ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

AUTOMATIC GEOREFERENCING OF HISTORICAL CADASTRES

MASTER SEMESTER PROJECT

Théo Damiani

January 6, 2023

Contents

1 Introduction **1**
1.1 Previous work . . . 1
1.2 Data . . . 2

2 Proposed Method **3**
2.1 Nomenclature . . . 3
2.2 Preprocessing . . . 3
2.2.1 Semantic segmentation . . . 3
2.2.2 Resizing and dilatation . . . 4
2.2.3 Patches extraction . . . 5
2.2.4 Alignment parameters . . . 5
2.3 Main algorithm's steps . . . 6

3 Experiments **8**
3.1 Genetic Algorithm Implementation . . . 8
3.1.1 Naïve GA . . . 10
3.1.2 Restricted GA . . . 13
3.2 Loss function selection . . . 16
3.2.1 Mean Mask Symmetric . . . 16
3.2.2 Zero-Normalized Cross-Correlation (ZNCC) . . . 17
3.2.3 Intersection Over Union . . . 18
3.2.4 HOG-based function . . . 19
3.2.5 Structural Dissimilarity (DSSIM) . . . 19
3.3 Raster graphics representation . . . 20
3.3.1 Raster graphics with Euclidean distance transform . . . 20
3.3.2 Raster graphics with EDT thresholded . . . 22
3.4 Adjacent patch . . . 22
3.4.1 Adjacent patches of size 100x100 . . . 23
3.4.2 Adjacent patches of size 300x300 . . . 24
3.4.3 Adjacent patches of size 500x500 . . . 24
3.5 Best match selection . . . 27
3.5.1 Disruptive Selection . . . 27

4 Discussion **28**
4.1 Best experiments . . . 28
4.2 Future improvements . . . 30

5 Conclusion **30**

Abstract

This semester project has been hosted by the Digital Humanities Laboratory founded by Professor Frédéric Kaplan. It will focus on the georeferencing, i.e. spatial recalibration of 19th-century "Napoleonic" cadastral sheets. This project attempts to address the need for the automatic grouping of huge corpora like the Napoleonic cadastres. Indeed, a single map at the level of a city or a region would be a significant tool to facilitate the studies of historians and geographers.
Our method focuses on alignment by pattern matching. A genetic algorithm is used to search for the match of a pattern in another image. Several patches are extracted from the anchor image, and then, the best match is employed to compute the homography from the anchor to the target image.
This report is organized as a walkthrough to understand the different experiments tested but our best results were obtained with a restricted GA, the loss function Intersection Over Union and a disruptive selection of the best match.

ACKNOWLEDGEMENTS *We would like to thank specially Rémi Petitpierre for his advices and his availability during the project.*

1 Introduction

Napoleonic cadastres are one of the main historical sources of the 19th century. The cadastres cover a large part of Europe, and after their digitization, there is a database of several hundred thousand maps. Moreover, the Napoleonic cadastres are one of the first to list the owners of parcels. The parcels are numbered and a legend provides the name of the owner. Thus, with their legends and by cross-referencing them with other documents, the Napoleonic cadastres are important tools for environmental, sociological, and economic sciences. Therefore, to facilitate the analysis of these cadastres, the realignment of large corpora has been studied.
The first step is the extraction of useful informations, also called semantic segmentation. Usually, based on a convolutional neural network (CNN), it is possible to retrieve the text, the contours of each parcel and the road of the maps for example [4, 10].
Then, the realignment of the historical maps is performed on today's topology, typically with feature extraction methods. The algorithm will try to find the most descriptive features of the cadastre and match them to today's topology[7].
One objective of our new approach is to be as general as possible. We want to align the Berney cadastre[3] of the first half of the 19th century, but above all to provide a method capable of aligning as many cadastres as possible. That is why The 19th-century Napoleonic cadastres have been chosen. It offers an important regularity. Indeed, they were governed by the "Recueil méthodique des lois, décrets, règlemens, instructions et décisions sur le cadastre de la France..."[5] which details the instructions for surveyors to follow when creating the cadastre. Besides its uniformity, the Berney cadastre covers a large part of the peri-urban areas[13] which is not common in older cadastres. Thus, these new maps can further help to study urban evolution but it becomes more difficult to realign them by feature extraction because the peri-urban areas of the 19th century have changed a lot. Therefore our method does not focus on aligning with contemporary topologies but will try to align the cadastres with each other by recognizing border patterns.
The main challenge of this project is to obtain a satisfactory alignment in a reasonable computing time. These two objectives are at the core of the discussions of the report and will allow us to evaluate the developed methods

1.1 Previous work

A great starting point in this project was the previous work "Alignment of XIXth century cadasters"[11]. This previous project had the same objective: "The idea is to attach every map from a cadaster in order to get a single map of each city[11]". But then our new method will try to tackle some limitations and problems found with this previous report. First, the AlignmentTool[11] offers a highly supervised and interactive method to perform cadastres alignment. For the alignment of two maps, the user must manually specify the orientation of the two maps. He must also select in the first image a template, and in the second, an area where the method will try to find the previously selected template. One of the objectives of this new project is to be as unsupervised as possible. It aims to be generic so that the method can be used on any set of cadastral maps without any specific knowledge. The user only has to specify the list of maps he wants to align and no more human interaction is required. Another problem is that the tool does not perform a scale-invariant template matching, which can be a problem given the variation in 19th-century cadastres. Our method provides the user with a range of scales to consider in the matching process.

1.2 Data

The data involved in this report is the Berney cadastre of Lausanne from 1827. This cadastre has already been aligned, so it can be used as a control measure during the development of our solution. They are presented in '.tif' files which are georeferenced on a contemporary geographic coordinate system like OpenStreetMap[2].

2 Proposed Method

2.1 Nomenclature

Definitions were chosen for simplicity and convenience in the report. This table presents the project nomenclature.

Nomenclature	Definition
Patch	A patch or template is a small square subset of an image.
Anchor	The image from where patches are extracted at the beginning of the pipeline.
Target	The image in which the algorithm tries to find the previously extracted patches.
tx, ty	Optimization parameters of translations sought in the target.
θ	Optimization parameter of rotation applied to the patch.
α	Optimization parameter of scaling applied to the patch.
CNN	Convolutional Neural Network
GA	Genetic Algorithm
IoU	Intersection Over Union
ZNCC	Zero-Normalized Cross-Correlation
HOG	Histograms of Oriented Gradients
DSSIM	Structural Dissimilarity
EDT	Euclidean Distance Transform

2.2 Preprocessing

2.2.1 Semantic segmentation

The digitalized cadastres are too noisy to be the source images in the pipeline. The street names and lot indices are written on the maps. This is why the cadastres had to be segmented. By segmentation, we mean the process of partitioning a digital image into several regions. In our case, the segmentation [1] only detects the main shape of the streets and the delimitation of the land lots. This pre-processing step has already been performed and was given to us by our supervisor Rémi Petitpierre.

Figure 1: Digitalized image of the map Berney 001.

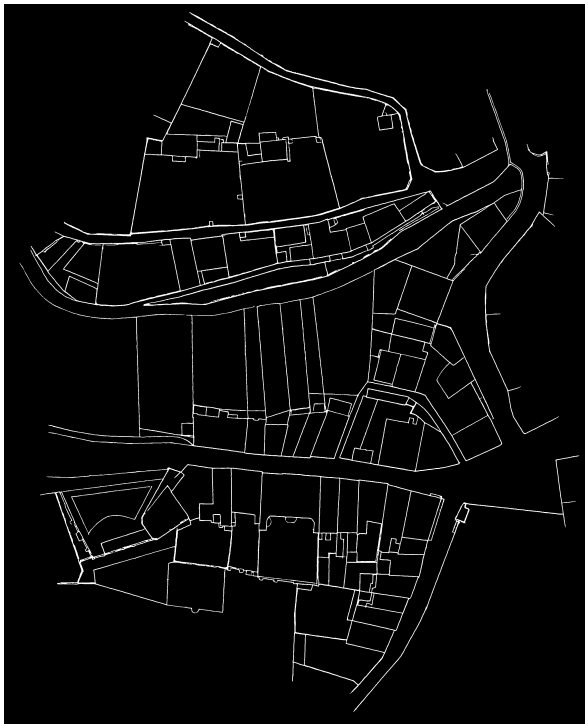


Figure 2: Segmented image of the map Berney 001.

2.2.2 Resizing and dilatation

The Berney dataset contains 75 segmented maps. The average size of the images is about 3587 pixels in height and 3820 pixels in width.

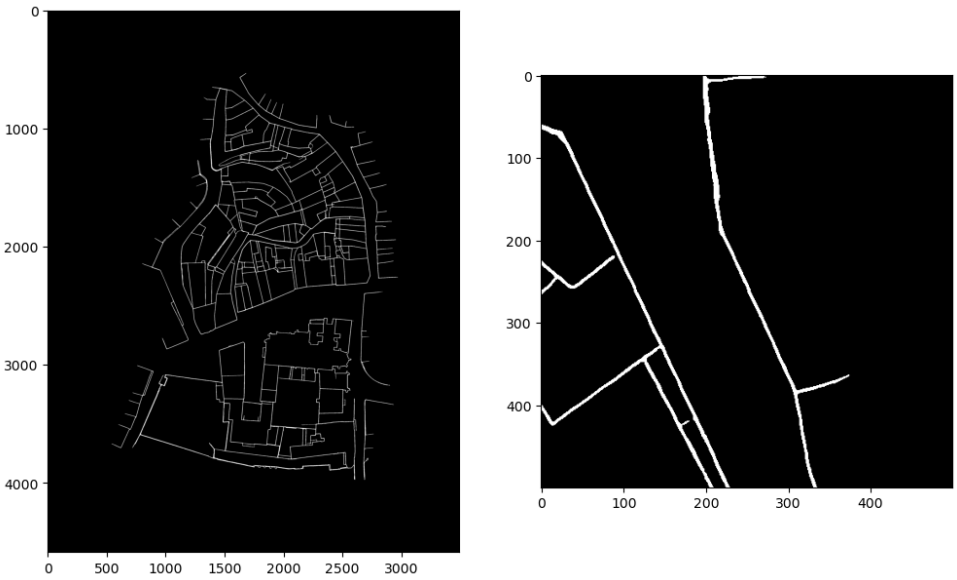


Figure 3: On the left, there is the map Berney 002 after segmentation and on the right, an arbitrary patch from the map Berney 001.

This size is too large for the computations to be carried out. So we proceed to a resizing of each image.

But in order not to lose too much information (the segmented lines), we also had to dilate the images. After testing different parameters, we found that a dilation by a 7x7 kernel shape followed by a resizing factor of 1/7 was a good balance between a small size for the resulting image and keeping enough information needed for the alignment.

