. Yan. Reversible recursive instance-level object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2016.

[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[53] S. Liu, J. Jia, S. Fidler, and R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017.

[54] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[55] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen. A modular hierarchical approach to 3d electron microscopy image segmentation. *Journal of neuroscience methods*, 226:88–102, 2014.

[56] T. Liu, M. Seyedhosseini, and T. Tasdizen. Image segmentation using hierarchical merge tree. *IEEE transactions on image processing*, 25(10):4596–4607, 2016.

[57] T. Liu, M. Zhang, M. Javanmardi, N. Ramesh, and T. Tasdizen. Sshmt: Semi-supervised hierarchical merge tree for electron microscopy image segmentation. In *European Conference on Computer Vision*, pages 144–159. Springer, 2016.

[58] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu. Affinity derivation and graph merge for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–703, 2018.

[59] F. Malmberg, R. Strand, and I. Nyström. Generalized hard constraints for graph segmentation. In *Scandinavian Conference on Image Analysis*, pages 36–47. Springer, 2011.

[60] Y. Meirovitch, A. Matveev, H. Saribekyan, D. Budden, D. Rolnick, G. Odor, S. Knowles-Barley, T. R. Jones, H. Pfister, J. W. Lichtman, et al. A multi-pass approach to large-scale connectomics. *arXiv preprint arXiv:1612.02120*, 2016.

[61] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[62] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

[63] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk. Solving large multicut problems for connectomics via domain decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–10, 2017.

[64] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga, X. Zhang, B. Matejek, L. Kamentsky, J. W. Lichtman, and H. Pfister. Anisotropic em segmentation by 3d affinity learning and agglomeration. *arXiv preprint arXiv:1707.08935*, 2017.

[65] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[66] K. Perlin. Noise hardware. *Real-Time Shading SIGGRAPH Course Notes*, 2001.

[67] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6656–6664, 2017.

[68] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[69] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2013.

[70] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016.

[71] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[72] S. Saalfeld, R. Fetter, A. Cardona, and P. Tomancak. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature methods*, 9(7):717, 2012.

[73] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE transactions on Image Processing*, 9(4):561–576, 2000.

[74] P. Schlegel, M. Costa, and G. S. Jefferis. Learning from connectomics on the fly. *Current opinion in insect science*, 24:96–105, 2017.

[75] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers. Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer, 2018.

[76] T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34, 1948.

[77] S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung. Maximin affinity learning of image segmentation. pages 1865–1873, 2009.

[78] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.

[79] M. G. Uzunbas, C. Chen, and D. Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical image analysis*, 27:31–44, 2016.

[80] S. Wolf, A. Bailoni, C. Pape, N. Rahaman, A. Kreshuk, U. Köthe, and F. A. Hamprecht. The mutex watershed and its objective: Efficient, parameter-free image partitioning. *arXiv preprint arXiv:1904.12654*, 2019.

[81] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Kothe, and F. Hamprecht. The mutex watershed: Efficient, parameter-free image partitioning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 546–562, 2018.

[82] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proc. ICCV'15*, pages 1395–1403, 2015.

[83] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[84] J. Yarkony, A. Ihler, and C. C. Fowlkes. Fast planar correlation clustering for image segmentation. In *European Conference on Computer Vision*, pages 568–581. Springer, 2012.

[85] T. Zeng, B. Wu, and S. Ji. Deepem3d: approaching human-level performance on 3d anisotropic em image segmentation. *Bioinformatics*, 33(16):2555–2562, 2017.

