ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

SEMESTER PROJECT REPORT

AUTOMATIC VECTORIZATION AND QUANTITATIVE ANALYSIS OF XIXTH CENTURY MAPS OF PARIS

KANG, Tianqu

13th June 2022

CHAPTER 1

ABSTRACT

> *Over the 18th and 19th century, the city of Paris is represented by more than 1500 maps and plans. Thanks to state-of-the-art technologies in computer vision, these maps can be vectorized in order to extract the road network. This process relies on CNN-based semantic segmentation and computer vision algorithms for extracting the geometries. In the first part of the project, the student will experiment these techniques. In a second time, the project aims to exploit these data of a great historical value to monitor the evolution of the road network over two centuries, in a period of drastic transformation of the urban fabric. The student will develop quantitative techniques based on graph analysis and network theory to investigate the morphological mutations in the urban fabric. The aim of the project is to better understand the impact of the Haussmanian transformations, and precursory works, on the effectiveness of the transport flow and infrastructure, using a dataset of unprecedented temporal granularity.*

Rémi Petitpierre (IAGS)
Paul Guhennec (DHLAB)

This project analyzes the urbanization process with the help of the historical maps of Paris.

Maps have errors, especially the one published one hundred years ago. One strategy to understand what Paris looks like historically is to align the average maps we have each year. Therefore a sophisticated alignment algorithm is required.

In this project, two alignment algorithms are discussed. The first algorithm with affine transformation improves the alignment result for all the maps. The second algorithm with homography transformation has better performance and sometimes aligns maps perfectly with each other. However, this algorithm only works 50% of the time.

CHAPTER 2

INTRODUCTION

2.1 CONTEXT, MOTIVATION AND GOALS

Haussmanian renovations of Paris took place between 1853 and 1870. Commissioned by Emperor Napoleon III and directed by Georges Eugène Haussmann, the renovation significantly reshaped Paris. Crowded and gloomy medieval streets were demolished. Wide avenues, new parks, and squares were built. This transformation makes Paris a special example in the study of urbanization [1].

More than 1500 maps carefully describe Paris in the 18th and 19th centuries, and we can now easily digitize the maps and analysis the urbanization development in Paris on a computer.

With all these data, the first problem to work on is precisely deriving what Paris looked like in different years. Maps always have errors, no matter how carefully they are drawn. We may have several maps of Paris published in the same year, but there are always some differences between them.

In order to obtain a ground truth map of Paris for each year, one method is to align and take the average of all maps published that year. There are already applications where alignment can be performed manually by matching the corresponding points of two maps. However, this can be time-consuming if we have a large data set for analysis. This report discusses two alignment algorithms that would automate the process.

2.2 DATA SET

The data set contains 340 maps of Paris from the Bibliothèque Nationale de France (French national library, BnF). Each map is labeled by its publishing year, longitude and latitude converge. The earliest map in the data set is published in 1760, and the latest is in 1949. The histogram of the map publishing year is shown below in Figure 2.1.

2.3 PRIOR WORK

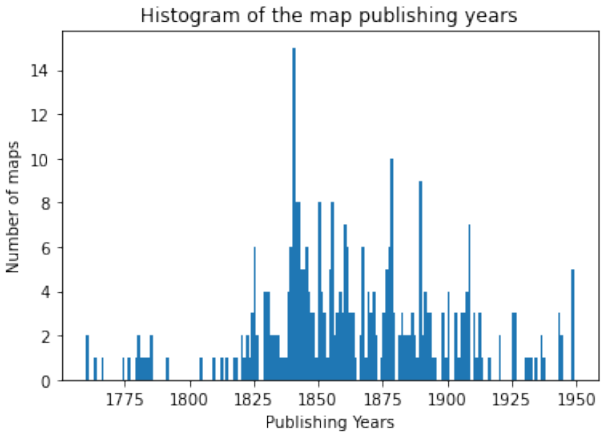
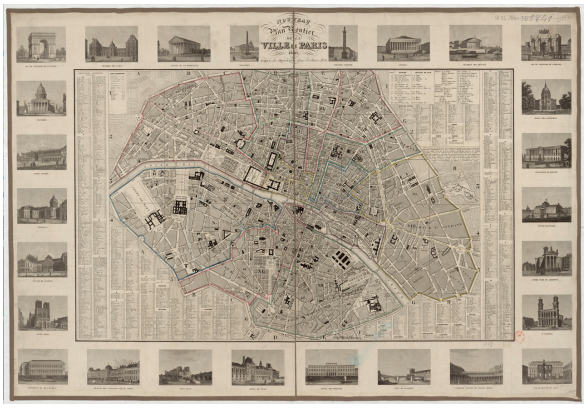


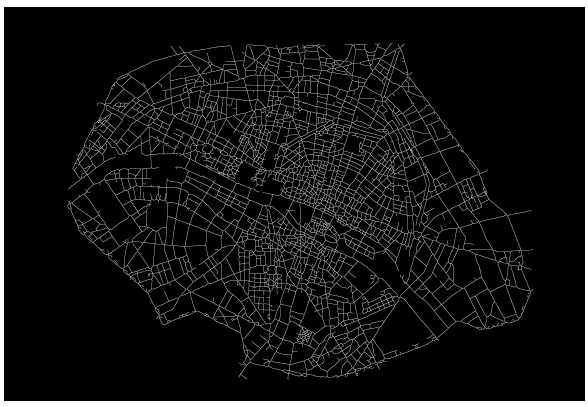
Figure 2.1: Histogram of the map publishing years

2.3.1 VECTORIZATION ALGORITHM

Vectorization of the map can be performed by one CNN-based algorithm [2]. The algorithm output the coordinates of the starting and ending point of all the road segment detected. We draw the vectorized map on a canvas of 10000×10000 pixels. The vectorized map is a 2D matrix. Each entry corresponds to one pixel in the map. The road pixels have a value of 1, and other pixels have a value of 0. Two examples of the vectorized map and its original map are shown below in Figure 2.2 and 2.3. To better visualize the result, the road pixel is scaled to a value of 255. A dilation is applied with the kernel $K = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$.



