BACHELOR'S DEGREE THESIS

Degree in Data Science and Engineering

Title: Revisiting Graph Sampling for map comparison

Author: Jordi Aguilar Larruy

Advisor: Rodrigo Ignacio Silveira

Department: Department of Mathematics

Month and year: June, 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona
Facultat de Matemàtiques i Estadística
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Facultat de Matemàtiques i Estadística

Degree in Data Science and Engineering

Bachelor's Degree Thesis

Revisiting Graph Sampling for map comparison

Jordi Aguilar Larruy

Supervised by Rodrigo Ignacio Silveira
Department of Mathematics

June, 2021

I would like to express my appreciation to my supervisor Rodrigo Silveira for his guidance and bringing me the opportunity to work closely with him and his fellow researchers in an international research project. Thanks also to Kevin Buchin, Maike Buchin, Erfan Hosseini and Carola Wenk for their help during the development of the project.

Abstract

This project offers a review of the Graph Sampling Method, a metric that computes the similarity between two road networks, commonly used in the context of map construction algorithms. The intuition of this metric is simple: first we sample points from both maps to then match pairs of points, one from each map. A high number of matched points indicates that the maps have a high degree of similarity.

Although this metric has gained popularity in the publications related to map construction, we typically find differences in the approaches and the parameters chosen for the metric, causing that the scores of these publications may not be comparable. Often, the details of the implementation are not provided, which causes that the results will be hardly reproducible.

The goal of the project is to understand the different implementations that can be found in the two steps of this metric, sampling and matching. Moreover, we study the implications of each choice and compare the advantages and disadvantages of each. We also provide a repository with the implementation of the Graph Sampling Method in order to provide a common metric for the map construction community.

Keywords

Map comparison, map construction, map evaluation, graph sampling

Contents

1. Introduction

1.1 Motivation

Accurate road maps are crucial for travel efficiency and safety. Furthermore, with the arrival of autonomous vehicles (AV), having precise and updated maps is a requirement needed in order to provide an adequate service. During the last years, the creation and the maintenance of road maps has had the support of either high cost surveys or the online community in platforms such as OpenStreetMap (OSM), but this volunteer-driven process suffers from significant variability in both the skill level and availability of volunteers in different parts of the world. It is for this reason, and due to the increase of available data from all kind of sources, that new methodologies to create maps automatically and at a cheaper cost have emerged. Two of the most common approaches to create maps are using aerial imagery and using GPS trajectories. This kind of data can be used to generate entirely new sections of the road network, or to improve the accuracy and detect the changes of existing road maps. To achieve these tasks, one of the primary steps is the comparison and evaluation between two maps since it will allow us to track the performance of our constructing algorithms or to find the differences between two maps. This work will focus on the analysis of an evaluation metric for road maps: the Graph Sampling Method.

1.2 Preliminaries

In this section we will illustrate different notions related to the map construction topic to help the reader be more familiar with the different concepts that will be shown during this thesis. Figure 1 shows a good example of one of the main tasks that we have faced a lot of times during the course of this research: looking at the output of the Graph Sampling Method.

Road map: In this thesis we will mention a lot of times *road map*, *road network* or just *map*. They all refer to a *geometric graph*, a special case of graphs that are embedded in the Euclidean plane. Each node of this graph has two coordinates, and the edges are straight lines between two nodes. They are not necessary planar (i.e. edges don't intersect) since it can have features such as bridges. Moreover, they can include more features such as directed edges or turn restrictions.

Ground truth and reconstructed map: The ground truth map is the real geometric graph, provided by observation and measurement. The reconstructed map is the inferred map from GPS trajectories. Note that we will refer to the ground truth map as G and to the reconstructed map as H. In mapconstruction.org examples can be found of both trajectories and ground truth datasets as well as the output of map construction algorithms.

Map construction algorithms: These are the algorithms used to build the reconstructed road maps. In this work we will be working with maps outputted mainly from algorithms that transform GPS trajectories into graphs. This algorithms belong to the area of computational geometry.