# 7 Supplementary material

**Algorithm 2** Implementation of GASP, generalized algorithm for signed graph partitioning

**Input:** $\mathcal{G}(V, E, w^+, w^-)$ with $N$ nodes and $M$ edges; boolean `addCannotLinkConstraints`
**Output:** Final clustering

$\tilde{\mathcal{G}}(\tilde{V}, \tilde{E}) \leftarrow \mathcal{G}(V, E, w^+, w^-)$ ▷ Init. contracted graph
`UF` $\leftarrow$ initUnionFind($V$) ▷ Init. data structure representing clustering
PQ.push($|w_e|, e$) $\quad \forall e \in E$ ▷ Init. priority queue in desc. order of $|w_e| = |w_e^+ - w_e^-|$, $\mathcal{O}(|E|)$
`canBeMerged`$[e] \leftarrow$ `True` $\quad \forall e \in E$ ▷ Init. cannot-link constraints

**while** PQ is **not** empty **do**
  $\tilde{w}, e_{uv} \leftarrow$ PQ.popHighest() ▷ $\mathcal{O}(\log |E|)$
  **assert** `UF`.find($u$) $\neq$ `UF`.find($v$) ▷ Edges in PQ always link nodes in different clusters
  **if** ($\tilde{w} > 0$) **and** `canBeMerged`$[e_{uv}]$ **then**
    PQ, `canBeMerged`, $\tilde{E} \leftarrow$ UPDATENEIGHBORS($u, v$)
    $\tilde{V} \leftarrow \tilde{V} \setminus \{v\}, \quad \tilde{E} \leftarrow \tilde{E} \setminus \{e_{uv}\}$ ▷ Update contracted graph
    `UF`.merge($u, v$) ▷ Merge clusters, $\mathcal{O}(\alpha(|E|))$
  **else if** ($\tilde{w} \leq 0$) **and** `addCannotLinkConstraints` **then**
    `canBeMerged`$[e_{uv}] \leftarrow$ `False` ▷ Constrain the two clusters
**return** Final clustering given by union-find data structure `UF`

**function** UPDATENEIGHBORS($u, v$)
  $\mathcal{N}_u = \{t \in \tilde{V} | e_{ut} \in \tilde{E}\}$
  $\mathcal{N}_v = \{t \in \tilde{V} | e_{vt} \in \tilde{E}\}$
  **for** $t \in \mathcal{N}_v$ **do** ▷ Loop over neighbors in $\tilde{\mathcal{G}}$ of deleted node $v$
    $\tilde{E} \leftarrow \tilde{E} \setminus \{e_{vt}\}$
    $\tilde{w}_{vt} \leftarrow$ PQ.delete($e_{vt}$) ▷ $\mathcal{O}(\log |E|)$
    `canBeMerged`$[e_{ut}] \leftarrow$ `canBeMerged`$[e_{ut}]$ **and** `canBeMerged`$[e_{vt}]$
    **if** $t \in \mathcal{N}_u$ **then** ▷ $t$ is a common neighbor of $u$ and $v$
      $\tilde{w}_{ut} \leftarrow$ PQ.delete($e_{ut}$) ▷ $\mathcal{O}(\log |E|)$
      PQ.push($|f(\tilde{w}_{ut}, \tilde{w}_{vt})|, e_{ut}$) ▷ $\mathcal{O}(\log |E|)$
    **else**
      $\tilde{E} \leftarrow \tilde{E} \cup \{e_{ut}\}$
      PQ.push($|\tilde{w}_{vt}|, e_{ut}$) ▷ $\mathcal{O}(\log |E|)$
  **return** PQ, `canBeMerged`, $\tilde{E}$

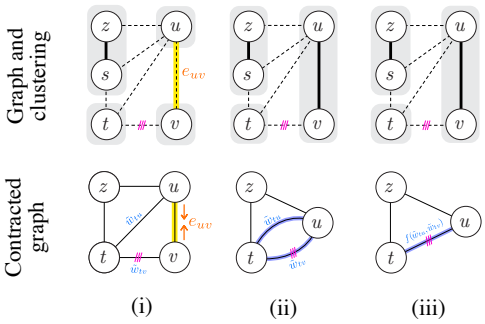
## 7.1 Implementation details and complexity of GASP