(a) Original map

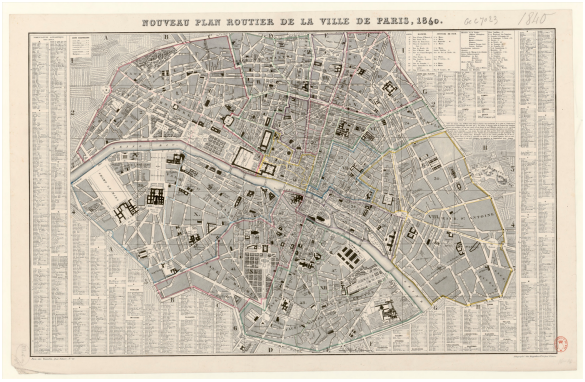


(b) Vectorized map

Figure 2.2: Original and vectorized version of the map *Nouveau plan routier de la ville de Paris 1841*

2.3.2 ALIGNMENT ALGORITHM WITHOUT ORB

One algorithm initially developed for image alignment [3] can be used to align the vectorized map we have. The algorithm tries to align the image with affine transformation, which combines rotation, translation(shift), scale, and shear. The affine transformation has six independent



(a) Original map

(b) Vectorized map

Figure 2.3: Original and vectorized version of the map *Nouveau plan routier de la ville de Paris 1840*

parameters. The algorithm used Enhanced Correlation Coefficient (ECC) Maximization [3] to find these parameters and align the maps.

METHOD

The algorithm is a direct application of the tutorial given in [1], with `number_of_iterations` initialized as 5000 and `termination_eps` initialized as 10^{-10} [3].

Before feeding to the algorithm, the road pixel of the vectorized map is scaled to 255. A dilation is then applied with the kernel $K = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$, followed by a Gaussian blur with `ksize` [4] set to be $[5, 5]$

RESULT

Map *Nouveau plan routier de la ville de Paris 1841* and *Nouveau plan routier de la ville de Paris 1840* are selected as alignment examples.

If two maps are directly overlapped, the result is shown in Figure 2.4.

In comparison, the alignment result of the ECC Maximization algorithm discussed before is shown in Figure 2.5.

DISCUSSION

We chose these two maps as they have similar shapes but can not be aligned by directly overlapping with each other. In this case, the alignment algorithm's performance is more apparent.

Compare Figure 2.4 with Figure 2.5, ECC Maximization algorithm improve the alignment result. Generally, it can not do worse than simple overlapping. However, an affine transformation is too simple to align maps perfectly. In Figure 2.5, even though the bottom left corner is perfectly aligned, there are still misalignments in the upper right corner of two maps.