Graph Sampling Method: The main topic of this thesis: the Graph Sampling Method. The utility of this method is to compare one road map to another one. It consists of two steps: the first one samples

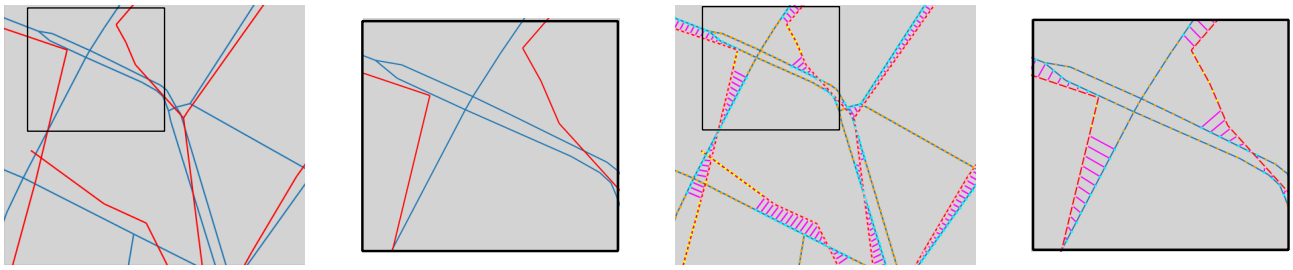


Figure 1: Two road maps (red, blue) and graph sampling results: matched points are connected by pink lines, unmatched points are yellow or orange.

points from each map, the second one matches pairs of points from both maps in a 1-to-1 fashion. The more points matched, the more similar the maps are considered.

1.3 Comparing two maps: previous work

The need of evaluating quantitatively a reconstructed map was acknowledged already in 2012 by Liu *et al.*. In [17], they explain the issue of having only qualitative evaluations that rely on "eyeball" tests and enumerate some reasons to have a quantitative metric. The reasons were the following:

1. Measurement of progress in the field
2. Comparison of many algorithms in one plot
3. Use of machine learning techniques to derive parameter choices from training data

In this same paper, a metric that computes the precision and the recall of a map is presented. It follows a similar pattern than the graph sampling technique: taking samples only in the ground truth, we compute which portion of this samples are within a perpendicular distance threshold m to a (nearest) edge in the reconstructed map. Since we take samples every meter, the total number of samples can be though as the total distance correctly predicted. Finally, to obtain the recall we divide the correctly predicted length by the length of the ground truth. To obtain the precision, the correctly predicted length is divided by the length of the reconstructed map.

This same year, Biagioni *et al.* introduced in [7] the foundations of what will be one of the most popular methods for evaluating a reconstructed road map. He distinguished two different approaches, the global one (which is called GEO) and the local one (which is called TOPO). The idea of GEO is to take samples on both maps to then try to perform a 1-to-1 matching between both sets. In order for two points two be matched, they must fulfill some criteria like a maximum distance between them or a maximum difference in the orientation of their edges. TOPO adopts the same idea but it is implemented in a more sophisticated way: first we generate random points (also called seeds) in the ground truth. Then, for each of this seeds we traverse the edges of the map starting at the seed and taking samples only if we are within a radius from the seed. For the reconstructed map, we do the same procedure starting at the edges close to the seed. Once we have samples in both maps, we proceed to match them following a certain rule and we move to the next seed. The key idea is that for each of these seeds we are only traversing those edges connected with the initial edge(s). This means that if the map has issues with the connectivity (it either has too many connected edges or too many disconnections between edges) we are going to take them into account. The local approach is described in detail in section 3.1.

From then on, several metrics have appeared. Some metrics [1, 16, 5] create paths in both maps that are later compared, both the geometry of the map and the connectivity play an important role in these metrics. Other metrics that rely on distances between trees and graphs such as [13, 4] have been proposed, but many variants are NP-complete. The same happens to other metrics such as the edit distance of geometric graphs [11].

In this paper we will focus on the graph sampling metric [8, 7, 17] since it is one of the most used in the literature.