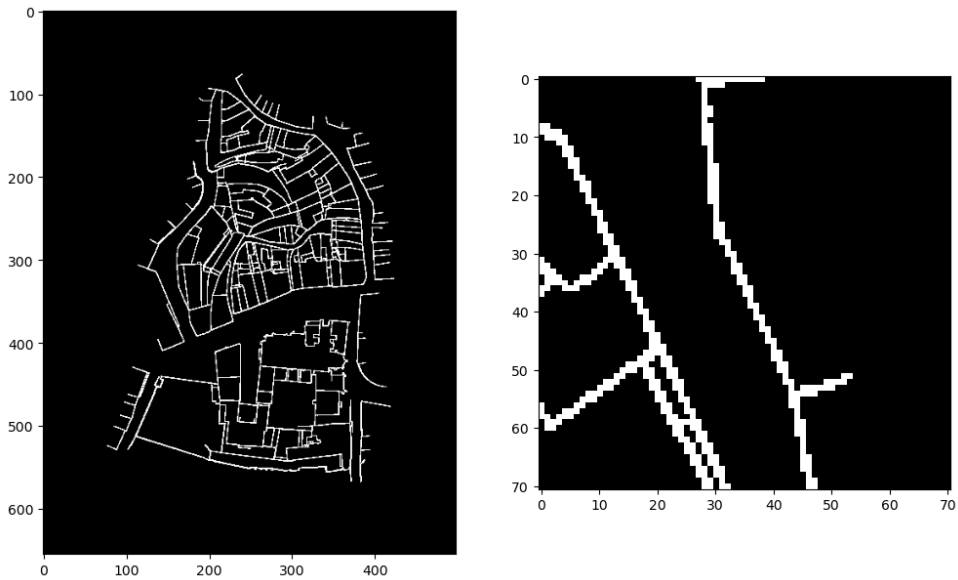


Figure 4: Same images than in the Figure 3 but after the resizing and dilatation processing. In this example, the target goes from 4600x3500 to 657x500 and the patch from 500x500 to 71x71

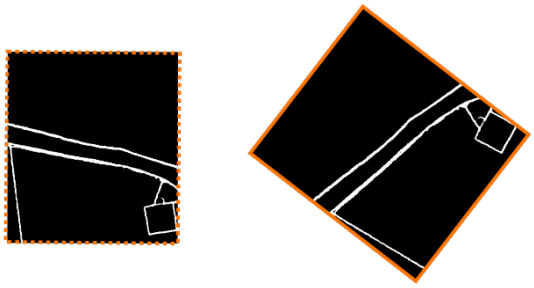
2.2.3 Patches extraction

The algorithm tries to align each cadastre with each others to create a mosaic. It is the edges of the cadastres that are of interest, the center of the cadastres does not influence the alignment. The patches are therefore extracted from the outermost lines of the segmented anchor image. To extract the arbitrary orange patch, an angle ω is taken between the fixed horizontal red line and the green line. Then, the pixel farthest from the center of the image and alongside the green line is selected as the center of the patch. Given an anchor image, a uniform distribution of ω is taken between the range $[0, 2\pi]$. The number of patches has been fixed at 20 and every patch has a size of 500x500 pixels.

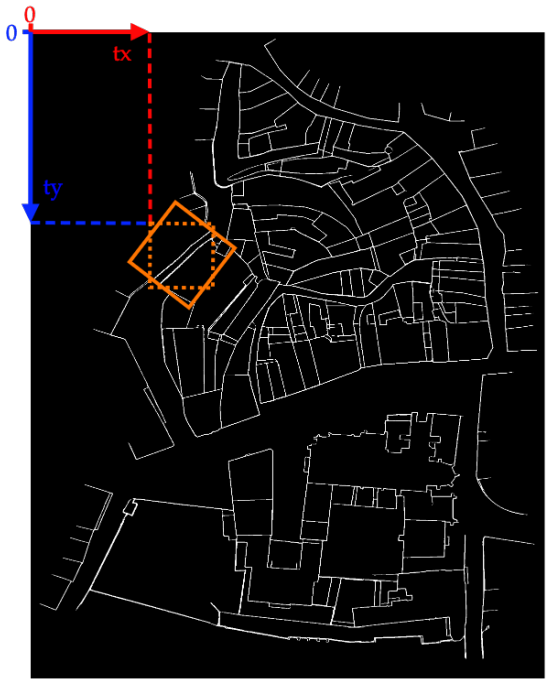
2.2.4 Alignment parameters

For a given patch, the algorithm tries to find the best loss function score by playing with 4 parameters. There are the translations in x and y noted tx and ty, the rotation of the patch noted θ, and the scale parameters α.

tx and ty are searched in the domain of the target image. The rotation θ, between $[0, 2\pi]$, and the scaling α, between $[0.95, 1.05]$, are applied to the patch for computing time optimization. It is faster to rotate and re-scale and small patch of size 500x500 pixels than the target images usually much bigger.



The patch extracted from the anchor has dotted orange contour. The patch with bold orange contour is after the rotation ω and the scaling α. On the left, we have the target image on which we placed at random x and y translations the processed patch. The target image on the left remains fixed, only the patch rotates and is re-scaled.

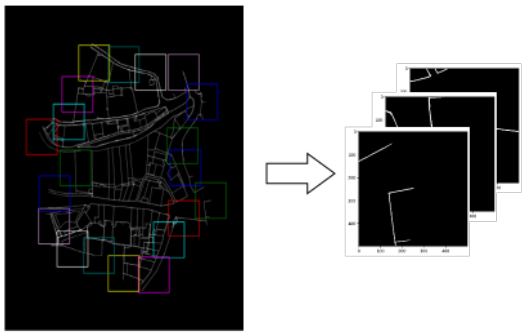


2.3 Main algorithm's steps

To summarize the main step of our solution, our solution will take several images and will process the data pairwise. In each pair, one image is defined as the anchor and the other one as the target. Some templates will be extracted from the anchor and the algorithm will try to find them in the target. Therefore, some pairs of homologous points will be known and the homography between them will be computed so that the target can be aligned with the anchor.

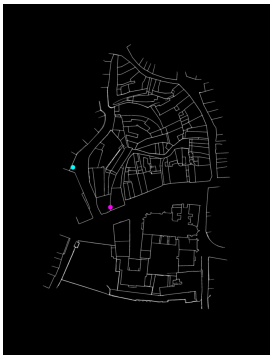
1. Take a pair of cadastral map. The left map is defined as the anchor and the right one as the target.

2. Extract some patches from the anchor.



3. For each patch, try to find it in the target by running the algorithm.

4. Compute homography between the coordinates of the best patch in the anchor and the coordinates of the same patch found in the target.



5. Wrap the target with the previous homography to align it with the anchor.

3 Experiments

In this section, the experiments tried during the development of our solution are described, they follow the path of our thinking for the realization of the project. The experiments are divided into subparts and all those belonging to the same subpart are compared and the one with the best result is selected. This means that we started with the choice of the GA implementation, there are two experiments: 3.1.1 Naïve GA and 3.1.2 Restricted GA. Then, the loss function has been studied with 4 experiments: 3.2.1, 3.2.2, 3.2.3, and 3.2.4.
Each experiment was tested on 20 alignments of cadastral pairs unless otherwise specified. The computation time for each experiment is listed at the top of the corresponding sub-section with the format *HH:MM:SS*.

3.1 Genetic Algorithm Implementation

The first thing to discuss in the pipeline is the method to find the best solution for the alignment of two different sheets. The very simple way to achieve this, is to test all possibilities by brute force. This choice was not retained because obviously, the computational complexity is too high.
The important thing to realize in this project is what the solution set looks like. In our template recognition problem, the fitness landscape is not smooth. The convergence of the loss function does not follow an easy ascent or descent to the global optimum as in a simple gradient descent. Our fitness landscape is rugged. The optimum solution will live in a very small niche and the others regions of the search space do not give any information about the direction the algorithm should take.
For example in the Figure 5, each pixel represents the fitness value of the first patch from the cadastre Berney 001 on the target cadastre Berney 002 at the given translations along the x and y-axis. The rotation and scaling parameters have been removed on purpose to be able to display the landscape in an easily 2D readable map. The Figure 5 shows that the landscape is extremely rugged. By comparison, Figure 6 shows a smooth landscape that would have been much more convenient for the algorithm to find the optimal solution.

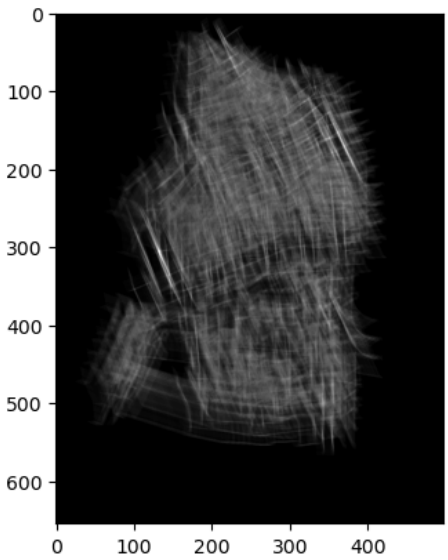


Figure 5: Fitness landscape of the first patch from Berney 001 to the target Berney 002.

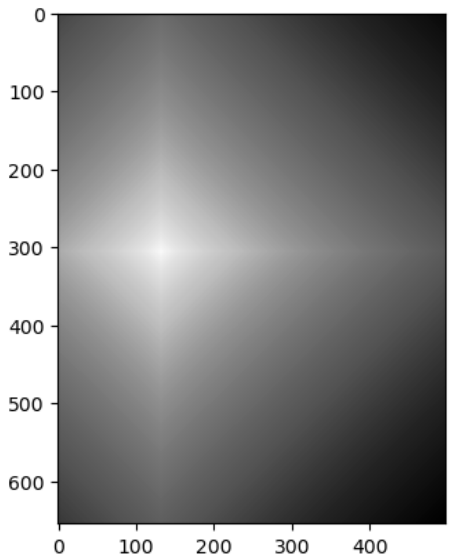


Figure 6: Hypothetical smooth fitness landscape for the alignment of the first patch between cadastres Berney 001 and 002.

The optimal solution lives at the center of Figure 7 or at y=306 and x=132 in Figure 5. We can see in figure 7 that there is almost no great area around the optimal solution, only 3 or 4 pixels around the optimal solution are also quite satisfying. In this situation, we do not take into account false positives. Indeed, the optimal solution for the loss might not be the desired match, but it will be discussed later.

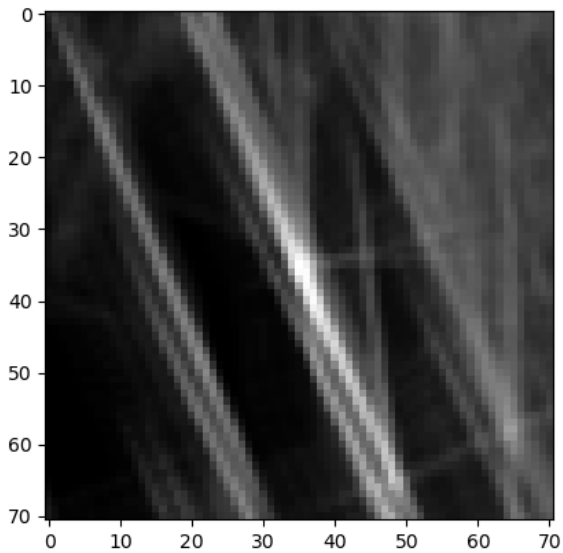


Figure 7: Fitness landscape zoomed-in on the optimal solution.

Our method takes into account, rotation and scaling, this means that the fitness landscape is composed of multiple landscapes as the Figure 5. Figure 5 is only with $\theta = 0$ and $\alpha = 1$, but θ lives in $[0, 2\pi]$ and α $[0.95, 1.05]$. θ and α are coded as Float in the program but if we simplify them and considered they are only Integers with 360 possibilities for θ and 10 for α, then there are 3 600 landscapes and only 1 of them contains the optimal solution like Figure 5.
Since the fitness landscape is huge and extremely rugged, we chose to use a genetic algorithm (GA)[9]. GA is widely used for optimization problems when the search space and the number of local minima are large. For example, it offers great performance in solving the Rastrigin function.