Figure 2.4: Direct overlap result of two map examples

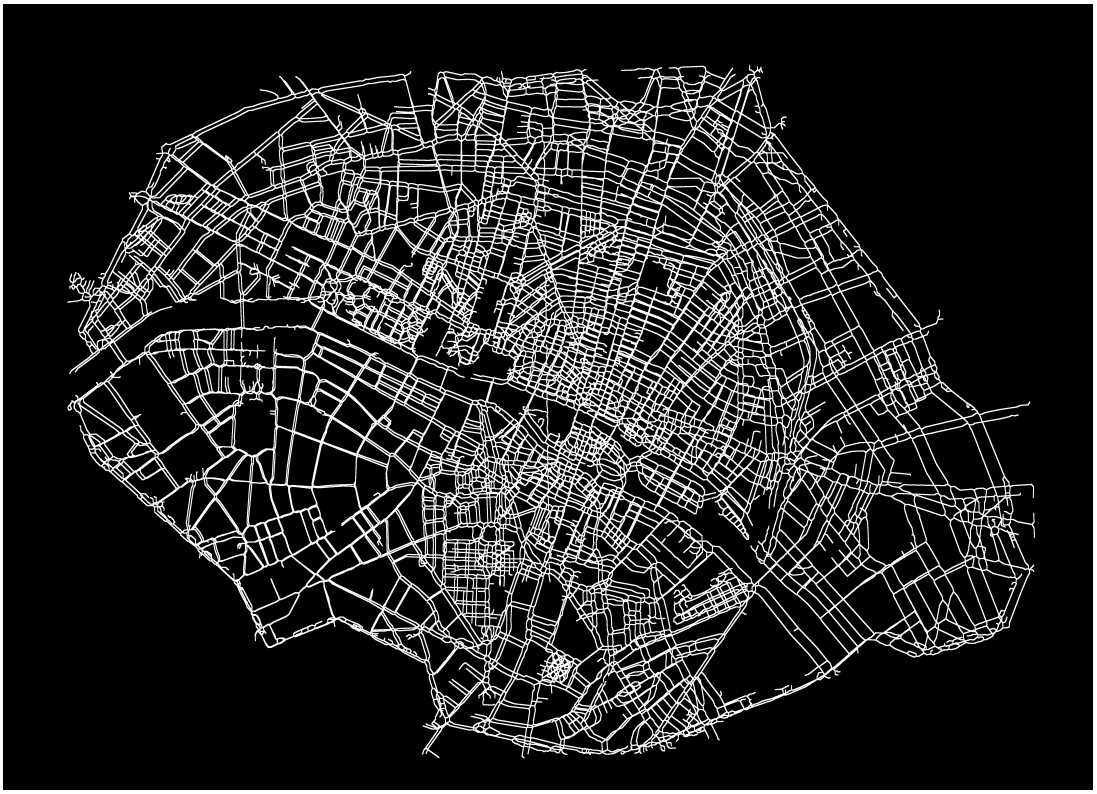


Figure 2.5: Alignment result of the ECC maximization algorithm

CHAPTER 3

ALIGNMENT ALGORITHM WITH ORB

3.1 METHODS

For ease of expression and notation, assume we are to align map B with map A. For better algorithm performance, the road pixel of the vectorized map is scaled to a value of 255. A dilation is then applied with kernel $K = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$, followed by a Gaussian blur with `ksize` set to be $[5, 5]$

An ORB feature detector [4] is first created by setting the maximum number of features to 500. It is then applied to both maps to find their feature points individually. Each feature point corresponds to one descriptor, which encodes the characteristic of this point. Moreover, this descriptor is simply an array of numbers. Most of the time, the same place of the two maps should have a similar descriptor.

Express the feature points in A as a_i, $i = 1 \dots 500$ and feature points in B as b_j, $j = 1 \dots 500$. Denote the position of a_i, b_j as (x_{a_i}, y_{a_i}) and (x_{b_j}, y_{b_j}) respectively.

Each feature point a_i of A is mapped with the feature points b_{a_i} of B, whose descriptor is the most similar. The similarity of the two descriptors can be quantified as their dot product divided by their norm product. Then b_{a_i} can be expressed as:

$$b_{a_i} = \arg\max_{b_j \in B} \frac{\langle a_i, b_j \rangle}{a_i b_j} \tag{3.1}$$

As two maps are already approximately aligned, to make the mapping reasonable, we force a_i to be mapped to feature points in B that are not far away from it. We set the search radius of the algorithm to be r. Under this constraint, the best mapping for a_i, $b^*_{a_i}$ can be expressed as

$$b^*_{a_i} = \arg\max_{b_j \in B_{a_i}} \frac{\langle a_i, b_j \rangle}{a_i b_j} \tag{3.2}$$

where

$$B_{a_i} = \{b_j \in B : (x_{a_i} - x_{b_j})^2 + (y_{a_i} - y_{b_j})^2 \leq r^2\} \tag{3.3}$$

The best map for b_j in B under constrained search radius, $a^*_{b_j}$, is found similarly.

Next, cross-validation is performed to select the mapping whose two points are each other's best mapping. Denote the collection of such mapping to be M, then

$$M = \{(a_i, b_j) : b_j = b^*_{a_i}, a_i = a^*_{b_j}\} \tag{3.4}$$

We feed mapping M to function `findHomography` [5], which would return the homography that can be used by function `warpPerspective` [5] to align two maps. Both functions are available in the Open CV python package.

3.2 RESULT

The alignment result of two maps in section 2.3.2 with this algorithm is shown in Figure 3.1. The search radius returned by the algorithm is 110 pixels. The plot of the number of active pixels versus search radius is shown in Figure 3.2. The corresponding alignment result for each search radius is shown in Figure 3.3