1.4 Context of this project

Professor Rodrigo Silveira has been deeply involved in map construction algorithms, and after contacting him, he suggested that I could join him and a group of international researchers that were currently working in understanding the implementation of the Graph Sampling method by Biagioni and Erikson [7]. This group is formed by PhD student Erfan Hosseini from Tulane University, Professor Carola Wenk from Tulane University, Associate Professor Kevin Buchin from Eindhoven University of Technology, Professor Maike Buchin from Ruhr University Bochum and Associate Professor Rodrigo Silveira from Universitat Politècnica de Catalunya. While working in a map construction algorithm from GPS trajectories, they raised awareness about the outputs obtained when evaluating their reconstructed map with the ground truth using Biagioni's implementation. The visual comparison between their map and another reconstructed map didn't correspond to the outputs that they were obtaining between the reconstructed maps. It was for this reason that they decided to dive into the analysis of the metric that they were using: what is it taking into account when comparing two maps?

The original publication of Biagioni's metric didn't include a detailed explanation of all the caveats needed to implement this metric from scratch. However, we had the original code of the implementation and together, we spent several weeks debugging and understanding the details of the implementation in order to have an starting point where we could figure out what was the actual behaviour of the evaluation metric and how to compensate it if needed.

This collaboration has brought me the opportunity to be a part of their research about the Graph Sampling method. Moreover, it has successfully concluded with the submission of the paper *Graph Sampling for Map Comparison* to the 29th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2021), where the Graph Sampling method is discussed in full generality. The main contributions of the paper are the proposal of alternatives for the metric, a comparison of ten different map construction algorithms using our unified metric, and making available a *graph sampling toolkit* that contains an updated implementation of the Graph Sampling algorithm as well as an interactive visualization tool.

Finally, I have also been awarded a Santander-UPC Initiation to Research Grant (INIREC) that helps last year students coursing a Bachelor degree or a Masters degree to step up their final thesis and have a first experience carrying out research.

2. Goals of the project

The goals of this project are:

- Identify which are the different steps taken in the design and implementation of the Graph Sampling

metric. Understand the current implementation of the Graph Sampling method developed by Biagioni and Erikson.

- Set global requirements both in the process of discretizing the graph in points and in the process of matching these points between the ground truth and the reconstructed map. These requirements should be the basis to argue whether the evaluation algorithms do a proper job or not. We should write down what we expect from a map comparison method, and try to formalize it.

- Evaluate the behaviour of the current method developed by Biagioni and Erikson as well as other variants of the algorithm, in terms of the score (precision and recall values, variance of these values...), in terms of some properties of the matchings (number of matches, consecutive matches) and in terms of isolated cases. Explain what works as expected and what is not desired.

- Propose a well defined alternative both for the discretization of the graph and the matching of the points that satisfies as much as possible the requirements stated in Goal 1. Analyse this alternative quantitatively and qualitatively.

3. Graph Sampling Method

The graph sampling method in an evaluation metric for maps that has been widely used in the literature [8, 7, 14, 19, 3, 2, 9, 14] among others. As it will be explained in the following section, it discretizes the maps in order to find matchings between pairs of points of both maps.

The reasons why it is a popular method are probably that was one of the first methods to appear, the fact that it makes little assumptions of the embedded graphs, it is an intuitive method where one can easily get the general picture of it, its implementation is not computationally expensive...

Moreover, the scores given by this metric are two of the most common scores in modelling, the precision and the recall. In fact, their definition is also coherent with the map construction task: the precision is the fraction of relevant samples among the retrieved samples (the fraction of correctly predicted roads) while the recall is the fraction of relevant samples that were retrieved (the fraction of roads of the ground truth that we are covering). They are usually merged into the F1-score, the harmonic mean of precision and recall.

In the following section we are going to dive into the implementation of the Graph Sampling proposed by J.Biagioni and J.Erikson.

3.1 Graph Sampling Distance by Biagioni and Erikson

In *Inferring road maps from global positioning system traces: Survey and comparative evaluation* by J. Biagioni and J. Erikson [7] from 2012, three map construction algorithms are compared for the first time using a quantitative evaluation. This paper introduces an algorithm that evaluates both the geometric and topological similarities of two maps in a single metric, and it is of crucial importance since it explains the foundations of one of the standard metrics of map construction algorithms that has been more used in the literature. This method has been called TOPO, and in this section we are going to explain in detail the implementation that was provided to us done by J. Biagioni. During this paper, we are going to refer to this implementation as Biagioni's implementation.

