

# E4 project : Maximin Affinity Learning of Image Segmentation

Quentin Garrido, Tiphane Lamy Verdin, Josselin Lefèvre, Annie Lim

January 2020

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Theoretical background</b>	<b>4</b>
<b>3</b>	<b>Maximin Affinity learning</b>	<b>5</b>
3.1	Method's presentation . . . . .	5
3.2	Optimizing the Rand Index . . . . .	5
3.3	Maximin affinity and the maximin edge . . . . .	6
3.4	Computing an affinity graph . . . . .	6
3.5	Training a classifier . . . . .	7
<b>4</b>	<b>Implementation</b>	<b>8</b>
4.1	Computing the affinity graph . . . . .	8
4.2	Training the neural network . . . . .	8
<b>5</b>	<b>Results</b>	<b>9</b>
5.1	Evaluation method . . . . .	9
5.2	CREMI . . . . .	9
5.3	ISBI 2012 . . . . .	9
<b>6</b>	<b>Going forward</b>	<b>10</b>
<b>7</b>	<b>Teamwork</b>	<b>11</b>
7.1	Team organization . . . . .	11
7.2	Task distribution . . . . .	11
7.3	Obstacles and overcoming them . . . . .	11
<b>8</b>	<b>Conclusion</b>	<b>12</b>

# **1 Introduction**

## 2 Theoretical background

### 3 Maximin Affinity learning

#### 3.1 Method's presentation

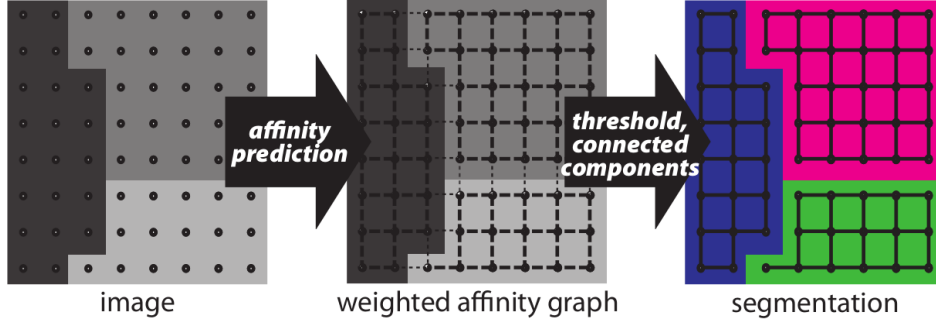


Figure 1: Description of the method, from [?]

The method, as illustrated in figure 1 works in two steps.

The first step is to compute an affinity graph  $G$ , with nodes  $V$  representing each pixel in the original image and edges  $E$  weighted by the affinity between neighbouring pixels. The graph  $G$  will be 4-connected and for every pair of pixels  $i, j \in V \times V$  there exist an edge  $(i, j) \in E$  iff  $i$  and  $j$  are neighbours. We will note the affinity between neighbouring pixels  $A_{ij}$ .

An affinity of 1 between  $i$  and  $j$  means that they belong to the same object, and an affinity of 0 means that they belong to different objects.

Once we have our affinity graph, with affinities between 0 and 1, we will threshold it and remove edges under a certain affinity to obtain connected components that will be our objects, and this thresholded affinity graph will be our final segmentation.

#### 3.2 Optimizing the Rand Index

There exist various methods of evaluating an image segmentation, one of them being the Rand Index which is defined as follow :

$$1 - RI(\hat{S}, S) = \binom{N}{2}^{-1} \sum_{i < j} |\delta(s_i, s_j) - \delta(\hat{s}_i, \hat{s}_j)|$$

With  $S$  our groundtruth segmentation  $\hat{S}$  our predicted segmentation and  $\delta(s_i, s_j)$  the indicator function taking value 1 if  $s_i = s_j$  (if pixels  $i$  and  $j$  are in the same segment/object) and 0 otherwise.

Intuitively, this can be seen as the fraction of image pixels where both segmentations agree.

This gives us a way to evaluate an image segmentation that penalizes when two objects are merged or split in our final segmentation as we can see in figure 2. On the left, only one edge is misclassified but this merges two objects and will be heavily penalized by the Rand Index (we would see the same results if an object was split). On the contrary if we have misclassified edges inside of objects that do not create any splits, this will not be penalized at all by the loss since our objects are still correctly delimited.