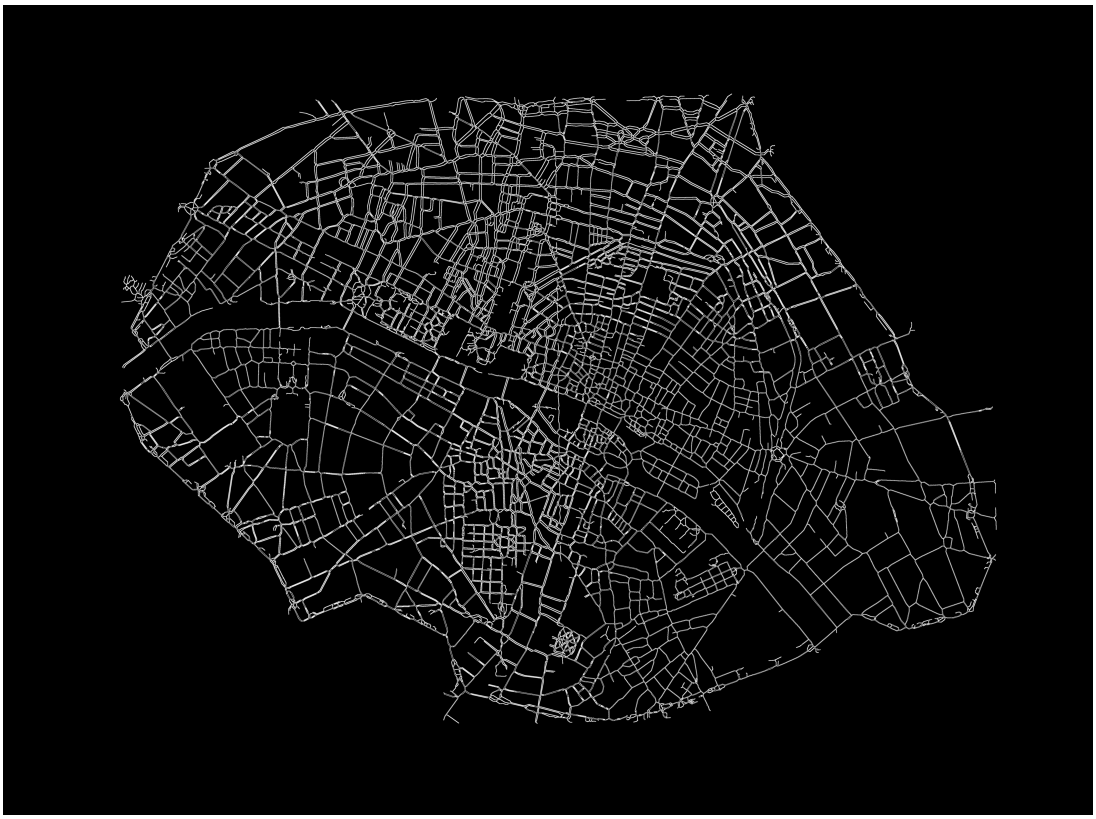


Figure 3.1: Alignment result of the alignment alrorithm with ORB

However, the algorithm does not work all the time. The alignment result of *Nouveau plan routier de la ville de Paris 1841* and some other maps published from 1839 to 1842 is shown in Figure 3.6 and Figure 3.7. While Figure 3.6 contains successful result and Figure 3.7 contains failed result. The corresponding plots of the number of active pixels versus search radius are shown in Figure 3.4 and Figure 3.5

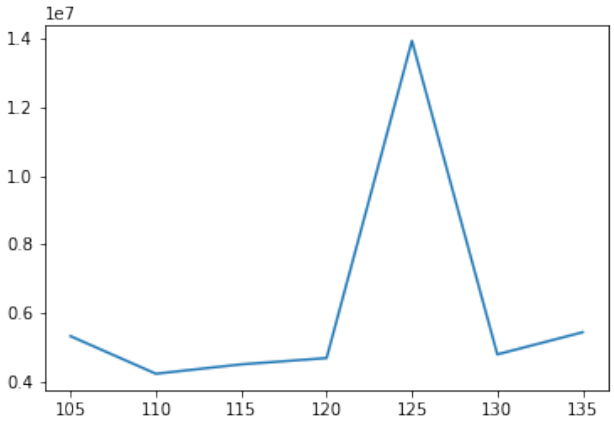


Figure 3.2: Plot of the number of active pixels in the overlapped map versus search radius

Overall, compared with the ECC maximization algorithm discussed in section 2.3.2, the alignment algorithm with ORB improves the alignment result 50% of the time.

The unit of the search radius is pixels. For this experiment, 10000 pixels is equivalent to $0°7'6.826''$ in latitude and $0°10'44.26''$ in longitude.

3.3 DISCUSSION

If the corresponding feature points detected by ORB are successfully matched, as in Figure 3.1 and 3.6, this algorithm can perfectly align two maps. However, we find out the algorithm only works 50% percent of the time. From Figure 3.7 and 3.5, it might be because we only test the search radius within 105 pixels and 135 pixels. Compare Figure 3.6 and Figure 3.7, the algorithm also performs better when two maps have a similar shape or structure.

As the ECC Maximization algorithm improves the alignment algorithm for all the maps, we can use the algorithm in this section as a supplement to further improve the alignment if possible.

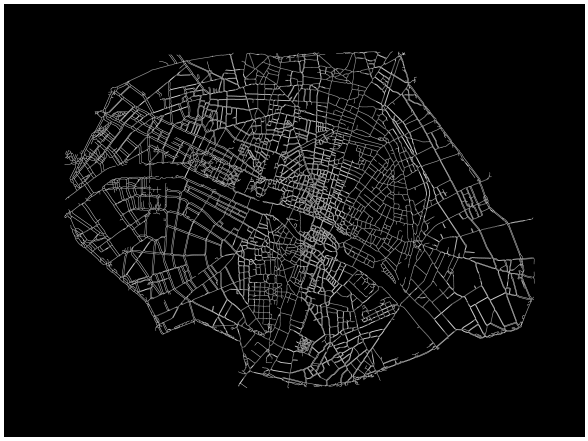
For this project, we align the maps published in the same period to get the ground truth map of Paris, which usually has similar content and structure. Therefore, the shortcomings of this algorithm are weakened in this project.

There are two other parameters that affect the performance of the algorithm greatly:

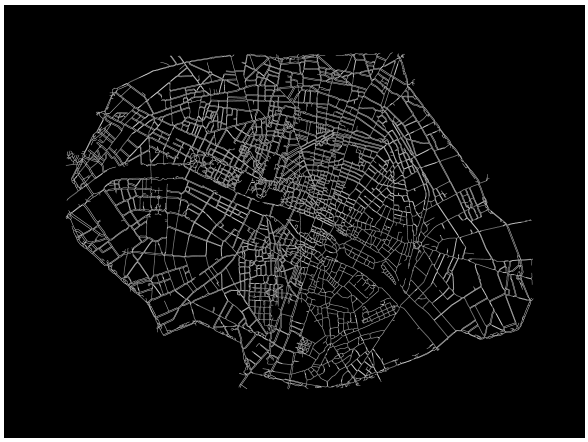
3.3.1 SEARCH RADIUS

Search radius greatly affects the performance of the algorithm. As illustrated in Figure 3.3 and 3.2. With a large search radius, points may be mapped to other unreasonable key points that are far away from themselves. With a small search radius, because of the misalignment, points may not be mapped to the key points representing the same place on another map. Ideally, the search radius should be defined in degree, the same as latitude and longitude. In this case, the scale of the map or the canvas size will not affect the choice of the search radius.