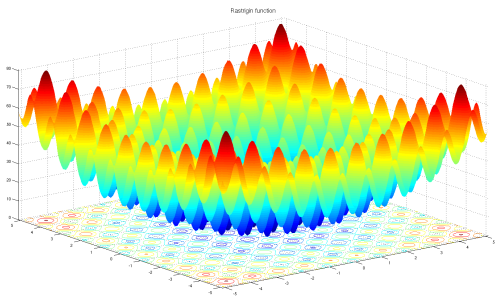


Figure 8: Rastrigin function[17]

3.1.1 Naïve GA

Elasped time: 08:01:32

First, we have chosen a classical implementation of a genetic algorithm. A GA element is composed of 2 integers representing the translations along the x and y axes, and 2 floats representing the rotation and scaling of the patch. At initialization the GA randomly draws its population, an arbitrary number of elements previously defined. Then, at each iteration, it evaluates each element of its population with a given loss function. Then, he must draw a new population. First, it retains a percentage of the best elements previously evaluated, they are called parents. And for the rest of his new population, he makes mutations on the parent elements. By repeating this process many times, the GA population will normally converge more or less the optimal solution.

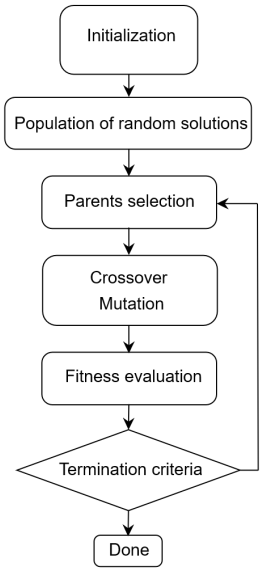


Figure 9: Basic flow of Genetic Algorithm.

In our problem, our fitness landscape is very steep, so we want to test the maximum number of possibilities. So we chose a fairly high mutation probability.

```
algorithm_param = {'max_num_iteration': 2000,
                   'population_size': 200,
                   'mutation_probability':0.7,
                   'elit_ratio': 0.01,
                   'crossover_probability': 0.9,
                   'parents_portion': 0.15,
                   'crossover_type':'uniform',
                   'max_iteration_without_improv':None}
```

For all experiments, we evaluate their results with the number of valid patches per alignment. For each alignment, we review manually all of the 20 matches found and note which ones propose a positive alignment.

An alignment is said to be positive or a match is said to be valid when the solution found by the GA gives a satisfactory alignment. Figure 10 is an example of a positive alignment. The patch in red correctly overlaps the target in blue. Thus, if we recover the homologous points, the 4 corners of the patch in red and the 4 corners of the area in blue, we obtain the alignment in Figure 12 by calculating the homography between them.

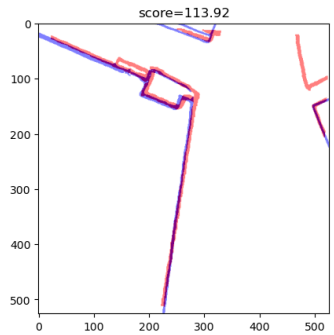


Figure 10: In red the patch number 2 from the Berney 001 anchor and in blue the area selected by the GA from the Berney 002 target.

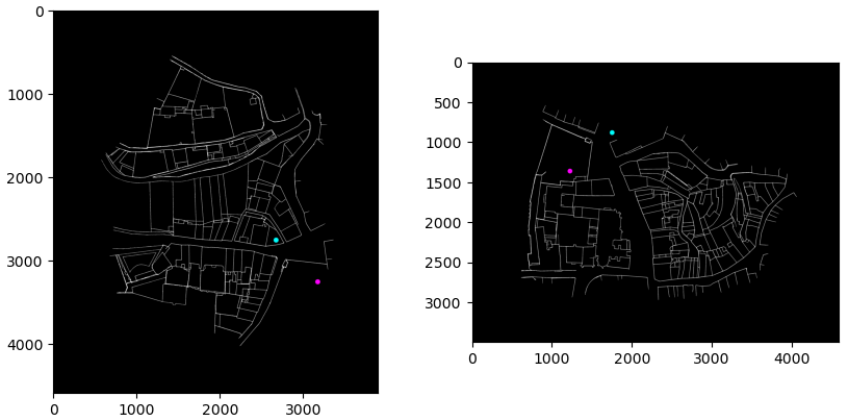


Figure 11: On the left this is the anchor Berney 001 with the blue point, the top left corner, and in purple, the bottom right corner of the patch in red in the Figure 10. On the right the target Berney 002 with the top left corner and the bottom right corner of the patch in blue in Figure 10

.

Figure 12: Alignement of the anchor Berney 001 and the target 002 with the homography found by the GA from the patch number 2 (Figure 10).

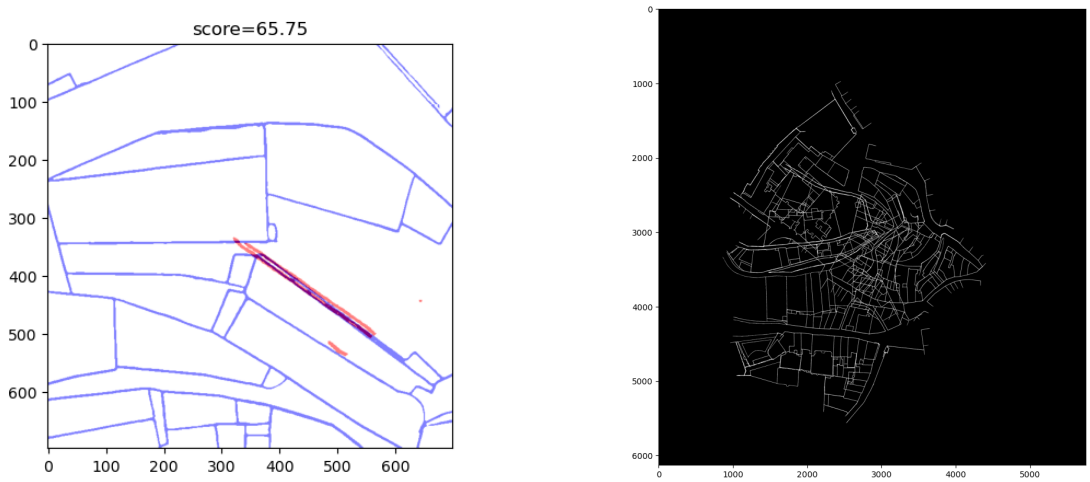


Figure 13: Alignement found by the GA with the patch number 17 (Anchor:001, target:002). This alignement is not valid.

By repeating this check for all 20 matches in each alignment, we can measure the amount of satisfactory patch.

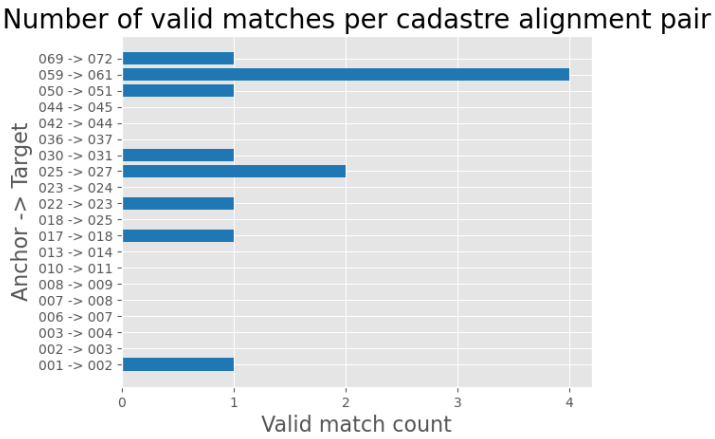


Figure 14: Experiment 3.1.1: Thus, the alignment of 20 maps with a basic GA gives 12 valid matches. It is normal to get several valid matches per cadastral pair since the alignment can be along a common border more or less large. But we can see that it fails to align 12 maps.

At this point, we counted the number of valid matches. However, we still need to see which match is considered the best by the algorithm. Indeed, to compute the homography between the anchor and the target (like in Figure 12), we have to select the homologous points of the best or a set of the best matches. Let's take the case of anchor 001 and target 002, we can see that there is only one valid match (Figure 14) and that it is with the patch number 2 (Figure 10). Unfortunately, Figure 15 tells us that patch number 2 is not the one with the lowest score. So the algorithm will take match number 17, an invalid match, to compute the homography and align cadastre 001 and 002. It will result in a wrong alignment, Figure 13.

Figure 15: Matches found by the GA between anchor 001 and target 002 and the fitness function values of their corresponding patches.

That's why we must also take into account the number of matches that are valid while having the best score. We have called these matches, the true positive matches. The goal would be to have at least one true positive match by alignment of anchor-target pair.

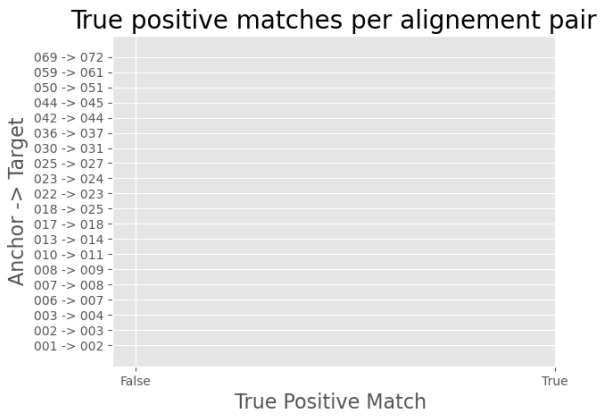


Figure 16: Experiment 3.1.1: this experiment contains no true positive match.

3.1.2 Restricted GA

Elasped time: 07:33:49

The big problem with this project is the GA search space. For all cadastres, the search space is much too large. For example, cadastre 002 after preprocessing has a size of 655x498. Let's take simple integer dimensions for rotation and scaling, e.g. 360 and 10 (normally we have many more possibilities as rotation and scaling are continuous spaces). The search space for a patch in target 002 is thus $655 \times 498 \times 360 \times 10 = 1174284000$. This problem induces too much variance in the GA result, for example in the alignement of the map Berney 001 and 002, the GA will find the valid match for the patch number 2, 5 times out of 20 trials.

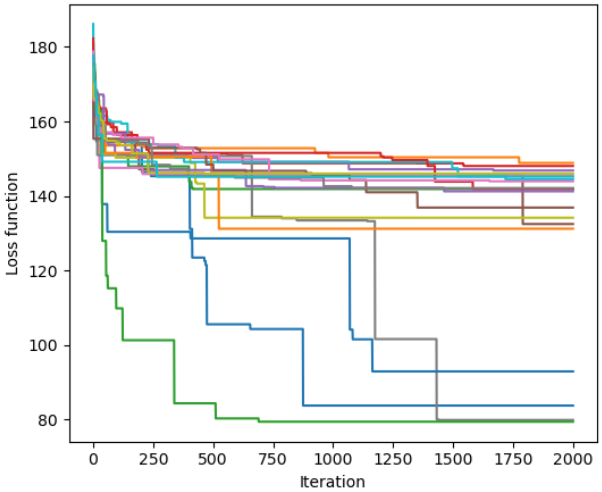


Figure 17: With 001 as anchor and 002 as target, score of patch 2 according to the number of iterations of the GA, 20 different trials.

Thus we decided to restrict the search space for translations. In this experiment, the translations that the GA can try are those defined in the set near the outer edge of the target cadastre. The implementation of this method is done before the GA. A heat map indicating the distance from a pixel to the nearest edge along the line passing through the center of the image is computed, Figure 18. Then, using a treshold, the points with the shortest distances are selected in a list.