**Update rules** During the agglomerative process, the interaction between adjacent clusters has to be properly updated and recomputed, as shown in Algorithm 1. An efficient way of implementing these updates can be achieved by representing the agglomeration as a sequence of *edge contractions* in the graph. Given a graph $\mathcal{G}(V, E, w)$ and a clustering $\Pi$, we define the associated *contracted graph* $\tilde{\mathcal{G}}_\Pi(\tilde{V}, \tilde{E}, \tilde{w})$, such that there exists exactly one representative $|\tilde{V} \cap S| = 1$ for every cluster $S \in \Pi$ . Edges in $\tilde{E}$ represent adjacency-relationships between clusters and the signed edge weights $\tilde{w}_e$ are given by inter-cluster interactions $\tilde{w}(e_{uv}) = \mathcal{W}_{S_u, S_v}$. For the linkage criteria tested in this work, when two clusters $S_u$ and $S_v$ are merged, the interactions between the new cluster $S_u \cup S_v$ and each of its neighbors depend only on the previous interactions involving $S_u$ and $S_v$. Thus, we can recompute these interactions by using an *update rule* $f$ that does not involve any loop over the edges of the original graph $\mathcal{G}$:

$$\mathcal{W}(S_u \cup S_v, S_t) = f\Big[\mathcal{W}(S_u, S_t), \mathcal{W}(S_v, S_t)\Big] = f(\tilde{w}(e_{ut}), \tilde{w}(e_{vt})) \tag{1}$$

In Fig. 6 we show an example of edge contraction and we list the update rules associated to the linkage criteria we introduced in Table 1.

**Implementation** As we show in Algorithm 2, our implementation of GASP is based on an union-find data structure and a heap allowing deletion of its elements. The algorithm starts with each node




| Linkage criteria | Update rule $f$ |
|---|---|
| Sum: | $f(\tilde{w}_1, \tilde{w}_2) = \tilde{w}_1 + \tilde{w}_2$ |
| Absolute Maximum: | $f(\tilde{w}_1, \tilde{w}_2) = \begin{cases} \tilde{w}_1 & \text{if } \lvert\tilde{w}_1\rvert > \lvert\tilde{w}_2\rvert \\ \tilde{w}_2 & \text{otherwise} \end{cases}$ |
| Average: | $f(\tilde{w}_1, \tilde{w}_2) = \text{weightAvg}\{\tilde{w}_1, \tilde{w}_2\}$ |
| Maximum: | $f(\tilde{w}_1, \tilde{w}_2) = \max\{\tilde{w}_1, \tilde{w}_2\}$ |
| Minimum: | $f(\tilde{w}_1, \tilde{w}_2) = \min\{\tilde{w}_1, \tilde{w}_2\}$ |

Figure 6: **Left:** Example of edge contraction. First row: original graph $\mathcal{G}$; clustering $\Pi$ (gray shaded areas) with dashed edges on cut; cannot-link constraints (violet bars). Second row: contracted graph $\tilde{\mathcal{G}}_\Pi$. In step ii), edge $e_{uv}$ is contracted and node $v$ deleted from $\tilde{\mathcal{G}}_\Pi$. In step iii), double edges $e_{tu}$ and $e_{tv}$ resulting from the edge contraction are replaced by a single edge with updated interaction. **Right:** The table lists the update rules $f(\tilde{w}_1, \tilde{w}_2)$ associated to the linkage criteria of Table 1 and that are used to efficiently update the interactions between clusters.

assigned to its own cluster and sorts all edges $e \in E$ in a heap/priority queue (PQ) by their absolute weight $|w_e| = |w_e^+ - w_e^-|$ in descending order, so that the most attractive and the most repulsive interactions are processed first. It then iteratively pops one edge $e_{uv}$ from PQ and, depending on the priority $\tilde{w}_{uv}$, does the following: in case of attractive interaction $\tilde{w}_{uv} > 0$, provided that $e_{uv}$ was not flagged as a cannot-link constraint, then merge the connected clusters, perform an edge contraction of $e_{uv}$ in $\tilde{\mathcal{G}}_\Pi$ and update the priorities of new double edges as explained in Fig. 6. If, on the other hand, the interaction is repulsive ($\tilde{w}_{uv} \leq 0$) and the option `addCannotLinkContraints` of Alg. 2 is `True`, then the edge $e_{uv}$ is flagged as cannot-link constraint.