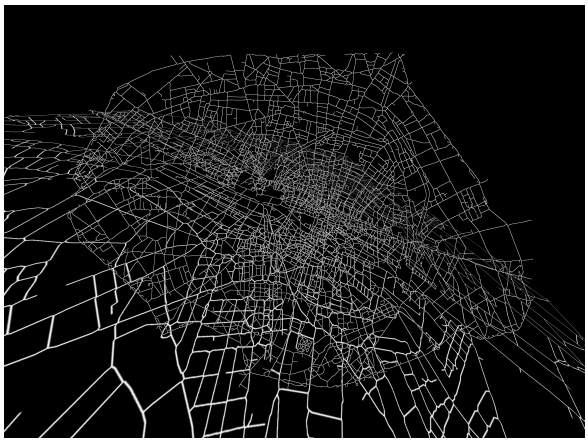
(a) Search radius bring 105 pixels



(b) Search radius bring 115 pixels

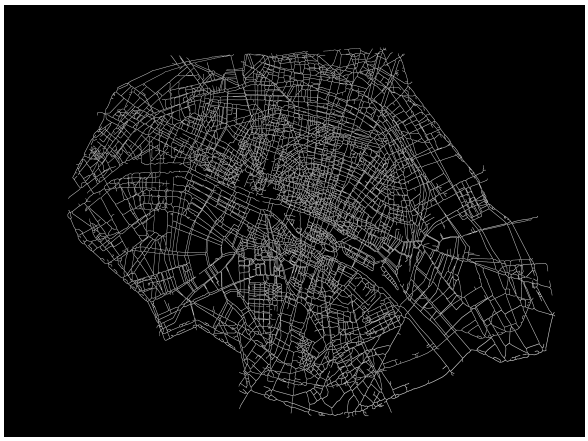


(c) Search radius bring 120 pixels



(d) Search radius bring 125 pixels

(e) Search radius bring 130 pixels

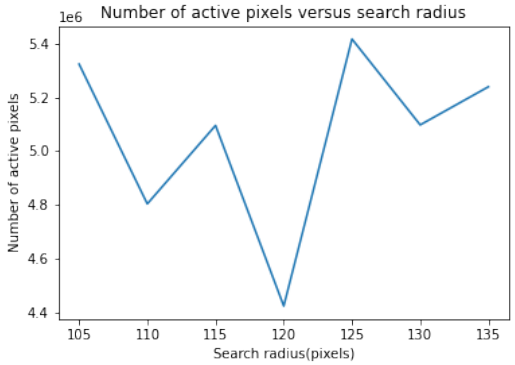


(f) Search radius bring 135 pixels

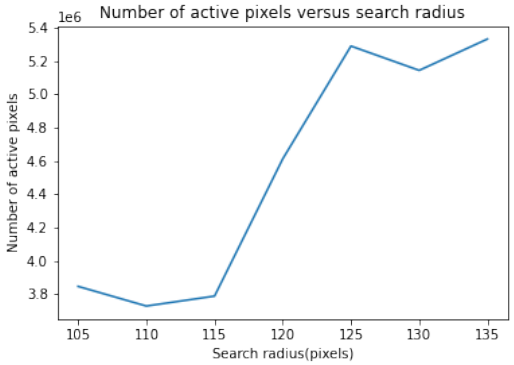
Figure 3.3: Alignment result with different search radius.

There is no global choice of search radius that works for all. It is highly dependent on how misaligned two maps are. In this algorithm, we set a reference search radius, and the algorithm would use the best search radius it can find around this reference.

The performance of the search radius is measured by the number of active pixels of the overlapped map of A and B. A pixel is active if it is not of value 0. Denote number of active pixels of

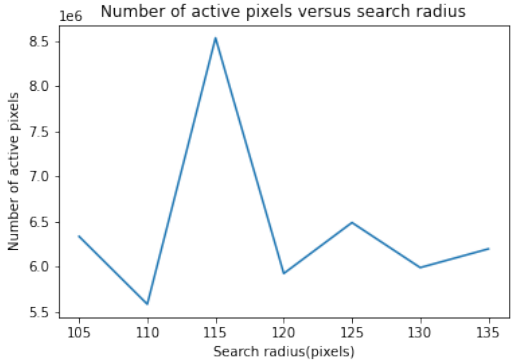


(a) *Nouveau plan routier de la ville de Paris 1839*

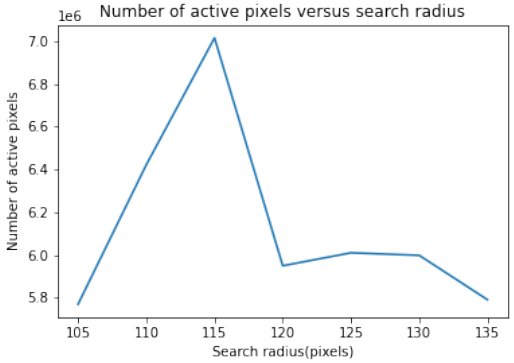


(b) *Nouveau plan routier de la ville de Paris 1840*

Figure 3.4: Plot of the number of active pixels versus search radius of alignment in Figure 3.6, each plot is labeled by the other map name