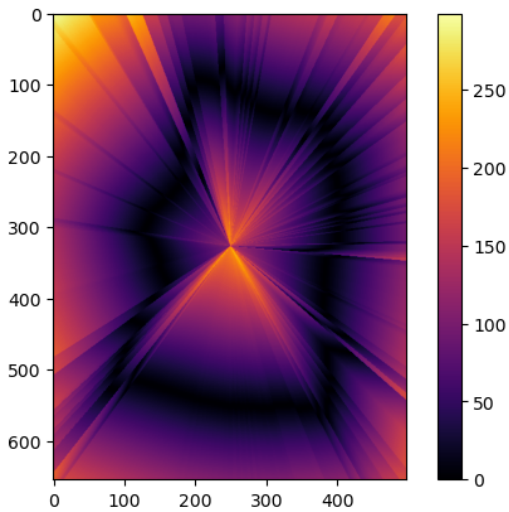


Figure 18: Heatmap of the distances to the outer edges of the Berney 002 cadastre.

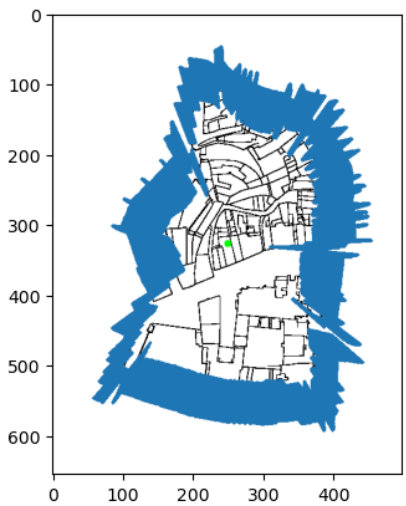


Figure 19: Tresholded heatmap, blue points are selected to be the set of possible translations.

Now, tx and ty are no longer represented as integers but as a pair of valid translations. Thus when initializing the population the new GA randomly takes a pair of valid translations contained in the blue space of Figure 19.
In addition, we modified the GA so that the mutations of all parameters, translation (tx, ty), rotation θ and scaling α follow a Gaussian distribution. We set the mean to the current value for each parameter. The standard deviation for rotation is 5 degrees and 0.2 for scaling. For translations, the standard deviation is 20 pixels and the GA checks that the new translation pair is included in the valid set. So now, at each iteration, we have in the population:

- 15% of parents.
- 15% of mutated elements from the parents.
- 70% of new random elements.

Each elements contains translation in the valid set of translations pre-defined.
The rest of the GA does not change, and we kept the same number of iterations and population size.

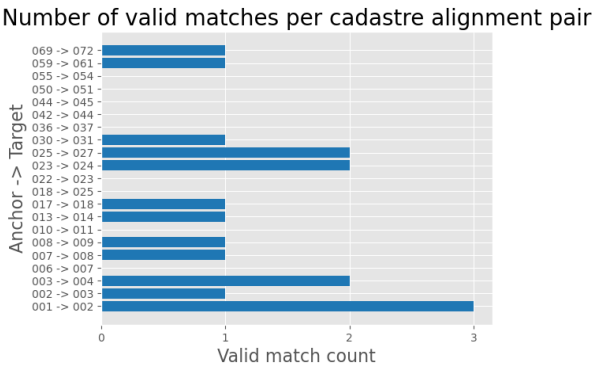


Figure 20: Number of valid matches during the experiment 3.1.2

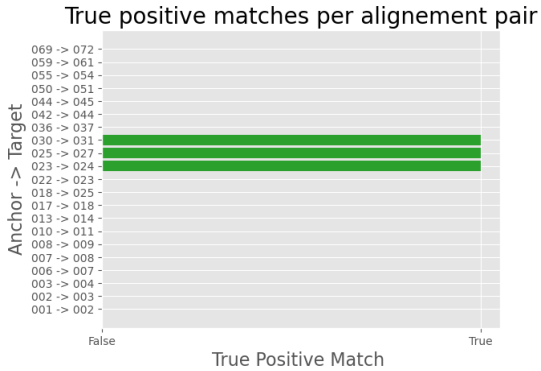


Figure 21: Number of true positive match during the experiment 3.1.2

This restricted GA performs better than the basic one. It covers 11 cadastres while experiment 3.1.1 covers 8. And the method finds 3 true positive match. Besides computation time are nearly the same, so we have chosen to use the restricted GA for the following steps.

As computation times are nearly the same for both experiments but results are more satisfying with experiment 3.1.2, we have chosen to use the restricted GA for the following steps.

3.2 Loss function selection

In this section, we will study different loss functions. Previously, we used a very simple loss function, but we felt it was a priority to define the genetic algorithm correctly.

3.2.1 Mean Mask Symmetric

Elasped time: 06:05:40

The idea behind this function is to average each pixel of an image under the pattern of another image when the two are overlapped.
When the GA tests a solution, for example, in Figure 22, this function will average each pixel of the target being overlapped by the pattern of the patch. Figure 22 exists just to visualize the result, in reality, the algorithm works with images as in Figure 23. So by overlap, we mean that the function sums the pixels of one image under the black pixels, those equal to 0, of another image when both are tested with a certain combination of tx, ty, θ, and α.

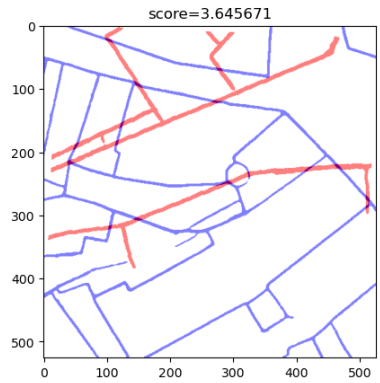


Figure 22: Random solution tested by GA. In red the patch in blue the target area.

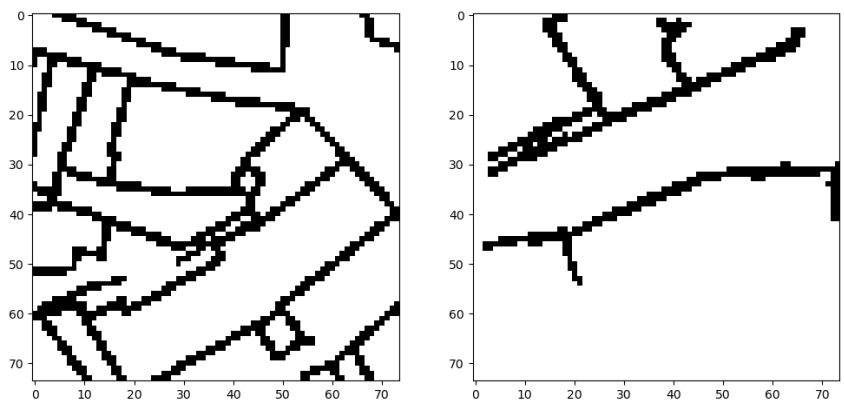


Figure 23: The real representation of the solution tested in Figure 22.

For a question of symmetry, the pixels of the target under the patch should be averaged and the pixels of the patch under the target should be averaged.
For more clarity here is the code of this function. Note that we need to check if the target section, X, contains a pattern, maskX not null.

```
def mask_mean_symmetric(X, Y):
    maskY = Y==0
    maskX = X==0

    yc = Y[maskX]
    if not np.any(yc):
        return (np.mean(X[maskY])+1000)/2
```

```
    else:
        return (np.mean(X[maskY])+np.mean(yc))/2
```

So with this loss function, we get the following result:

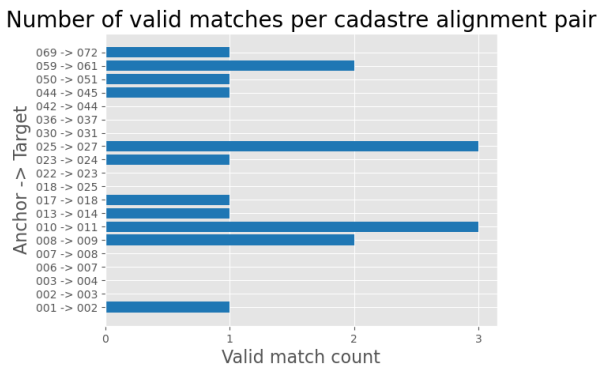


Figure 24: Number of valid matches during the experiment 3.2.1

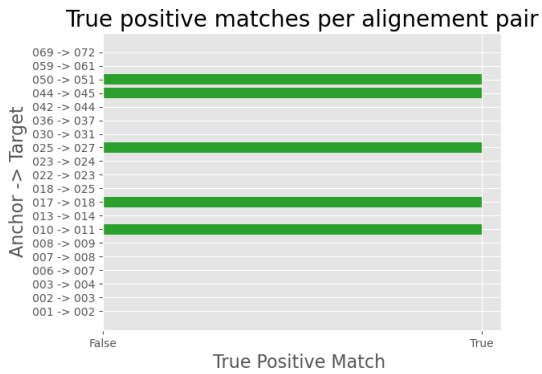


Figure 25: Number of true positive match during the experiment 3.2.1

With this loss function, our method covers 11 cadastres and finds 5 true positive matches.

3.2.2 Zero-Normalized Cross-Correlation (ZNCC)

Elasped time: 08:21:23

NCC is a common measure of the similarity of two signals or series. It is also widely used for pattern recognition when the imagery is sensitive to brightness. This is not our case since we are in binary representation, but the cross-correlation is still widely used in template matching, we wanted to see if there were benefits to use it in our problem.
The ZNCC function is defined as follows:

$$ZNCC(t,p) = \frac{1}{n}\sum \frac{1}{\sigma_p \sigma_t}(p-\mu_p)(t-\mu_t) \tag{1}$$

With n the number of pixels in t and p, μ_p the average of the patch, μ_t the average of the target area and σ_p, σ_t the respective standard deviations.

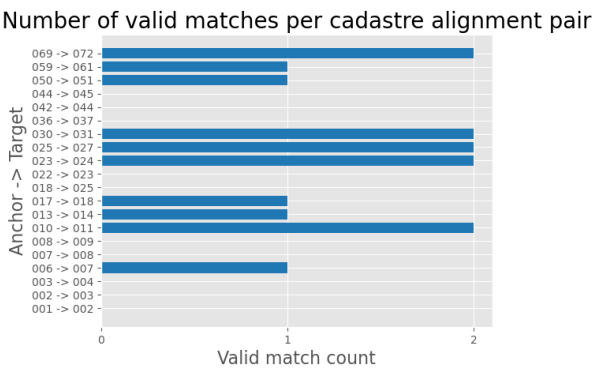


Figure 26: Number of valid matches during the experiment 3.2.2

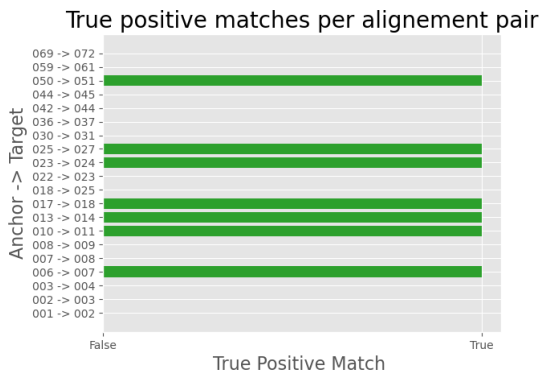


Figure 27: Number of true positive match during the experiment 3.2.2

So, with ZNCC, the method covers 10 cadastres and finds 7 true positive matches.