**Complexity** In the main loop, the algorithm iterates over all edges, but the only iterations presenting a complexity different from $\mathcal{O}(1)$ are the ones involving a merge of two clusters, which are at most $N - 1$. By using a union-find data structure (with path compression and union by rank) the time complexity of `merge`$(u, v)$ and `find`$(u)$ operations is $\mathcal{O}(\alpha(N))$, where $\alpha$ is the slowly growing inverse Ackerman function. The algorithm then iterates over the neighbors of the merged cluster (at most $N$) and updates/deletes values in the priority queue ($\mathcal{O}(\log |E|)$). Therefore, similarly to a heap-based implementation of hierarchical agglomerative clustering, our implementation of GASP has a complexity of $\mathcal{O}(N^2 \log N)$. In the worst case, when the graph is dense and $|E| = N^2$, the algorithm requires $\mathcal{O}(N^2)$ memory. Nevertheless, in our practical applications the graph is much sparser, so $\mathcal{O}(|E|) = \mathcal{O}(N)$. With a single-linkage, corresponding to the choice of the *Maximum* update rule in our framework, the algorithm can be clearly implemented by using the more efficient Kruskal's Minimum Spanning Tree algorithm with complexity $\mathcal{O}(N \log N)$. Moreover, in the next section, we present an efficient implementation of GASP with *Absolute Maximum* linkage that has empirical $\mathcal{O}(N \log N)$ complexity.

**Algorithm 3** Mutex Watershed Algorithm proposed by [81]

**Input:** $\mathcal{G}(V, E, w^+, w^-)$ with $N$ nodes and $M$ edges
**Output:** Final clustering

`UF` $\leftarrow$ initUnionFind($V$)
**for** $(u, v) = e \in E$ in descending order of $|w_e| = |w_e^+ - w_e^-|$ **do**
  **if** `UF`.find($u$) $\neq$ `UF`.find($v$) **then** ▷ Check if $u, v$ are already in the same cluster
    **if** ($w_e > 0$) **and** `canBeMerged`$(u, v)$ **then** ▷ Check for cannot-link constraints
      `UF`.merge($u, v$) and inherit constraints of parent clusters
    **else if** ($w_e \leq 0$) **then**
      Add cannot-link constraints between parent clusters of $u, v$
**return** Final clustering given by union-find data structure `UF`

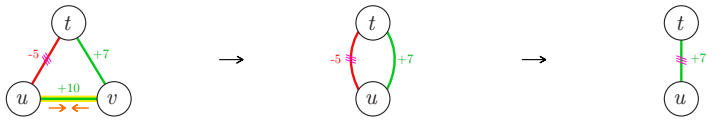



Figure 7: GASP with *AbsMax* linkage: Example representing the only case of edge contraction $e_{uv}$ that would introduce a positive attractive interaction between two constrained clusters. Note this can actually never happen with an *AbsMax* linkage, because edge $e_{ut}$ has a lower absolute priority as compared to $e_{uv}$, so clusters $u$ and $t$ cannot have been constrained before $u$ and $v$ are merged.

### 7.2 Properties of GASP with *Absolute Maximum* linkage

**Remark on graph notation** The definition of a graph proposed by [81] makes a distinction between a set of positive edges $E^+$, associated with a set $W^+$ of positive scalar attributes representing merge affinities, and a set of negative edges $E^-$, associated with a set $W^-$ of positive attributes representing split tendencies. On the other hand, in our definition $\mathcal{G}(V, E, w^+, w^-)$ each edge have both an attractive $w_e^+$ and a repulsive $w_e^-$ attribute, so we can make them equivalent by defining:

$$E^+ = \{e \in E \text{ s.t. } w_e = w_e^+ - w_e^- > 0\}, \qquad E^- = \{e \in E \text{ s.t. } w_e = w_e^+ - w_e^- \leq 0\} \tag{2}$$

$$W^+ = \{|w_e| \text{ s.t. } e \in E^+\}, \qquad W^- = \{|w_e| \text{ s.t. } e \in E^-\} \tag{3}$$