As such the Rand Index seems a good measure of segmentation quality for our problem (edge-based segmentation). We will have to see how exactly can we optimise this metric in our case.

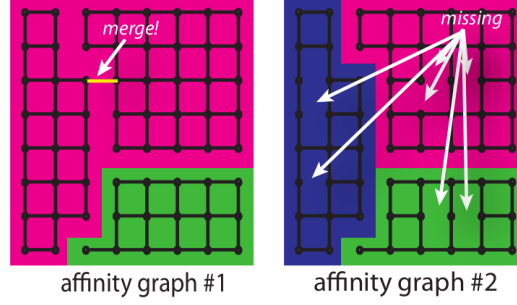


Figure 2: Example image segmentations and their affinity graph. On the left we have a merging of two objects due to a misclassified edge and on the right we have edges that should be of value 1 that have been removed. From [?]

### 3.3 Maximin affinity and the maximin edge

In order to optimize the Rand Index, we must find a way to compute  $\delta(\hat{s}_i, \hat{s}_j)$ . To do this we will use the concept of maximin affinity, as described in [?].

Let  $\mathcal{P}_{ij}$  the set of all paths between  $i$  and  $j$  in our image. For every path  $P \in \mathcal{P}_{ij}$  there is an edge(s)  $(k, l)$  with minimal affinity.

This allows us to define the maximin path, which is the path which maximizes the minimal affinity. It is defined as follow :

$$P_{ij}^* = \arg \max_{P \in \mathcal{P}_{ij}} \min_{(k,l) \in P} A_{kl}$$

The edge of minimal affinity in the maximin path will be called the maximin edge and will be written  $mm(i, j)$ .

It allows to define the maximin affinity which is simply the affinity of the maximin edge.

$$A_{ij}^* = \max_{P \in \mathcal{P}_{ij}} \min_{(k,l) \in P} A_{kl}$$

The most important consequence of this is that a pair of pixels  $i, j$  is connected in the thresholded affinity graph iff  $A_{ij}^*$  is greater than the threshold value, as shown in [?].

This means that if we can find the maximin affinity efficiently, we will be able to compute the Rand Index efficiently.

In practice the maximin affinities can be computed efficiently using a maximum spanning tree (MST), since any path in the MST is a maximin path.

### 3.4 Computing an affinity graph

As we can see, once we have our affinity graph, obtaining the segmentation is straightforward, but the issue is : How to obtain an affinity graph?

As stated before, an affinity graph is equivalent to the contours in the image, and as such any method to find the contours in an image can be used to find the affinity graph (with relative success).

The first idea that we could have would be to use a contour detector, such as the Canny filter for example, however this would lead to an obvious over segmentation, which is not desirable here.

Another idea (the idea used in [?]) would be to use a neural network to obtain our affinity graph. Neural networks, and particularly convolutional neural networks (CNNs) have proven to be an extremely powerful tool in image processing and especially in image classification and segmentation, which makes

them a good candidate for MALIS.

The architecture used originally is a fully convolutional neural network (FCNN) which is a CNN with only convolutional layers. As such they can work with various input sizes which is always a nice feature.

Now that we have seen how we can compute an affinity graph, let's see how we can train this classifier, and if it even possible to train it with the Rand Index as a loss function.

### 3.5 Training a classifier

As stated before, the goal of this method is to optimise the Rand Index, defined as:

$$1 - RI(\hat{S}, S) = \binom{N}{2}^{-1} \sum_{i < j} |\delta(s_i, s_j) - \delta(\hat{s}_i, \hat{s}_j)|$$

with  $S$  our groundtruth and  $\hat{S}$  our predicted segmentation.

We can define  $A_{ij}^*(I, \theta)$  the maximin affinity of pixels  $i$  and  $j$  in the predicted affinity graph of our classifier on  $I$  with parameters  $\theta$ . We can rewrite the previous equation as :

$$1 - RI(I, \theta, S) = \binom{N}{2}^{-1} \sum_{i < j} |\delta(s_i, s_j) - A_{ij}^*(I, \theta)|$$

However this function is not differentiable everywhere so we cannot use it directly for our training. We will replace the absolute value with a smooth loss function  $l$  such as the mean squared error. This relaxation will allow us to use gradient descent to train our classifier (refer to [?] for more details on this).