(a) *Plan itinéraire et administratif de la ville de Paris 1839*



(b) *Plan de Paris 1842*

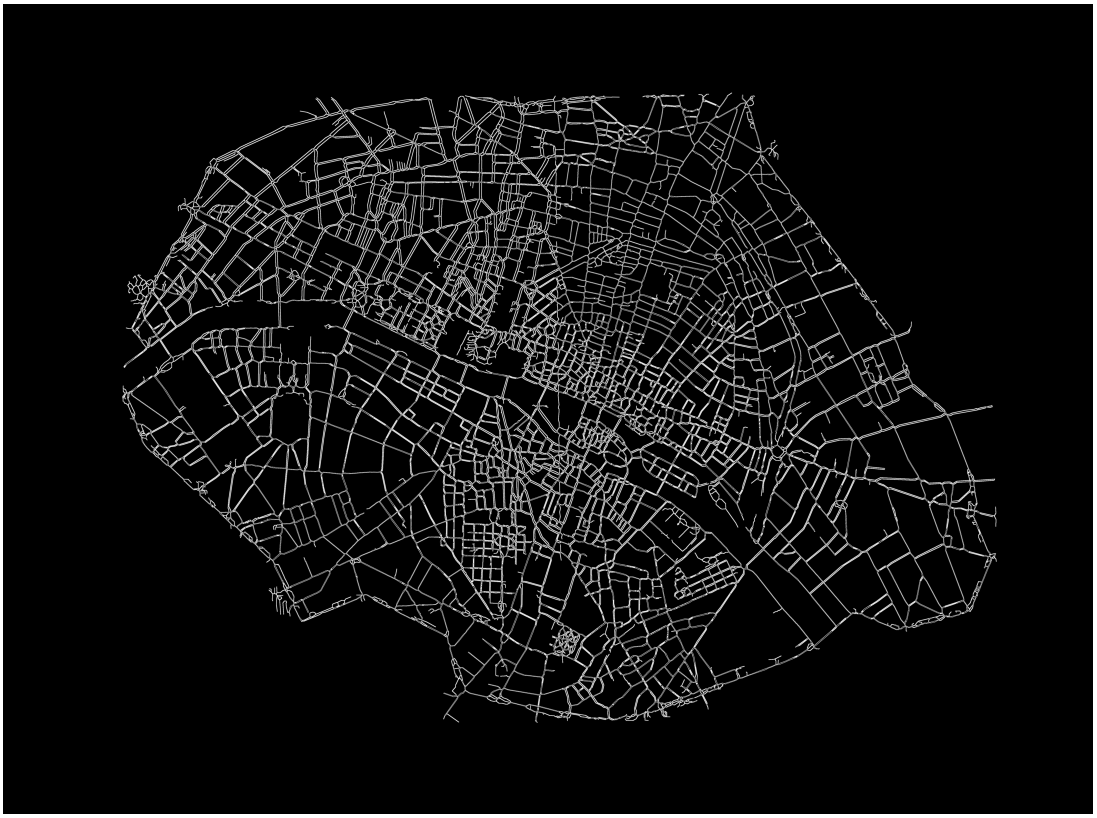
Figure 3.5: Plot of the number of active pixels versus search radius of alignment in Figure 3.7, each plot is labeled by the other map name

A, B, overlapped map as p_A, p_B and p_{overlap} respectively. In the ideal case, if A and B can be perfectly aligned, p_{overlap} would be close to p_A. Otherwise, the overlapped map would have more active pixels. In the extreme case where A and B are not aligned at all, p_{overlap} would be close to $p_A + p_B$.