**Proposition 7.1.** *The Mutex Watershed Algorithm 3 (MWS) with empirical $\mathcal{O}(N \log N)$ complexity introduced by [81] returns the same final clustering given by the GASP Algorithm 2 with the use of cannot-link constraints and an Absolute Maximum update rule:*

$$f_{\text{Abs.Max.}}(\tilde{w}_1, \tilde{w}_2) = \begin{cases} \tilde{w}_1 & \textit{if } |\tilde{w}_1| > |\tilde{w}_2| \\ \tilde{w}_2 & \textit{otherwise} \end{cases} \tag{4}$$

*Proof.* Both algorithms sort edges in descending order of the absolute interactions $|w_e|$ and then iterate over all of them. The only difference is that MWS, after merging two clusters, does not update the interactions between the new cluster and its neighbors. However, since with an Abs. Max. linkage the interaction between clusters is simply given by the edge with highest absolute weight $|w_e|$, the order by which edges are iterated over in GASP is never updated. Thus, both algorithms perform precisely the same steps and return the same clustering. □

**Proposition 7.2.** *The GASP Algorithm 2 with the Absolute Maximum linkage defined in Eq. 4 returns the same final clustering whether or not cannot-link constraints are enforced.*

*Proof.* In the GASP Algorithm 2, the clustering is updated only when two clusters are merged and the condition at line 9 is satisfied. We also observe that, in the unconstrained version of GASP, the predicate `canBeMerged` at line 9 can never be false because cannot-link constraints are never introduced at line 14. Let us now contradict the initial hypothesis and assume by absurd that the constrained version of GASP introduces a cannot-link constraints between two clusters sharing a positive interaction $\tilde{w} > 0$ and outputs a different clustering as compared to the unconstrained version. This can happen only in the situation shown in Fig. 7, when two clusters $u$ and $v$ are merged together and share a common neighboring node $t$ having the following two properties: a) $u$ and $t$ are already constrained and share a repulsive interaction $w_{ut} \leq 0$, b) $v$ and $t$ share an attractive interaction $w_{vt} > 0$ that is higher in absolute value $|w_{vt}| > |w_{ut}|$. Then, according to Eq. 4, the new merged cluster $uv$ and $t$ are constrained and share a positive interaction. But this case can never happen, since if $|w_{vt}| > |w_{ut}|$ then clusters $v$ and $t$ are merged before clusters $u$ and $t$ are constrained. □

### 7.3 Predicting signed edge weights with a CNN

Our CNN model outputs affinities in the form of pseudo-probabilities $p : E \to [0, 1]$, where $p = 0$ represents a boundary evidence. In order to use them as input of the algorithms in our framework,

we mapped them to positive and negative values[3]. The most common approaches use *additive* [1] or *logarithmic* [23, 2] mappings:

$$w_{e,\text{Add}} = p_e - \beta, \qquad w_{e,\text{Log}} = \log\left(\frac{p_e}{1-p_e}\right) - \log\left(\frac{\beta}{1-\beta}\right), \tag{5}$$

where $\beta \in [0,1]$ is a *bias* parameter that allow a tuning between over- and under-segmentation. We evaluated both of them empirically with each of the tested linkage and found that the additive mapping is the best option in all cases apart from the *Sum* linkage. Note that varying the parameter $\beta$ does not usually define a hierarchy of nested clusterings, thus it is not equivalent to varying a threshold parameter in HAC. This hierarchical property is only valid for GASP without constraints and with *Average*, *Max* or *Min* linkage.

### 7.4 Neuron segmentation and compared methods

**Training details** The data from the CREMI challenge is highly anisotropic and contains artifacts like missing sections, staining precipitations and support film folds. To alleviate difficulties stemming from misalignment, we use a version of the data that was elastically realigned by the challenge organizers with the method of [72]. We train a 3D U-Net [71, 13] using the same architecture as [26] and predict long-and-short range affinities as described in [48]. In addition to the standard data augmentation techniques of random rotations, random flips and elastic deformations, we simulate data artifacts. In more detail, we randomly zero-out slices, decrease the contrast of slices, simulate tears, introduce alignment jitter and paste artifacts extracted from the training data. Both [26] and [48] have shown that these kinds of augmentations can help to alleviate issues caused by EM-imaging artifacts. We use L2 loss and Adam optimizer to train the network. The model was trained on all the three samples with available ground truth labels.