3.2.3 Intersection Over Union

Elasped time: 05:02:42

Then we tried the Intersection Over Union function. It is defined as the size of the intersection divided by the size of the union of the sample sets. In our case, we perform the intersection and union of the patterns of the patch and the target area. This time, the images are reversed compared to Figure 23. Therefore, we can use the python binary operators "and" (&) and "or" (|). By its definition, this function provides symmetry directly.

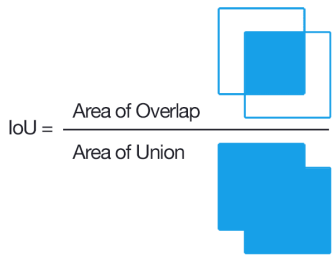


Figure 28: Intersection Over Union[14]

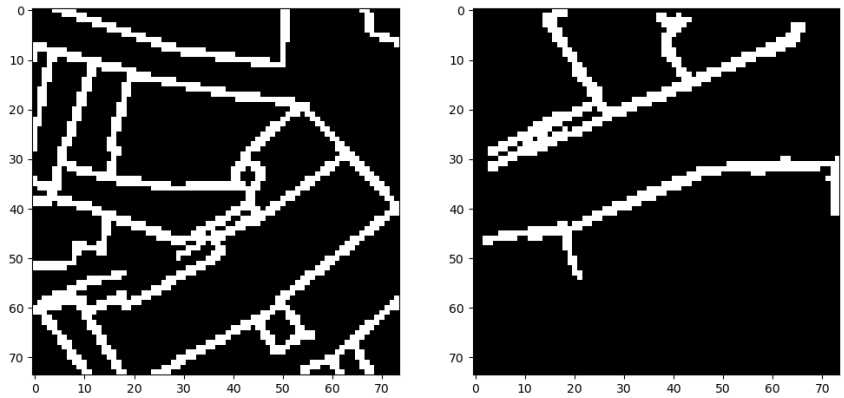


Figure 29: The representation of the patch and target section.

```
def IntersectionOverUnion(X, Y):
    i = np.sum(X&Y)
    u = np.sum(X|Y)
    return 1-(i/u)
```

So with this IoU function, we get the following result:

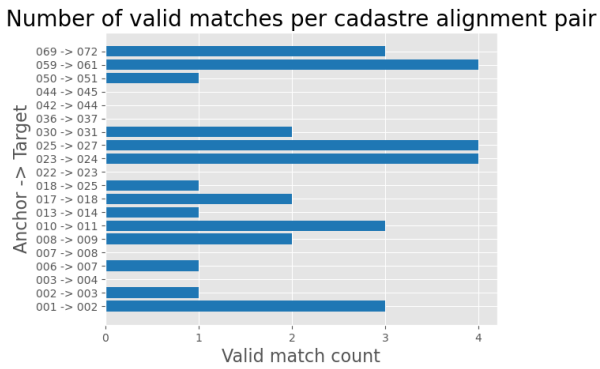


Figure 30: Number of valid matches during the experiment 3.2.3

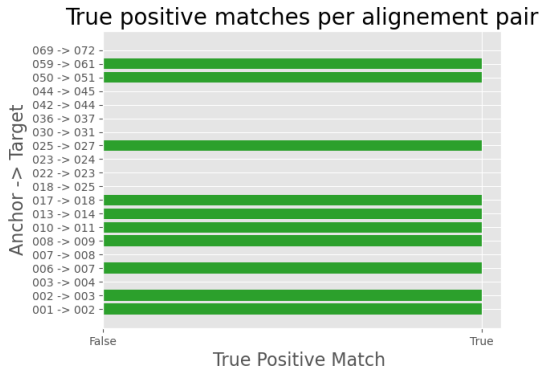


Figure 31: Number of true positive match during the experiment 3.2.3

Thus, our method covers 14 cadastres and finds 10 true positive matches.

3.2.4 HOG-based function

We tried as a loss function the distance of the histograms of the oriented gradients of the patch and the target. But the GA takes 20 minutes to perform 2000 iterations for a single patch, so we could estimate 133 hours of computation for the alignment of 20 cadastres, which is way too much with respect to other loss functions. Moreover, as the results of the alignment of a pair of cadastres were not convincing, we did not go further in the study of this function.

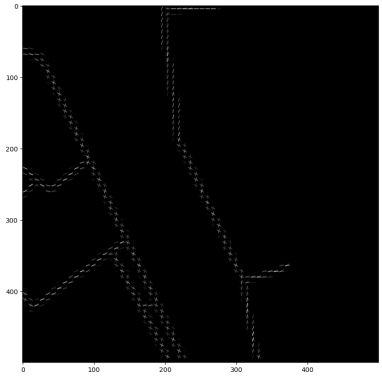


Figure 32: Histograms of the oriented gradients of a patch.

3.2.5 Structural Dissimilarity (DSSIM)

Elasped time: 05:08:34 for only 5 pairs of cadastres

The last loss function we tried is Structural Dissimilarity[15, 16].

$$DSSIM(p,t) = \frac{(1 - SSIM(p,t))}{2} \tag{2}$$

$$SSIM(p,t) = \frac{(2\mu_p\mu_t + c_1)(2\sigma_p\sigma_t + c_2)(cov_{pt} + c_3)}{(\mu_p^2 + \mu_t^2 + c_1)(\sigma_p^2 + \sigma_t^2 + c_2)(\sigma_p\sigma_t + c_3)} \tag{3}$$

With c_1, c_2 and c_3 variables to stabilize the division when the denominator is very low.

This function evaluates more the structure of the images than a pixel-to-pixel difference. However, the patches are not necessarily of the same structure. Thus the DSSIM will not succeed in correctly matching the patches that are not identical to the target. For example, in Figure 33, the patch is much smaller than the target in blue which overflows at the bottom. DSSIM will fall into a random match. While when the patches and targets are nearly identical as in Figure 4, DSSIM works correctly. However, these cases are not predominant.

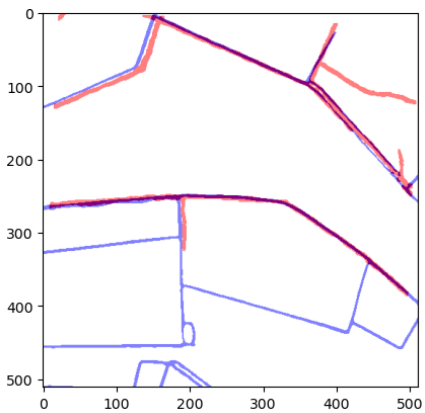


Figure 33: Match found with IoU loss. Very difficult for the DSSIM to find it.

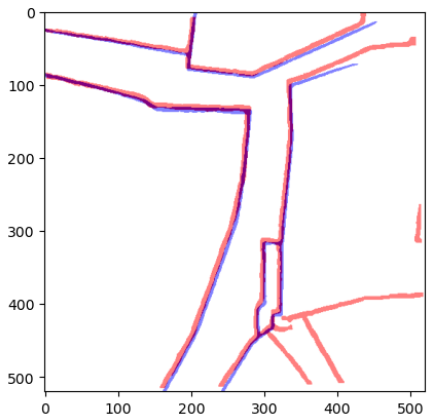


Figure 34: Positive match found by the DSSIM.

The computation time is also longer than IoU, ZNCC or Mask Mean, on average 3 times longer. So, as HOG function, we did not continue with the DSSIM.

In terms of computation times and quantity of true positive matches, IoU (3.2.3) is the best loss function and so it is the function selected.

3.3 Raster graphics representation

Before going further with the IoU loss function, we wanted to think of a new way to represent images.

3.3.1 Raster graphics with Euclidean distance transform

Elasped time: 06:54:34 for only 5 pairs of cadastres

So far, images are represented in binary. But, following our discussion of loss functions, we were wondering if we could guide the GA a bit more. For example, the IoU function is versatile and works quite well but for the GA to find the optimal solution, it has to fall by chance in a region close to the right solution. And this region is quite restricted. This problem already mentioned in 3.1 is that the fitness landscape is too steep. So we thought of computing the Euclidean distance of each pixel to the nearest segmented pixel. The image will be now composed of gradients. This may help the loss function to converge more smoothly to the optimal solution.

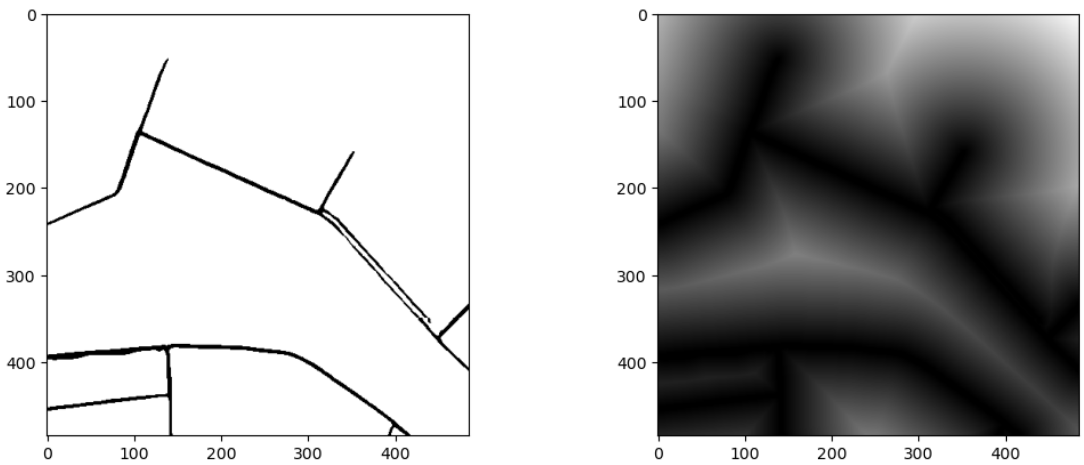


Figure 35: Patch before and after Euclidean distance transform

However, with this representation, it makes sense to use the symmetric mask-mean loss: the patch and target are taken as in Figure 23, and first, the mask-mean loss is computed with the target in Euclidean Distance Transform (EDT) and the normal patch, and then, for symmetry, the mask-mean loss is computed with the normal target and the EDT patch.
The transformation of images in EDT is performed in the loss function as we have rotation and scaling, so it can not be in a preprocessing step.

```python
def mask_mean_symmetric_edt(X, Y):
    X_edt = distance_transform_edt(X)
    l1 = mask_mean(X_edt, Y)

    Y_edt = distance_transform_edt(Y)
    l2 = mask_mean(Y_edt, X)

    return l1+l2
```

After launching 5 alignments, we stopped the algorithm since the GA found only one valid patch, Figure 36. Indeed, the GA seems to get stuck in the targets being dense. This can be seen in Figure 38, the Euclidean distance transformation gives too many low values when the area is dense, in terms of segmentation. And so, the patches are often matched in the densest area of the target cadastre.
Also, note that the computation time is at least 4 times longer with the EDT transformations than in experiment 3.2.3.