This algorithm is divided in three different parts. Next, a short summary will be given to later explain each step in detail.

In the first part a set of *seeds* is computed. These are points sampled at random locations along the ground truth G map. Once we have this set, for each seed s we select edges of G that are close to s. Starting at the beginning of each edge, we traverse G sampling points at a fixed interval distance and closer than a given threshold from the seed. We do the same for H. Notice that it is possible to sample different connected components of the graphs. Finally we want to generate matches between a sample in G and another in H. To do so, three requirements have to be met: the pair of points must be at a user-defined maximum distance between each other, the bearing difference (i.e., minimum angle between the edges) between their corresponding edges must not exceed a threshold and finally the point in H must be between the 10 nearest neighbours of the point in G. Since one point in G can have multiple matching candidates, a greedy algorithm prioritizes the shortest available match.

3.1.1 Compute seeds

Parameters:

- **Input** *(map)* - The input map in the form of a geometric graph.
- **num_evaluations** *(int)* - Number of evaluation runs of the algorithm.

Procedure:

Firstly it is interesting to notice that the number of evaluation runs is not the same as the number of seeds, since as we will see in section 3.1.2, we discard those seeds in G that are too far from an edge in H. For this reason, the total number of generated seeds is $num\_evaluations \cdot 50$, a high enough number such that one could expect that at least *num_evaluations* seeds will be close to H. To sample random seeds in a way that the number of seeds is proportional to the $len(G)$, where $len(G)$ denotes the total length of G, we assign to each edge an interval $[p, p + len(edge))$ that has the same size as its length, is between $[0, len(G))$ and doesn't intersect with any other edge interval. Then, we sample a random number n between $[0, len(G))$ and check which edge interval comprises n. Finally, we take the as a seed the point along the selected edge that is at a distance $n - p$ from the beginning of the edge. We repeat this process $num\_evaluations \cdot 50$ times.

3.1.2 Sampling the graph:

Parameters:

- **Input** *(map, seeds)* - The input map in the form of a geometric graph. A set of seeds.
- **sampling_interval** *(float)* - Sampling interval. Maximum separation between two samples.
- **sampling_distance_threshold** *(float)* - Sampling maximum distance. Maximum distance from the seed at which we can sample points.
- **edge_distance_threshold** *(Optional, float)* - Maximum distance from the seed at which we are going to consider edges of H to begin the sampling on H. The default value is the same as *match_distance_threshold*, defined at 3.1.3

Procedure:

The following procedure is done separately for each seed. First we select all the edges closer than *edge_distance_threshold* from the seed. Note that for G this set will never be empty since seeds always belong to a point in G, but it can happen for H. In this case, we skip this seed and proceed to the next one. Then, for each selected edge we begin a breadth-first search (BFS) at the input node of the edge or at the closest available point in case the input node is further than *sampling_distance_threshold* from the seed. While traversing that particular connected component, we sample points every *sampling_interval* meters following two restrictions. The first one is that the sampled points must be at a closer distance than *sampling_distance_threshold* from the seed. The second one is that when traversing the graph we must follow segments that lead away from the seed to "de-emphasize unlikely driving patterns and improve robustness". When we are at a distance d shorter than *sampling_interval* from the end of an edge, we will sample points in the following edges beginning at *sampling_interval* $- d$. This may cause that in a intersection with three or more edges, two samples from different edges are closer than *sampling_interval*. Pseudocode is provided in Algorithm 2

3.1.3 Matching:

Parameters:

- **Input** *(Samples_map1, Samples_map2)* - Two sets of samples belonging to each map.
- **match_distance_threshold** *(float)* - Maximum distance to match two samples.
- **bearing_difference_threshold** *(float)* - Maximum difference of bearing (inclination) between the edges where the samples are placed.
- **neighbourhood_threshold** *(Optional, float)* - The sample of H must be among the *neighbourhood_threshold–near* of the sample in G in order to obtain a match. Default is 10.

Procedure:

The matching algorithm implemented by Biagioni *et al.* is in fact a greedy algorithm. As explained previously in Section 3.1, three constrains must be fulfilled: the distance between two matched points must be equal or less than *match_distance_threshold*, the bearing difference between the corresponding edges of both samples must not exceed *bearing_difference_threshold* and the point in H must be between the *neighbourhood_threshold–nearest neighbour* of the sample in G. Note that the bearing of an edge is considered to be between $[0, 180]$ degrees. Moreover, the bearing difference between two edges is the minimum angle between them, thus, it can be at most 90 degrees.