**THRESH and WSDT** The basic post-processing methods we consider cannot take long-range affinities into account, so we only consider direct neighbors affinities and generate a boundary map by taking an average over the 3 directions. Based on this boundary map, we run connected components (THRESH) and we also run a watershed algorithm seeded at the maxima of the smoothed distance transform (WSDT). For WSDT, the degree of smoothing was optimized such that each region receives as few seeds as possible, without however causing severe under-segmentation. Due to the anisotropy of the data, we generate 2D WSDT superpixels by considering each 2D image in the stack singularly.

**Multi-step pipelines** Given the 2D WSDT superpixels, we build a 3D region-adjacency graph such that each node represents a superpixel. The weights of the edges connecting neighboring superpixels are computed by taking an average over both short- and long-range affinities connecting the two regions. We then convert the edge probabilities to signed weights using the logarithmic mapping defined in Eq. 5 and solve the multicut problem on this graph. For our experiments, we use the approximate Kernighan-Lin solver [36, 35] (WSDT+MC). In some cases, the long-range affinities predicted by the CNN can connect two superpixels that are not direct-neighbors. Thus, in these cases we introduce additional *lifted* edges in the graph and an instance of the lifted multicut problem (WSDT+LMC). This time, similarly to the methods mentioned in [8], we used a combination of approximate solvers consisting in GAEC and Kernighan-Lin.

### 7.5 GASP on the full CREMI dataset

**Pre-merge processing** For the predictions on the full dataset from the CREMI challenge, we used the padded volumes provided by the challenge. The crops on which we performed a prediction have a size of $1500 \times 1500 \times 127 = 2.86 \cdot 10^8$ voxels or larger. Building a graph with $10^8$ nodes can easily incur a large use of memory, so we decided to perform a preprocessing step by initially merging some nodes together. Simply down-sampling the predictions of the CNN would have led to a loss of resolution and performances in the most difficult parts of the dataset. Thus, we decided to pre-merge the most connected components of the graph that would be anyway clustered during the first iterations of GASP. To do this, we used a simple approach: we generated a boundary probability map by taking for each voxel an average over affinities in all directions (both short- and long-range ones) and we

[3]Note that in general attractive and repulsive interactions $w^+$ and $w^-$ can be independently estimated with different classifiers.

| GASP linkage | CREMI-Score (higher better) | Rand-Score (higher better) | VI-merge (lower better) | VI-split (lower better) | Runtime (lower better) |
|---|---|---|---|---|---|
| Average | **0.226** | **0.936** | 0.315 | 0.494 | $3.49 \cdot 10^4$ |
| Sum + CLC [49] | 0.282 | 0.906 | 0.358 | 0.510 | $4.64 \cdot 10^4$ |
| Abs Max [81] | 0.322 | 0.897 | 0.286 | 0.735 | $1.24 \cdot 10^4$ |
| Max + CLC | 0.324 | 0.893 | 0.292 | 0.698 | $6.31 \cdot 10^4$ |
| Sum [36] | 0.334 | 0.872 | 0.461 | 0.444 | $4.74 \cdot 10^4$ |
| Average + CLC | 0.563 | 0.772 | 0.259 | 1.142 | $2.95 \cdot 10^4$ |
| Min | 2.522 | 0.030 | **0.197** | 6.365 | $2.97 \cdot 10^3$ |
| Min + CLC | 2.522 | 0.030 | **0.197** | 6.365 | $4.77 \cdot 10^3$ |
| Max | 2.626 | 0.028 | 7.069 | **0.026** | $\mathbf{6.04 \cdot 10^2}$ |

Table 5: Performances achieved by different versions of GASP on the CREMI 2016 training set. CREMI-Score [25], is given by a combination of the Adapted Rand-Score (Rand-Score) and the Variation of Information Score for under-clustering (VI-merge) and over-clustering (VI-split) [4]. CLC stands for cannot-link constraints. For all algorithms, the chosen value of bias parameter was $\beta = 0$. We used a machine with CPU Intel(R) Xeon(R) E7-4870 @ 2.40GHz for our comparison experiments.