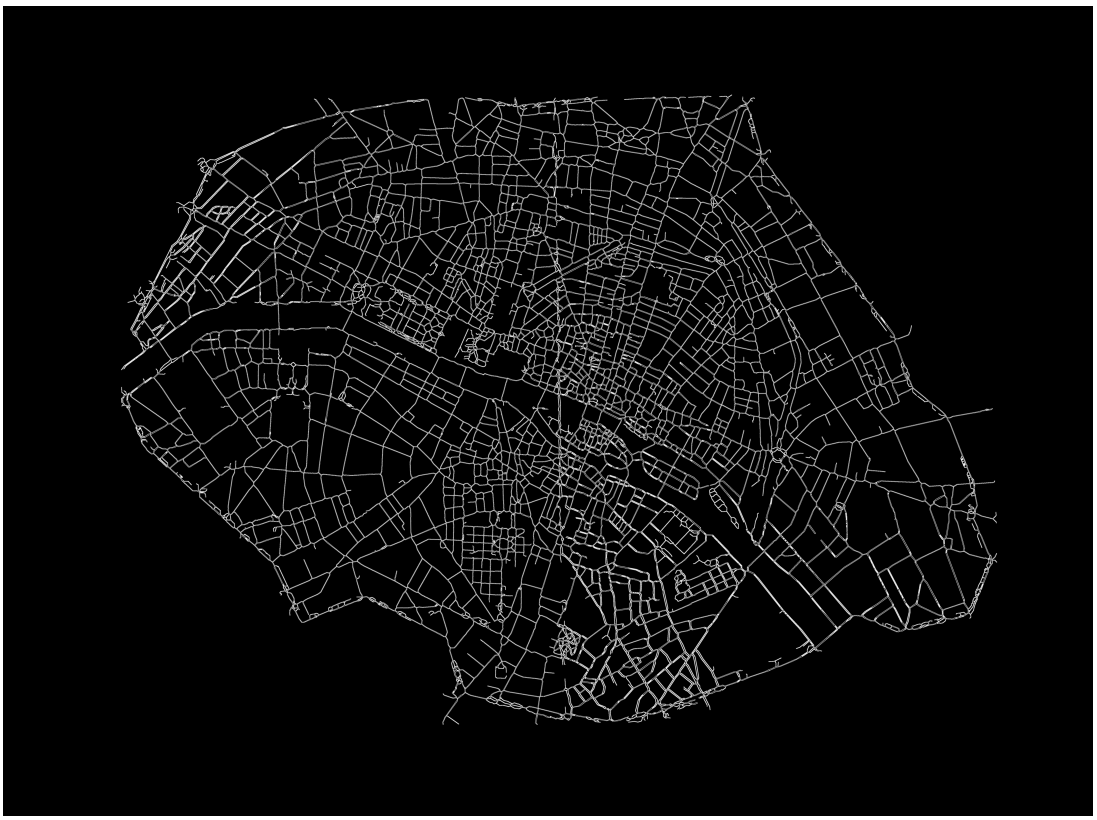
This quantification method is proved by Figure 3.1, 3.2 and 3.6. The search radius with the lowest number of active pixels does have the best alignment result.

3.3.2 DILATION AND GAUSSIAN SMOOTHING

The vectorized map is mathematically a binary matrix. It turns out that ORB usually fails to detect the feature point of such kind of image. We find out that with the same search radius, the map being dilated and smoothed has a better alignment result.