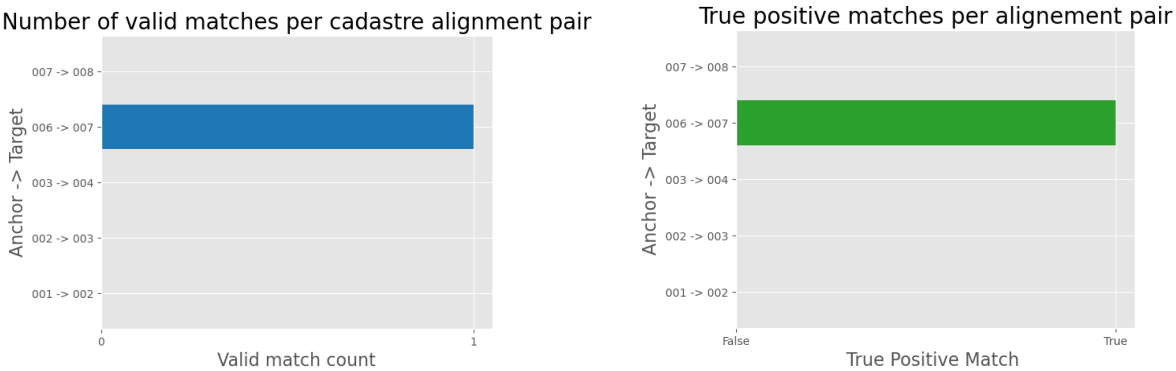


Figure 36: Result of 5 alignments with EDT representation.

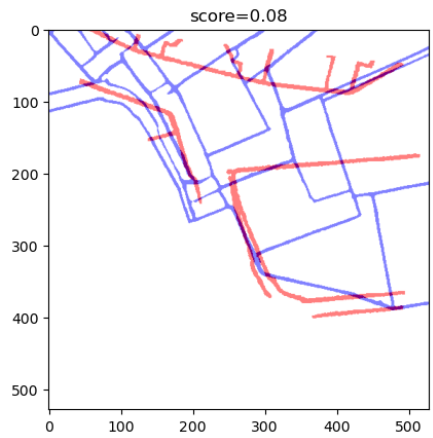


Figure 37: Example of match found with EDT raster representation.

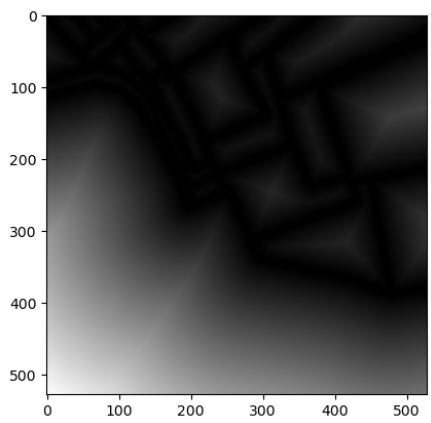


Figure 38: EDT representation of the blue target from Figure 37.

3.3.2 Raster graphics with EDT thresholded

To compensate for the too-low values in dense regions, we tried to add a threshold to the Euclidean distance transformation. But the problem is that we need a high threshold so that the dense areas are not all black. And so we get stuck, either we take advantage of the EDT but the dense areas are too black or the dense areas are thresholded but the advantage of the EDT becomes almost nonexistent.
For example, it is necessary to threshold the target of Figure 38 up to the rightmost EDT of Figure 39 so that the patch does not get stuck in the dense area, otherwise for the 3 other EDTs, in Figure 39, the GA ends as in Figure 37.

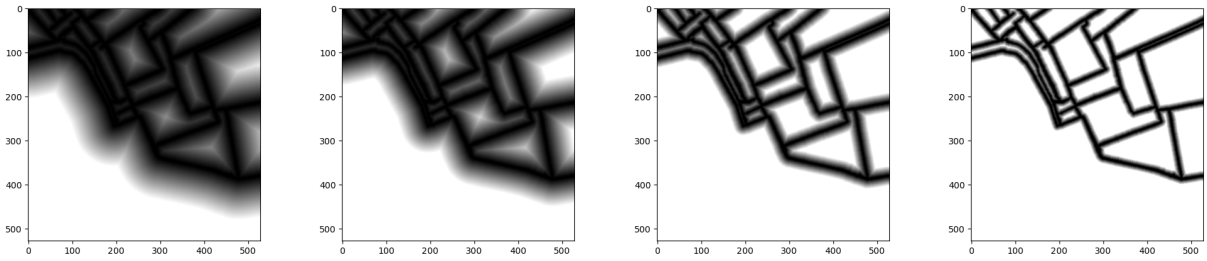


Figure 39: From left to right: central patch number 3 from anchor 034, its adjacent patches and its corrected adjacent patches.

Thus, we keep the representation of the patches as in Figure 29 and we continue with IoU as loss function.

3.4 Adjacent patch

One problem with our method is the appearance of false positives. For example, the extraction of patches for the Berney 030 anchor will give problematic patches, patch numbers 10 and 11. Indeed, these patches are very simple, they are just a line. So the GA manages to match them with a very low loss function, Figure 40, even though they are not included in the target.

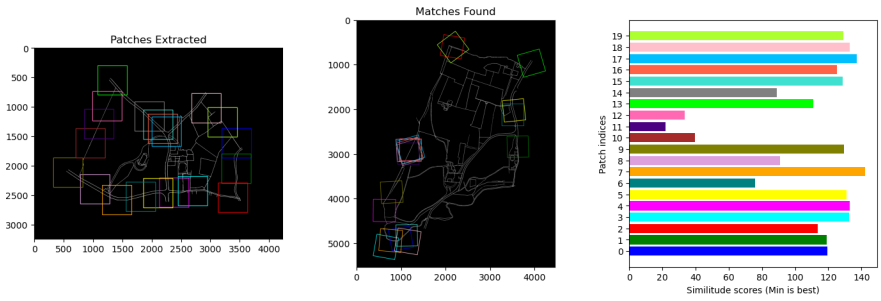


Figure 40: Loss function summary of the alignement of anchor 030 and target 031. The valid patch is the number 6.

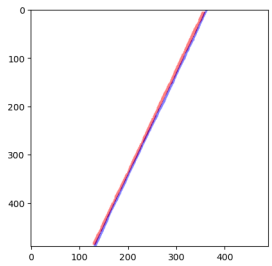


Figure 41: Alignement patch number 11.

So to try to mitigate these false positives, we decided for each patch to extract also two adjacent and neighboring patches. So we try to match on a larger surface each patch, which should increase the loss

function of these false positive patches.

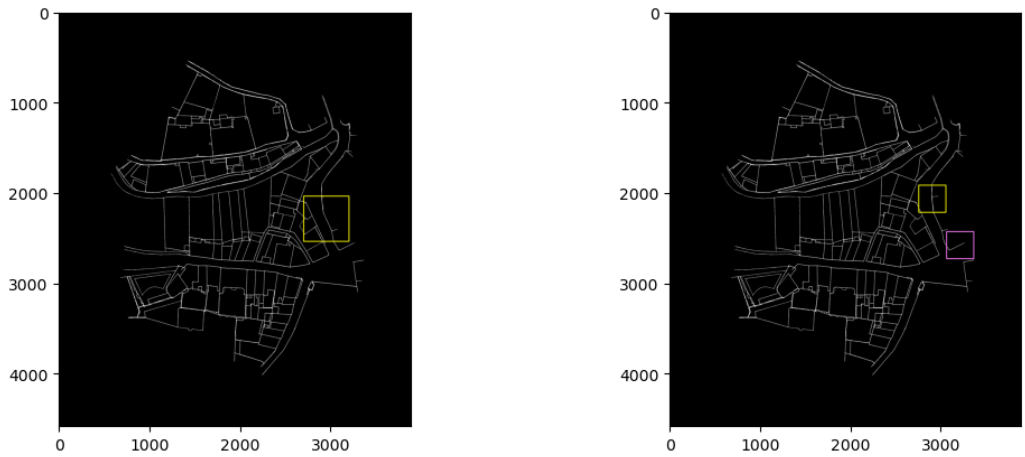


Figure 42: Main patch on the left and its adjacent patches on the right.

The GA works exactly as before, but now the loss function is the sum of the loss of the center patch and the loss of the adjacent patches. The adjacent patches use the same solution as the central patch, we just look at where the adjacent patches are when the central patch is matched to a solution (tx, ty, θ, α).
So for the adjacent patches, we can't just use theta. Indeed theta is the rotation through the center of the main patch. We must therefore perform calculations to estimate the rotation through the center of an adjacent patch as a function of theta. These calculations are relatively time-consuming for use in a loss function. Thus the experiments with adjacent patches are much longer than the previous ones.

3.4.1 Adjacent patches of size 100x100

Elasped time: 10:12:31 for only 9 pairs of cadastres

Once again we stop the calculation because of the bad result, Figure 44 on the first cadastres, only 3 true positive matches. For these 6 alignments, the algorithm should have gotten 6 true positive matches to compete against other methods like 3.2.3.

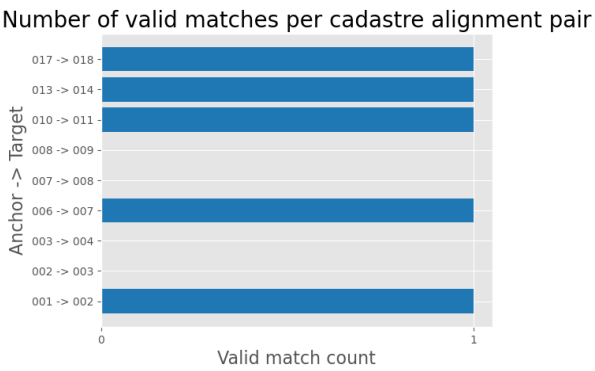


Figure 43: Number of valid matches with adjacent patches of size 100x100.

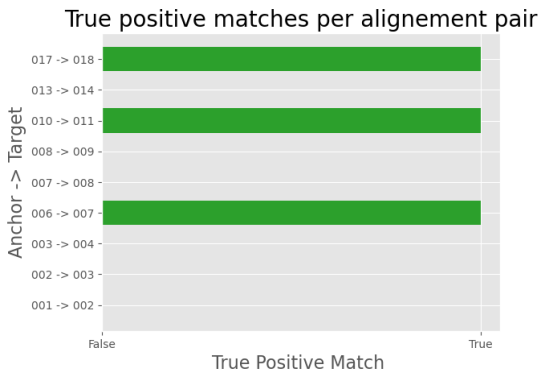


Figure 44: Number of true positive match with adjacent patches of size 100x100.

3.4.2 Adjacent patches of size 300x300

Elasped time: 24:59:16

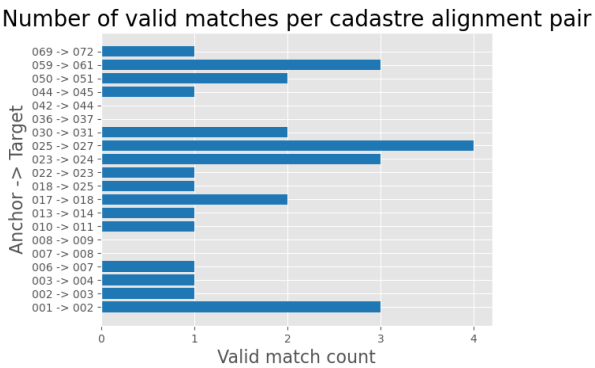


Figure 45: Number of valid matches with adjacent patches of size 300x300.

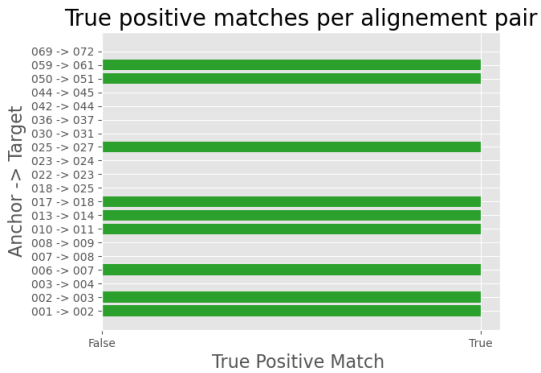


Figure 46: Number of true positive match with adjacent patches of size 300x300.