run THRESH with a conservative threshold parameter to find the connected components. With this approach, pixels are pre-clustered only when they are far away (in all directions) from predicted boundaries. To make sure that in this preprocessing step different neurons are never merged together by mistake, we intersected the segments given by THRESH with the segments given by WSDT. We tested GASP both on the full grid-graph and on this preprocessed graph and we did not notice any major differences in the final clustering or achieved scores, although the version with a pre-processed graph was significantly faster. To reduce the runtime and memory requirements even further, we used only 10 % of the long-range connections in the pixel-graph, since adding all of them did not improve the scores.

**Removing small segments** After running GASP, we use a simple post-processing step to delete small segments on the boundaries, most of which are given by single-voxel clusters. On the neuron segmentation predictions, we deleted all regions with less than 200 voxels and used a seeded watershed algorithm to expand the bigger segments.

**Enforcing local merge** In 2D images of urban-scenes, due to partial occlusion, one object instance can be given by multiple components that are not directly connected in the image plane. This is not the case in neuron-segmentation, where each neuron should be given by a single 3D connected component in the volume. In order to enforce it, we modified the implementation of GASP so that two clusters are merged only when they represent two adjacent supervoxels in the 3D volume and if this condition is not satisfied, the merge is postponed until there is a direct connection. This then avoids the introduction of "air-bridges" between segments due to attractive long-range connections in the initial voxel grid-graph. This approach achieved superior performances to the one proposed in [81], where all long-range connections in the grid-graph are associated to a negative repulsive edge weight.

### 7.6 GASP sensitivity to noise: adding artifacts to CNN predictions

Additionally to the comparison on the full training dataset, we performed more experiments on a crop of the more challenging CREMI training sample B, where we perturbed the predictions of the CNN with noise and we introduced additional artifacts like missing or fictitious boundary evidences.

In the field of image processing there are several ways of adding noise to an image, among which the most common are Gaussian noise or Poisson shot noise. In these cases, the noise of one pixel does not correlate with its neighboring noise values. On the other hand, predictions of a CNN are known to be spatially correlated. Thus, we used Perlin noise[4], one of the most common gradient noises used

[4]In our experiments, we used an open-source implementation of simplex noise [66], which is an improved version of Perlin noise [65]

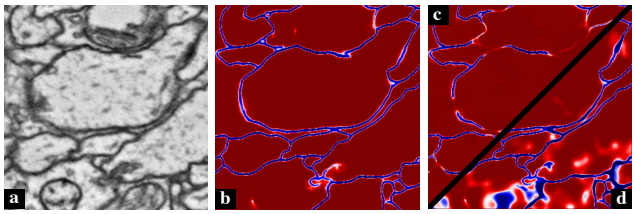



Figure 8: CNN predictions on a slice of the CREMI neuron segmentation challenge with and without additional noise. (**a**) Raw data (**b**) Original CNN predictions $F(x)$, where blue pixels represent boundary evidence (**c**) Under-clustering biased version $\tilde{F}_+(x;\mathcal{K})$ of the predictions defined in Eq. 6 with $\mathcal{K} = 8$ (**d**) Over-clustering biased version $\tilde{F}_-(x;\mathcal{K})$. Long-range predictions are not shown.

| GASP linkage | AP | Bias $\beta$ |
|---|---|---|
| Average | 34.3 | 0.35 |
| Average + CLC | 33.9 | 0.25 |
| Max + CLC | 32.5 | 0.50 |
| Abs Max | 32.1 | 0.45 |
| Sum + CLC | 31.9 | 0.55 |
| Sum | 31.3 | 0.55 |
| Max | 24.3 | 0.85 |
| Min | 0.00 | 0.50 |
| Min + CLC | 0.00 | 0.50 |