Given that a sample in G can have several candidates in H, the algorithm proceeds to first compute the closest neighbour in H for every sample in G. Note that a point in H can be the pair of multiple points. Then we try to capitalize the closest pair. Given the closest pair, if the sample in H is available, we confirm this match, set the sample in H as used and proceed to the next shortest match. If the sample in H is not available, we select a new pair in H that fulfills all the restrictions and has not been used. Then, we proceed to the next shortest matching. If the sample in G doesn't find any available pair, it is discarded and remains unmatched. The pseudocode of this greedy algorithm is shown in Algorithm 1.

Computing the score. Although in their publication is not clearly specified, we assume that the way to compute the precision and recall is by aggregating, that is, summing all the outputs among all the seeds. This means that the final scores will be:

$$precision = \frac{overall\_matched\_samples}{samples\_in\_H \cdot num\_evaluations} \quad recall = \frac{overall\_matched\_samples}{samples\_in\_G \cdot num\_evaluations}$$

$$F\text{–}score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Algorithm 1: Greedy Matching

Input : Set of samples $S_G \subseteq G$, and $S_H \subseteq H$, parameter k
Output: A 1-to-1 matching $M \subseteq S_G \times S_H$

```
M_init = ∅ // Priority queue, sorted by matched distance
// Create initial 1-to-many "matching"
for all s_G ∈ S_G :
    s_H = closest among k-nearest neighbors of s_G that are within distance (and bearing) threshold
    Add (s_G, s_H) to M_init
// Convert to 1-to-1 matching, prioritizing shortest distances
while M_init ≠ ∅ :
    (s_G, s_H) = M_init.pop() // Pop closest pair
    if s_H not used
        Add (s_G, s_H) to M; mark s_H as used
    else
        new_s_H = closest unused sample among k-nearest neighbors of s_G that are within distance
          (and bearing) threshold
        if new_s_H found // If not found, s_G is discarded
            Add pair (s_G, new_s_H) to M_init
return M
```

3.2 Debugging the code

In order to understand deeply the Graph Sampling Method proposed by Biagioni and Erikson we needed to spend several weeks debugging and testing their code. The main cause why the debugging was a daunting task is because we didn't know beforehand the details of the implementation, so we didn't know what to expect. The paper [7] didn't explain the implementation details related to each step. For example, the greedy algorithm for matching described in Algorithm 1 was not explained elsewhere.

The code provided to us was written in Python and used custom made classes to define maps and their respective nodes and edges. It also used third-party libraries such as *Rtree* to work efficiently with spatial queries like retrieving the k-nearest neighbours of a point. Some of the bugs that we found were that edges were sampled more than once, that in some cases we were sampling only half of the edge, or in other cases we had duplicated edges. The matching algorithm also included some bugs like a mistake when computing the bearing difference between two edges or a bug that made that samples which their closest sample didn't fulfill the bearing condition didn't have a second chance to find another match.

One of the keys to understand and to debug the code was to use a visualization tool to show both maps, the samples of each map and the matched pairs. It can even show the root locations or the initial trajectories. An example of the visualization tool can be seen in Figure 10.

Algorithm 2: Sampling the graph

```
Input  : A graph G and a seed s
Output: A set of samples
initial_edges = edges of G closer than match_distance_threshold from s
for all edge in initial_edges :
    edge.start_ditance = 0
    bfs_queue = ∅ ; add edge to bfs_queue
    // Begin BFS search
    while bfs_queue not empty :
        current_edge = bfs_queue.pop()
        if current_edge not visited :
            mark current_edge as visited
            location = current_edge.start_distance
            // Take samples along current_edge
            while dist(location, s) < sampling_distance_threshold AND dist(location, s) increases :
                Take sample on location
                Increase location by sampling_interval
            if end of current_edge reached :
                for all neighbour of current_edge :
                    neighbour.start_distance = sampling_interval - (current_edge.length - location)
                    add neighbour to bfs_queue
```

3.3 Other implementations

The Graph Sampling method has been widely used in the literature. However, the implementation of this evaluation metric varies from one paper to another. In this section we will go through the different steps of the graph sampling method and explain the different approaches that can be taken along with their implications.