Thus our final loss function will be :

$$L(I, \theta, S) = \binom{N}{2}^{-1} \sum_{i < j} l(\delta(s_i, s_j), A_{ij}^*(I, \theta)) = \binom{N}{2}^{-1} \sum_{i < j} l(\delta(s_i, s_j), A_{mm(i,j)}(I, \theta))$$

We can now look at what the training loop will look like.

---

**Algorithm 1** MALIS training loop, from [?]

---

- 1: **repeat**
- 2:   Predict the affinity graph for an image  $I$
- 3:   Pick two random pixels  $i, j$  from  $I$
- 4:   Find the maximin edge  $mm(i, j)$
- 5:   Update our parameters  $\theta$  with gradient

$$\nabla l(\delta(s_i, s_j), A_{mm(i,j)}(I, \theta))$$

- 6: **until** convergence is reached
- 

In practice we won't train on the whole image since it would be too long to compute the maximin edge in the whole image. We will instead train on 21x21 patches of the image. We have to choose these patches carefully as most of them will mostly be in an object and not around a border.

Since borders between objects are far less common than the "inside" of an object, we may most of the time train only for pixels inside a same object.

This would result in class imbalance between our "borders" (affinity of value 0) and or "inside" (affinity of value 1). Training with such imbalance would lead to worsened results since borders would not be present most of the time in the chosen image. (one example of such pathological behaviour would be the prediction of affinity 1 everywhere, far from what we desire)

As such when training we need to be careful about what training image we select.

## **4 Implementation**

### **4.1 Computing the affinity graph**

### **4.2 Training the neural network**



## 5 Results

### 5.1 Evaluation method

We evaluated our method with two different datasets : CREMI and ISBI datasets. Both are composed of *Drosophila melanogaster* adult brain images.

Our 3D image of size  $X \times Y \times Z$  was easier to predict in two-dimension. That's why we considered  $Z$  images of size  $X \times Y$  which were stacked together to get back a 3D image.

A very simple architecture is used for the training. It's composed of 6 layers of convolution. In ??, the original architecture had 4 layers with 5 features. We decided to add more layers and features to obtain better results. Batch normalisations were also added as it didn't exist before.

Layer	Kernel	Strides	Features	BN	Activation	Output shape
Input						(21, 21, 1)
Convolution	(5, 5)	(1, 1)	8	Y	ReLU	(21, 21, 8)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	8	Y	ReLU	(21, 21, 8)
Convolution	(5, 5)	(1, 1)	2	N	sigmoid	(21, 21, 2)

### 5.2 CREMI

The CREMI (Circuit Reconstruction from Electron Microscopy Images) dataset has three volumes but we decided to use only one (volume A) for our training.

This 3D image has a size of 1250x1250x125. Its corresponding groundtruth was also provided in the dataset. The segmentation has labeled connected components with really thin edges. For the evaluation, we used the CREMI library that was given with the dataset in Python 2. We, then, adapted it in Python 3.

A question arises : how to get an image segmentation from the affinity graph ? We used two different methods to answer this problem. First of all, a BPT (Binary Partition Tree) and a graph cut could get a good image segmentation. However, even with a strong threshold (around 0.99), isthmus appeared and fusion two different objects together. It's a big issue as it affects our scores. Secondly, to get rid of isthmus, we did an average affinity. Isthmus issues can be solved by improving our post-processing, and it should disappear with a better architecture.

Our results are promising as the different objects are well segmented. Yet, there are a few oversegmentations when regions are darker. Nucleus are not detected as a same object as the cell.

The second volume (volume B) was used as the test set. The image was similar to the one in the first volume but the objects are more stretched out. Here the objects are well segmented but it was harder on darker regions.

With more details, we can evaluate with numerical results, according to the Rand index and the VOI (variation of information) merge and split. "The Rand index is a measure of the similarity between two data clustering." \*explain VOI\* The Rand index should be the highest as possible, closer to 1. The metric VOI should be lower to be better. In the original architecture of 4 layers, the Rand index is 0.53 while we got 0.61 with our training set and 0.53 with our test set. Our Rand index is higher to the one of the original architecture. Our VOI is also lower so better. Thus, our results are hopeful knowing our architecture used was really simple. It's still far from the state of the art but it could get even better with a more complex network.

### 5.3 ISBI 2012

## 6 Going forward

## **7 Teamwork**

### **7.1 Team organization**

### **7.2 Task distribution**

### **7.3 Obstacles and overcoming them**

## 8 Conclusion