Table 6: Average Precision (AP) scores achieved by different versions of GASP and chosen bias parameters $\beta$ on the cityscapes validation set. A bias value $\beta = 0$ returns one single cluster. CLC stands for cannot-link constraints

in procedural pattern generation. This type of noise $n(x) \in [0, 1]$ generates spatial random patterns that are locally smooth but have large and diverse variations on bigger scales. We then combined it with the CNN predictions $p(x)$ in the two following ways:

$$\tilde{F}_{\pm}(x;\mathcal{K}) = F(x) \pm \big|\mathcal{K} \cdot \max\left(\pm N(x), 0\right)\big|, \tag{6}$$

where $N(x) = \text{Logit}[n(x)]$; $F(x) = \text{Logit}[p(x)]$ and $\mathcal{K} \in \mathbb{R}^+$ is a positive factor representing the amount of added noise. $\tilde{F}_+(x;\mathcal{K})$ represents then a under-clustering biased prediction, such that the probability for two pixels to be in the same cluster is increased only if $N(x) > 0$ (see Fig. 8b), whereas $\tilde{F}_-(x;\mathcal{K})$ is a over-clustering biased prediction with decreased probabilities when $N(x) < 0$ (Fig. 8c). In the implementation we used, the noise can be generated in an arbitrary number of dimensions and a smoothing factor can be specified for each direction independently. In our experiments, each pixel is represented by a node in the grid-graph and it is linked to $n_{\text{nb}}$ other nodes by short- and long-range edges. Thus, the output of our CNN model has $n_{\text{nb}}$ channels: for each pixel / voxel, it outputs $n_{\text{nb}}$ values representing the weights of different edge connections. We then generated a 4-dimensional noise that matches the dimension of the CNN output. The data is highly anisotropic, i.e. it has a lower resolution in one of the dimensions. Due to this fact, we chose different smoothing parameters to generate the noise in different directions.

The experiments summarized in Fig. 4 were performed in the following way: for each value $\mathcal{K}$, 30 random noise samples were drawn, from which median and percentiles statistics were computed for each different linkage criteria. For each sample, we randomly selected some of the long-range predictions from the CNN and added them to pixel grid-graph.

### 7.7 Fine-tuning the GMIS pipeline on CityScapes

For our experiments, we used the model from GMIS [58] that is publicly available. The model consists of two neural networks with similar structures, one predicting pixel level semantic scores and the other predicting pixel affinities between instances. We also used all the affinity post-processing methods proposed in [58], e.g. excluding background, resizing regions of interest or the proposed "affinity-refinement" method, which combines semantic and instance outputs. The instance-branch of the model was trained with a Binary Cross-Entropy loss, but we noticed how the short-range affinities were biased towards high probabilities, so that a strong short-range boundary evidence was never predicted by the model. In [58], they handle this problem by proposing a modified version of HAC that is done in stages (MultiStepHAC): initially only short-range affinities are used to run HAC and a low threshold in the hierarchy is chosen to define a first clustering; then a new HAC problem including long-range affinities is initialized with the first clustering; in the method proposed by [58], these steps are repeated three times.

Since MultiStepHAC is a rather complex post-processing method that requires to tune several hyper-parameters, we opted for a different approach to solve the problem of the unbalanced affinities.

We added two 1x1 convolutional layers to the instance-branch model and trained them by using the same loss used by [81] and is based on the Søresen-Dice coefficient [19, 76]. Compared to Hamming-distance based loss like Binary Cross-Entropy or Mean Squared Error, the advantage of this loss is its being robust against prediction and / or target sparsity, that is a desirable quality in this application since boundaries between instances can be sparse. During training, all the affinities involving at least one pixel belonging to the background were ignored in the loss. In this way, these last two layers specialized in improving the predictions of boundary evidence between adjacent instances (especially those belonging to the same class). We then considered an average of these new fine-tuned affinities with the ones predicted by the original model. During the fine-tuning process, only the parameters of the last two convolutional layers were updated.

Before to apply GASP, we performed a parameter-search for the bias $\beta$ defined in 5. Table 6 lists the best-case performances for each of the methods: note that depending on the GASP linkage criterion, it was necessary to bias more or less the predicted edge weights.

The semantic categories are assigned to each instance in the same way proposed by [58], i.e. with a majority vote based on the semantic output of the model.