Sampling the map: The two main approaches that the Graph Sampling metric has are global and local, also called GEO and TOPO. The basic difference remains in how the sampling is done, but the general idea is that global only takes into account the geometry of the graph, that is, where the nodes and edges are placed. In contrast, the local approach also cares about the connectivity of the edges.

The global sampling traverses all the edges of the graph in a single run. Then, it will proceed to match all the samples from both maps. On the other hand, the local sampling repeats the sampling and matching procedure for each seed. Given a seed, points are sampled within a radius from this seed, then they are matched and the algorithm is repeated for the next seed.

Given the description above, we can see that the only parameter shared between local sampling and global sampling is the *sampling_interval*, that is usually 5 meters. Notice that *sampling_interval* is in fact the only parameter needed in the sampling process of the global sampling. The only source of variability that this parameter can add is the strictness of this interval when it is not possible to ensure a separation of exactly 5 meters (for example in a cycle of the graph with a length not multiple of 5 or in intersections). Then, one must decide whether to allow separations shorter than *sampling_interval* or not. However, this

scenarios will not have a big impact in the number of samples if the *sampling_interval* is small and the edges of the maps are sufficiently large.

The sampling process of the local sampling approach has however more caveats that can cause significant differences between implementations. We can find the first issues deciding the number of seeds that are going to be used. Some papers do not provide this number, and other papers like [14, 15] state that they place seeds with a separation of 50 meters. In the first case it is common to place the seeds randomly, making the reproducibility of the evaluation unfeasible. A second source of variability related to seeds is how to choose the initial edge to begin with the graph traverse. As explained in Section 3.1.2, Biagioni's implementation chooses all edges within *edge_distance_threshold*, which happens to be the same as *match_distance_threshold*. An alternative can be to choose only the closest edge and not all the available ones. This may be the difference between sampling a single connected component or more than one. When using the local sampling approach one can also decide whether to respect the turn restrictions and the one-way streets, aside from including extra conditions such as that the traversing must lead away from the seed. Lastly, deciding which *sampling_distance_threshold* is going to be used can have a big impact on the connectivity that we are capturing. In the literature we can find values that include $100m$ [10], $300m$ [12, 14, 3, 2, 9, 8, 7] and even $2000m$ [19].

Matching the samples: Once we have two sets of samples (one of each map), we want to match pairs of points from both sets. A matching between a pair of points implies that the point in the ground truth has found a valid point in the reconstructed match which can be considered the same. A match also implies a correct prediction, so it is important to find a matching rule that aims at matching points that could be the same and has the right balance between being too selective or too inclusive.

In this section we are going to define three different rules: Maximum Matching (MM), Greedy Matching (Greedy) and Weighted Maximum Matching (WMM). The third one is a new matching rule proposed in our submitted paper. The first one has been used by [14] and the second one was found in the code by Biagioni and Erikson [7]. We are not aware of any other kind of implementation, in fact, most papers do not explicitly describe how the matching is done, they explain that it is a 1-to-1 matching and that two points are matched whenever they are within *match_distance_threshold*.

Maximum Matching (MM): In this matching, for each sample in the ground truth set we define a set of candidates in the reconstructed map that fulfill the distance condition. Then, following a 1-to-1 match the matching algorithm maximizes the total number of matches. It is interesting to notice that no distinction is made between the samples that are within the set of candidates.

Greedy Matching (Greedy): Computing a maximum matching can be costly and generate solutions where the points matched seem to not correspond to the same point in both maps (see Figure 8). It is for this reason that one can come up with a greedy matching that offers a fast but suboptimal solution. Since the distance between two points is the main factor when it comes to decide whether there should be a match, one could think of a greedy matching that prioritizes the closest available match. In the case of Biagioni and Erikson [8], they implement a matching where the shortest available pair of points is matched provided that they fulfill three requirements: the distance between both points is within *match_distance_threshold*, the inclination between the edges is less than *bearing_difference_threshold* and the point in H is among the *neighbourhood_threshold–nearest neighbours* of the sample in G. This algorithm is thoroughly explained in Section 3.1.3 and described in Algorithm 1.