In terms of coverage, this is the best result we have so far, 16 valid matches versus 14 without adjacent patches and with IoU (Experiment 3.2.3). However, there is one less positive true match, 9 against 10 and the computation time is extremely long, 25 hours against only 5 hours.

3.4.3 Adjacent patches of size 500x500

Elasped time: 28:40:21

In this experiment, the alignment 003 to 004 shows the solution to the false positives that we tried to implement. In Figure 47 drawn from experiment 3.2.3, without adjacent patches, we can see that patch 6 has the lowest loss and is therefore a false positive. Indeed, for this pair of cadastres, the valid patches are numbers 0, 1, and 2.

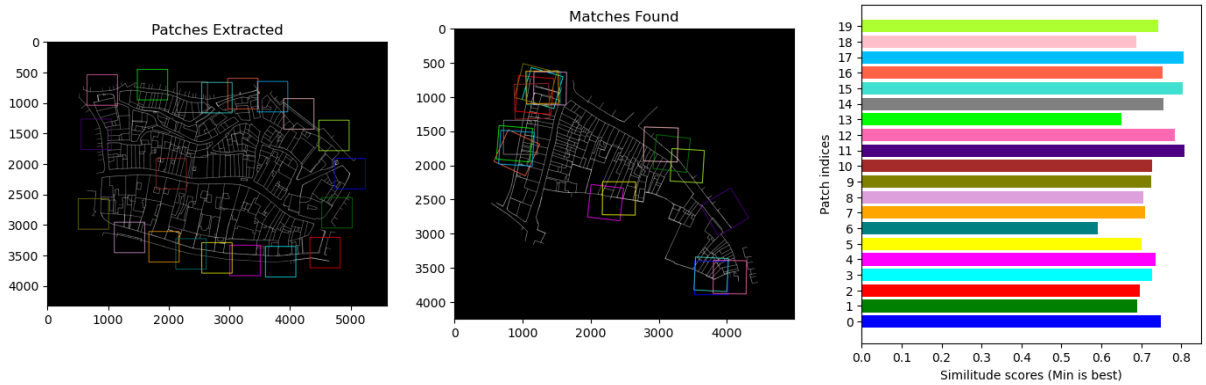


Figure 47: IoU loss function summary of the alignement of anchor 003 and target 004 with restricted GA and no adjacent patches.

Using adjacent patches of size 500x500, patch number 6 is relegated behind the valid patch number 2.

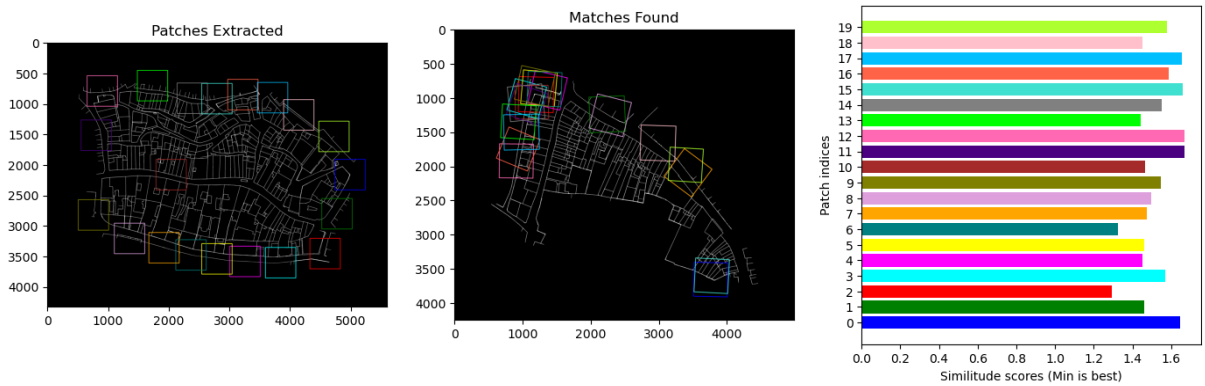


Figure 48: IoU loss function summary of the alignement of anchor 003 and target 004 with restricted GA and 500x500 adjacent patches.

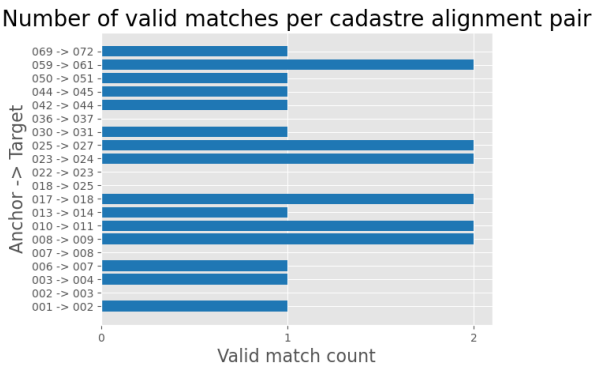


Figure 49: Number of valid matches with adjacent patches of size 500x500.

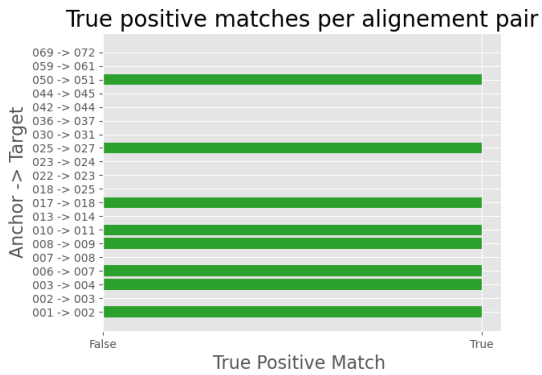


Figure 50: Number of true positive match with adjacent patches of size 500x500.

In terms of results, the algorithm with adjacent patches size 500x500 worked less well than with a size of 300x300. Although it is the only method that allows us to obtain the positive true match for the cadastres 003, and 004, it may become complicated for the algorithm to find simpler matches. The larger the adjacent patches are, the more variations in the parameters tx, ty, θ, and α will influence the loss function. This can be useful to remove false positives but can also be penalizing for other matches.
For example, patch number 11 from anchor 030 is still a false positive, Figure 52, the adjacent patchs do not give more different informations.

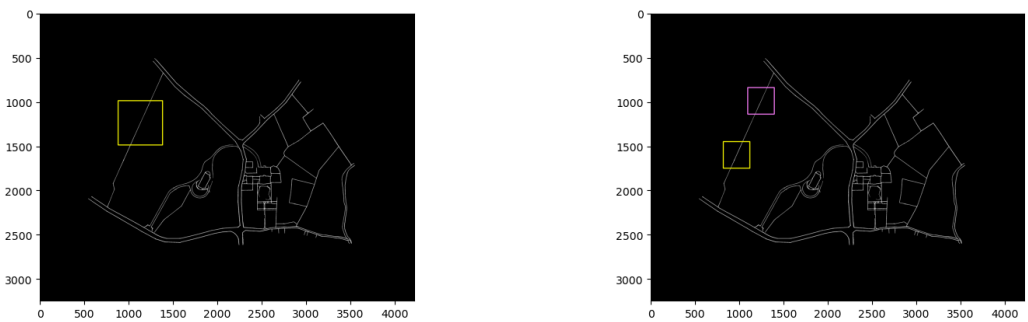


Figure 51: Main and adjacent patches number 11 from anchor 030.

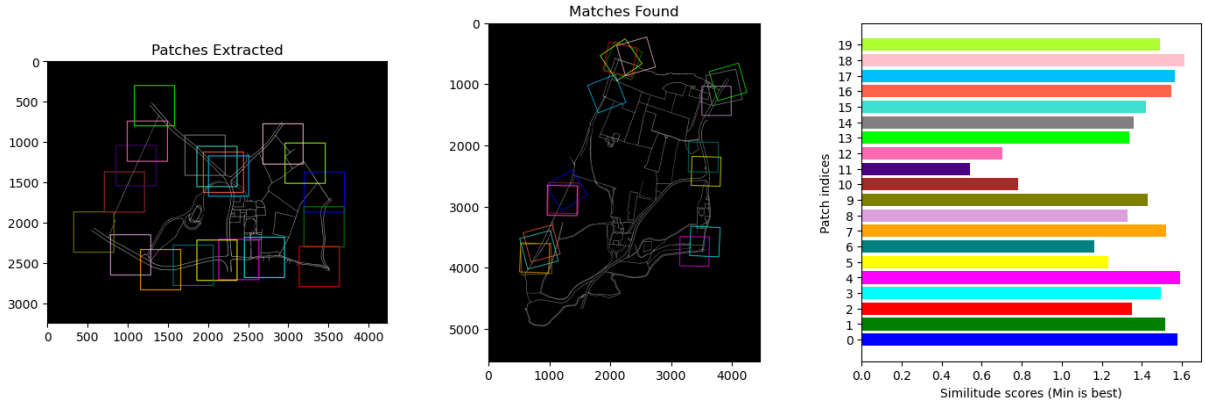


Figure 52: IoU loss function summary of the alignement of anchor 030 and target 031 with 500x500 adjacent patches.

In addition, we must be careful in extracting adjacent patches. They are extracted like the main patches and just a difference in the extraction angle is given to the function. However, for some patches located at "vertices", the adjacent patches will be very far away and thus no longer represent the neighboring areas of the selected patch. Thus, a correction had to be applied to the adjacent patches: all the centers of the adjacent patches farther by a threshold than the center of their central patch, are brought closer by reducing the angle difference in the extraction.

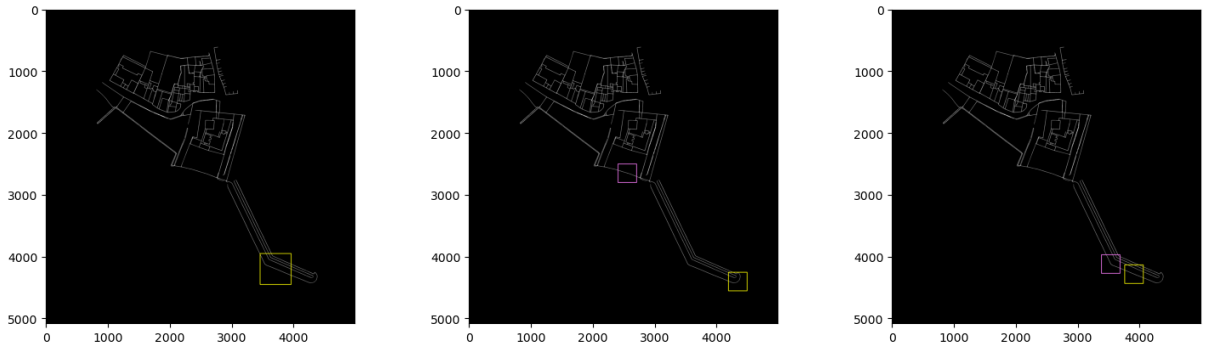


Figure 53: From left to right: central patch number 3 from anchor 034, its adjacent patches and its corrected adjacent patches.

The use of adjacent patches of size 300x300 seems interesting to improve the robustness of matches in some cases. However, the results obtained do not bring any breakthrough in the number of true positives either. Moreover, due to the long computation time of this method, we have chosen not to use adjacent patches.

3.5 Best match selection

Until now the selection of the patch to compute the homography between the anchor and the target was done by taking the patch with the smallest loss function value. But, sometimes the valid patch has a larger loss than a false positive one. For example, in Figure 30, for the pairs (069, 072), (030,031), (023,024), and (018,025), the valid matches are not the green selected matches, Figure 31.