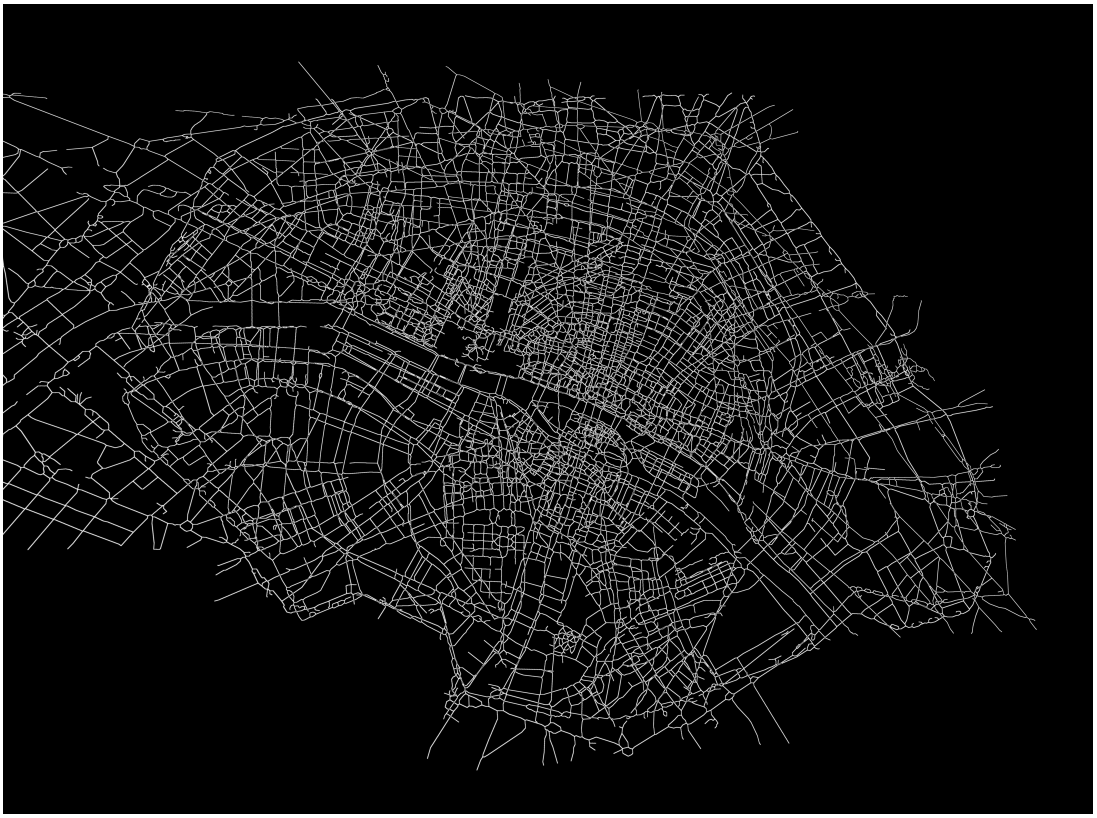


(a) *Nouveau plan routier de la ville de Paris 1839*

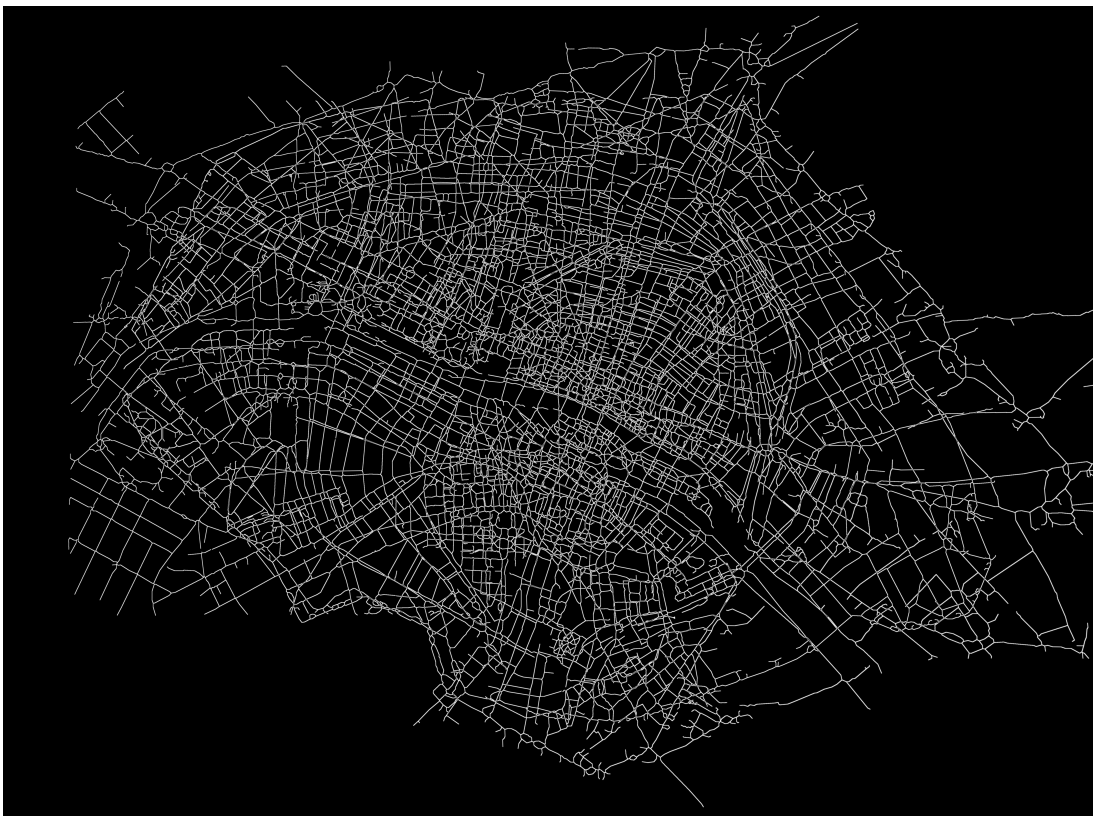


(b) *Nouveau plan routier de la ville de Paris 1840*

Figure 3.6: Successful result of other maps aligned to *Nouveau plan routier de la ville de Paris 1841*, each plot is labeled by the other map name



(a) *Plan itinéraire et administratif de la ville de Paris 1839*



(b) *Plan de Paris 1842*

Figure 3.7: Failed result of other maps aligned to *Nouveau plan routier de la ville de Paris 1841*, each plot is labeled by the other map name

CHAPTER 4

CONCLUSIONS AND FUTURE RESEARCH

In conclusion, two alignment algorithms are reported.

The first algorithm without ORB uses ECC Maximization algorithm to find the parameters for the affine transformation and improve the alignment result overall. The second algorithm uses ORB developed by Open CV to find and match corresponding feature points. The mapping is then used to find the parameters of the homography transformation. This algorithm can perfectly align two maps of similar shapes or structures. However, it only works 50% of the time. Therefore it can be used as a supplement to the first algorithm to improve the overall alignment result. The number of active pixels is used to quantize how good the alignment is. This is proved to be reasonable with the example given. Search radius and dilation with Gaussian smoothing both significantly affect the second algorithm's performance.

There is still a lot to work on for this project.

With the alignment algorithm, one can easily take the average and get the ground truth map of each period. With this, one intriguing topic is to compare the number of road pixels in a neighborhood in different periods. Different averaging methods can also be applied. One method is to only average maps published in the same year. Another method is to add maps of the adjacent years but with a lower weight. Search radius is a critical parameter of the second alignment algorithm with ORB. In this report, the discussion and research on the search radius are pretty limited. We initialized the search radius manually and only tested the search radius in a minimal range. The algorithm's success rate might be improved with more in-depth research of the search radius.

BIBLIOGRAPHY

[1] Wikipedia. *Haussmann's renovation of Paris*. Wikipedia. Apr. 2019. URL: https://en.wikipedia.org/wiki/Haussmann%27s_renovation_of_Paris (visited on 10/06/2022).

[2] Rémi Petitpierre. *BnF-jadis/projet: v1.0-beta*. Version beta. May 2022. DOI: 10.5281/zenodo.6594483. URL: https://doi.org/10.5281/zenodo.6594483.

[3] Satya Mallick. *Image Alignment (ECC) in OpenCV (C++ / Python)*. LearnOpenCV. July 2015. URL: https://learnopencv.com/image-alignment-ecc-in-opencv-c-python/#download (visited on 10/06/2022).

[4] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000).

[5] Satya Mallick. *Feature Based Image Alignment using OpenCV (C++/Python)*. LearnOpenCV. Mar. 2018. URL: https://learnopencv.com/image-alignment-feature-based-using-opencv-c-python/ (visited on 10/06/2022).