Weighted Maximum Matching (WMM): Finally we propose a matching that maximizes a function that takes into account the length of the matchings. For a sample in the ground truth, we consider the same candidates as in the Maximum Matching, but this time we add a weight to each pair defined as

$match\_distance\_threshold - ||p - q||$, where $||p - q||$ is the Euclidean distance between p and q. The idea now is that the algorithm has to maximize the total weight of the matches, enforcing short matches. This option can produce as many matches as MM, but we know that they are going to take into account the length between matches. The main difference with the Greedy is that we get rid of the parameter of the neighbours and that in the Greedy algorithm a point will have a unique pair, whereas in WMM, a point may change its pair according to *match_distance_threshold*.

Computing the scores: Precision and recall are the two scores typically used to quantify the results of the graph sampling method. In this context, precision is measured as the number of matched samples divided by the total number of samples on H and recall is the number of matched samples divided by the total number of samples on G. This two scores are usually merged using the F-score, defined as $F = 2 \cdot (precision \cdot recall)/(precision + recall)$.

In global sampling, measuring the precision and recall is straight-forward since we sample a single time the whole map and do a single matching. Hence, we can clearly define the set of matched samples, the set of samples on H and the set of samples on G. However, when using local sampling we have multiple iterations of the "sampling-matching" process. Moreover, there are seeds where we can't find any edge of H close to that seed. In this case two approaches can be taken: the first one is to ignore those seeds that do not have sampled points in H or the other case is to consider them only for the recall. To compute the results of local sampling we have found two different options in the literature. The first one is to compute the precision and recall aggregating all samples over all seeds as described in Section 3.1.3 and the other option is to take the mean of the precision and recall of each seed.

One last detail about the evaluation of two maps that affects equally the global sampling and the local sampling is cropping the ground truth. It is common to find ground truth maps that contain multiple roads that have not been covered by any trajectory or with an area much larger than the one covered by trajectories. In this cases, recall may be substantially lower since it will be very hard for the reconstruction algorithm to find those roads. It is for this reason that in some works the ground truth is cropped in a way that only contains the roads traversed by trajectories. Doing this is not trivial and adds an extra source of variability. Some ways to do this is manually [20] or using map-matching algorithms like in [8].

4. How to compare two maps

In Section 2 were we define the goals we state that we need to define some requirements that should be the basis to argue whether the evaluation algorithms do a proper job or not. These requirements will help us to verify to what extent the evaluation metrics meet them. Otherwise, it is going to be really hard to compare the different methods to evaluate maps.

When comparing two maps, two main factors have to be evaluated: the geometry and the topology. Geometry takes into account if a map is overlaid over the second one, that is, if a road (or a segment of a road) is represented in both maps. Topology cares about the connections of the graph, the degree of the intersections, the cycles of the graph or the paths between two points of the graph. However, before diving into how does the evaluation metric have to deal with this topics we will focus on what is the degree of importance that the evaluation metric has to put into each part of the map. Road maps are composed of streets, highways, avenues, streets where each direction has a different edge... One may think that from a practical point of view, streets that are crowded must be more relevant when comparing two maps, or that attention must be paid to areas with a high density of roads. On the other hand, we could make no

assumptions about each road of the map and just evaluate them in function of their length. Other criteria may be that each edge has to have the same importance regardless of their length, or that each edge has to have a weight proportional to the time it takes for a vehicle to cross it: this way, a very long highway would have the same importance as a handful of short and complex streets of a city center, since, at the end, it takes the same time to complete these two trajectories. In this work, we will make no assumptions about the features of the edges, and we are going to consider that the length is the only thing that matters: one edge that is half the length of another should have half the weight in the final score.