3.5.1 Disruptive Selection

Elasped time: 07:05:23

We start with the assumption that single patches that are false positives can be matched to different locations on a target. For example, the patch in Figure 41, will have a fairly equivalent loss function between very different solutions. However, it is impossible to find a low loss value for a complicated patch like in Figure 33. That's why we tried to select the best patch by disruption. At the end of the GA iterations, we divide the score of the best solution, at index 0 in the GA population, by the score of the first solution in the population outside a predefined translation interval, we chose an interval of 20 pixels.

After running a simulation of 20 alignments with the same configuration as experiment 3.2.3, we obtained the results in Figure 54. In this case, the disruptive selection allows the method to find one more true positive match.

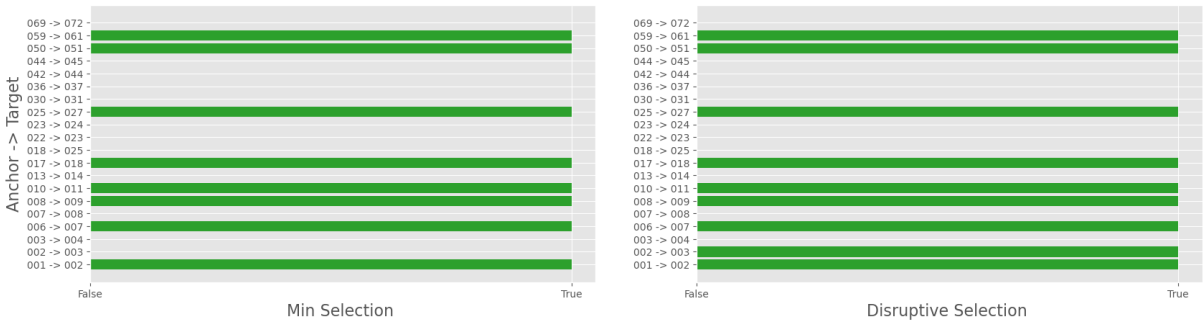


Figure 54: Comparison of the selection of the best patch by minimum score or disruption.

4 Discussion

4.1 Best experiments

The combination of experiments providing the best results is thus, the restricted GA with the Intersection Over Union as loss function and a disruptive selection for the homography match. This method provided the highest number of true positives on the set of 20 test cadastres and also covers as many cadastres as experiment 3.4.2, Figure 55. With these parameters, our method runs in about 7 hours for 20 pairs of cadastres.

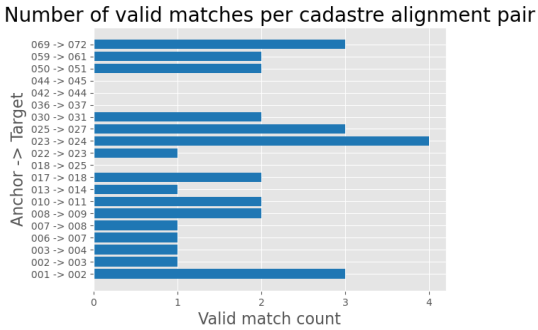


Figure 55: Number of valid matches with IoU and disruptive selection.

We present some alignments obtained with this method:

Figure 56: Alignment of anchor Berney 001 and target Berney 002

Figure 57: Alignment of anchor Berney 002 and target Berney 003

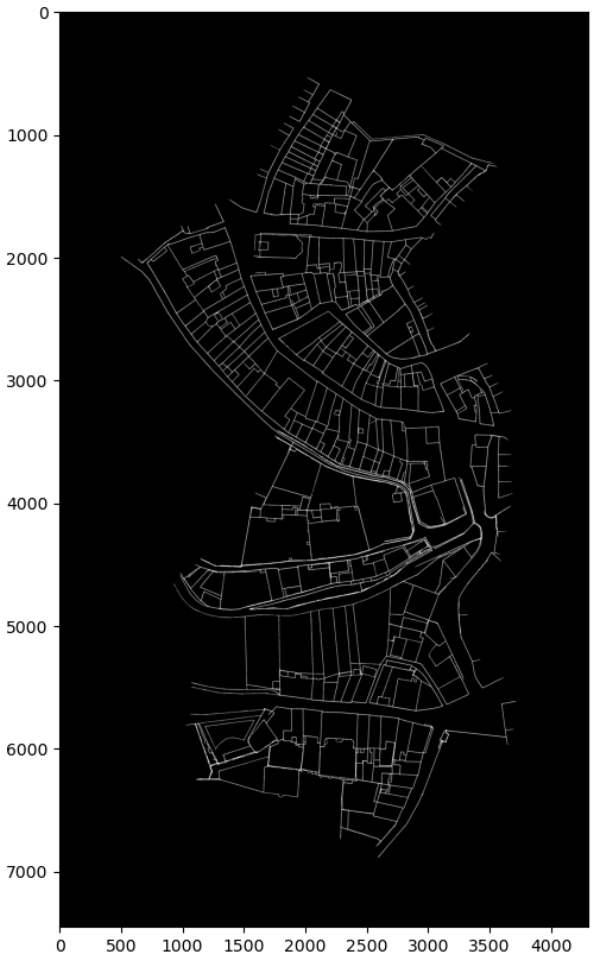


Figure 58: Alignment of anchor Berney 001 and target Berney 009

4.2 Future improvements

Thus, we have tried to think about all the steps: from the choice and implementation of the search algorithm to the similarity measure between two patches and the treatment of false positive matches.
However, there is still a lot of room for optimization. Indeed many steps of our method can be fine-tuned. Our model has a lot of parameters that can make the alignment between cadastres uncertain. Also, the problem of the false positive is not quite solved. With more time, a different approach could be to take a lot more patches, for example in hundred patches, then, performs the alignments as we did but instead of computing the homography between the anchor and the target with only one patch, we could use RANSAC algorithm in the function findHomography from OpenCV. This way, even if a false match has a better loss than a valid match, it could be overcome if there is a sufficient number of valid matches.
Moreover, the implementation of the genetic algorithm remains relatively simple, whereas genetic algorithms are the subject of many studies in terms of selection operators[8, 12] or trade-offs between exploration and exploitation[6]. Other types of search algorithms could also be tested, such as the Artificial Bee Colony[1], which can give better results than GA on specific problems.
The computation time of our experiments represented a significant difficulty in testing more different parameters. We approximate 120 hours of computation on our private computer only for the experiments reported in the report. It may a great improvement to run some instances of GA on a more efficient machine or to optimize the code of this project by using an optimized python library like Parallel Genetic Algorithm Python (PGAPy) which also enables parallelization.

5 Conclusion

In this report, we have presented a method to align two different segmented cadastres with each other based on translations, rotation, and scaling. First of all, it is necessary to perform a semantic segmentation of the cadastres. Then, to align two cadastres, patches are extracted from one and then a genetic algorithm will search for the match in the other cadastre by measuring the similarity between two patches with the Intersection Over Union function. Finally, the best match is selected by disruption and the homography linking the two cadastres can be computed. Furthermore, our method achieved a perfect match in 9 over 20 cadastres.
Our results are not satisfying but they are still promising. We succeed to propose a general alignment method where no prior knowledge about cadastres or human interaction is needed. Our method just needs a pair of cadastre in input and will align them together.
During the development of this project, the genetic algorithm was faced with a very large and steep search space. We managed to take into account the rotation and scaling of the cadastres, but the computation time for some methods was much too long on our personal computers. Thus optimization of the GA implementation or other steps seems to be necessary for future work. The other big problem of our method is the appearance of false positives when patches are defined in not complex enough areas. In the future, it would be interesting to modify our method to overcome this problem. The use of the RANSAC algorithm can be a good idea to find the homography even if false positives are present but the method must be able to perform more quickly. Other search algorithms like ABC could also be used.

References

[1] Muthiah A and Rajkumar R. "A Comparison of Artificial Bee Colony algorithm and Genetic Algorithm to Minimize the Makespan for Job Shop Scheduling". In: *Procedia Engineering* 97 (2014). "12th Global Congress on Manufacturing and Management" GCMM - 2014, pp. 1745–1754. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2014.12.326. URL: https://www.sciencedirect.com/science/article/pii/S1877705814033943.

[2] Steve Coast. *OpenStreetMap - Free editable geographic database of the world.* Community-owned. 2004. URL: https://www.openstreetmap.org/.

[3] EPFL. *City and territory*. 2022. URL: https://www.epfl.ch/schools/cdh/lausanne-time-machine/lausanne-time-machine-initiative/city-and-territory/.

[4] Ares Oliveira Sofia & Di Lenardo Isabella & Tourenc Bastien & Kaplan Frédéric. "A deep learning approach to Cadastral Computing". In: Digital Humanities Laboratory, Ecole Polytechnique Fédérale de Lausanne. 2019. URL: http://infoscience.epfl.ch/record/268282.

[5] Albert-Joseph-Ulpien (1758-1828). Hennet. *Recueil méthodique des lois, décrets, règlemens, instructions et décisions sur le cadastre de la France...* Impr. impériale (Paris), France. Éditeur scientifique. 1811. URL: https://gallica.bnf.fr/ark:/12148/bpt6k96475008/f7.item.

[6] Abid Hussain and Yousaf Shad Muhammad. "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator". In: 6.1 (Apr. 2019), pp. 1–14. DOI: 10.1007/s40747-019-0102-7. URL: https://doi.org/10.1007/s40747-019-0102-7.

[7] Maxime François Jan. *A generic method for cartographic realignment using local feature matching: towards a computational urban history*. EPFL. 2022.

[8] Ting Kuo and Shu-Yuen Hwang. "A genetic algorithm with disruptive selection". In: *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 26.2 (Apr. 1996), pp. 299–307. DOI: 10.1109/3477.485880. URL: https://doi.org/10.1109/3477.485880.

[9] MathWorks. *What Is the Genetic Algorithm?* 2022. URL: https://fr.mathworks.com/help/gads/what-is-the-genetic-algorithm.html.

[10] Rémi Guillaume Petitpierre, Frédéric Kaplan, and Isabella Di Lenardo. "Generic Semantic Segmentation of Historical Maps". In: vol. 2989. 27. 2021, pp. 21. 228–248. URL: http://infoscience.epfl.ch/record/298081.

[11] Marin Piguet and Noe Durandard. *Alignment of XIXth century cadasters*. EPFL - Foundation of Digital Humanities. 2022. URL: http://fdh.epfl.ch/index.php/Alignment_of_XIXth_century_cadasters#cite_ref-notebook_6-0.

[12] S Prayudani et al. "Analysis Effect of Tournament Selection on Genetic Algorithm Performance in Traveling Salesman Problem (TSP)". In: *Journal of Physics: Conference Series* 1566.1 (June 2020), p. 012131. DOI: 10.1088/1742-6596/1566/1/012131. URL: https://doi.org/10.1088/1742-6596/1566/1/012131.

[13] Lucas Rappo. *Historical Context of the Production of Vaud Cadastral Maps (Ancien Régime – 1840)*. EPFL. 2020. URL: https://www.epfl.ch/schools/cdh/lausanne-time-machine/historical-context-of-the-production-of-vaud-cadastral-maps-ancien-regime-1840/.

[14] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. pyimagesearch. 2016. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[15] scikit-image-docs. *Structural similarity index*. URL: https://scikit-image.org/docs/stable/auto_examples/transform/plot_ssim.html.

[16] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[17] Wikipedia. *Rastrigin function*. 2022. URL: https://en.wikipedia.org/wiki/Rastrigin_function.