Going back to geometry and topology, how can an evaluation metric take these factors into account? To evaluate the geometry of the map one could think that given a point s^G in the map G, we should identify the point s^H in H that exactly corresponds to s^G (if it exists) and match them. The main problem is that finding s^H is not trivial since the perturbation and noise of GPS traces plus the imperfections of the reconstruction algorithms produces that s^H may not be the closest point of H to s^G. However, one should set a limit to the search area since ultimately each node of a road network has its fixed coordinates in the space, and despite the noise of the GPS traces we should enforce a matching between points that are close in the space. A second key aspect is that hopefully we do not only want to match pairs of points, but we want to consider the map as continuous as possible. This means that ideally we would be able to match sub-edges of both maps or even sets of edges that encompass large regions of the road network. Intuitively, we are willing to find the minimum edit distance between both graphs in a continuous fashion. However, putting this idea into practice is not easy, since taking into account the position of the nodes of the graph is a crucial step in this task. Furthermore, we have to think on how to translate this idea of *merging one map into another* in a single score that will be used to compare different algorithms and do it in an efficient and feasible way that is not computationally expensive.

The final idea is that an evaluation metric should be deterministic and easy to reproduce, otherwise, it will loose a lot of value given that researchers won't be able to use the scores published in the papers to compare with their own experiments.

The following experiments will help us to determine whether the different approaches of the graph sampling method behave according to the conditions stated in this section.

5. Experiments

The aim of this section is not only to describe which experiments are going to be carried out and how are we going to do them, but to give a meaning to each one of them, to understand why are we doing these experiments. During the course of the thesis, I have personally faced a situation that was new to me, doing a *metaevaluation*: "a systematic and thorough evaluation of an assessment methodology" [18]. Realizing this raised two main concerns. The first one was how to assess the performance of an evaluation metric. To do it, first we need to know which aspects do we think that the evaluation metric has to take into account or what is the expected behaviour of the evaluation metric in different scenarios. The second main concern was how to do it. In the world of research it is common to find several metrics that evaluate the performance of a model in a given task, the success or failure of a methodology or whether a hypothesis has been confirmed or rejected. Even for the task of comparing a reconstructed map to a ground truth it is possible to find many types of metrics that range from simple and straight-forward counting of nodes and edges and doing ratios to hours of computation to find the edit distance of two graphs. However, finding the right way to evaluate an evaluation metric is not so extended in the literature and not trivial to find. We need to know what are we asking to the metric. For example one should know if the metric should

fulfill a general property, have a specific behaviour in a predefined test case or meet some constraints in its output. It is for this reason that we are going to use Section 4 to as the main requirements that we think that a comparison metric between two maps has to take into consideration. With this requirements, and with the help of the experiments, we will be able to ultimately find limitations, categorize and distinguish the different approaches related to the graph sampling metric that are used to compare two maps.

5.1 Sampling experiments:

The first set of experiments will be the ones concerning the different ways to sample the graph. With them, we want to understand the implications of choosing local sampling or global sampling.

5.1.1 Connectivity

One of the major claims of the TOPO method is that, as the name indicates, it takes into account the geometry of the graph as well as the topology. This term is a bit general and one may wonder what exactly the topology of a graph is. Instead, we prefer to say that the TOPO method measures the geometry of the graph and its connectivity. Given that for a specific seed, samples are dropped starting at edges closer than *edge_distance_threshold* until a maximum radius of *sampling_distance_threshold*, it exists the possibility that one edge within that radius that is not part of the connected components of the selected edges will not be sampled. If, for whatever reason, our reconstructed map H does include this edge among one of the initial connected components, we will end up with extra samples in this edge that are not present in G. Having more samples in H than in G will potentially cause a decrease in the precision since we won't be able to match those samples that are not present on G. The same example can be applied exchanging the maps: if H has disconnected edges that are connected in G, the recall will be lower since we are not sampling edges that will be sampled on G. This will be useful to punish those algorithms that tend to connect to many edges that shouldn't, or to disconnect edges that should be connected. This is also helpful to take into account features of the map such as bridges, tunnels, intersections of several edges, urban highways...

In Figure 2 we can observe the behaviour of the local sampling. Two almost identical maps (blue and red) are shown, the blue map is a single connected component, while the red map has two separate connected components. The cross ("x") in the figure shows the placement of the seed. We can see that we are only sampling one connected component of the map in red, the closest to the seed, whereas in the blue map we put samples everywhere.

While there is no doubt that to do a complete evaluation of a map at some point you have to take into account the connectivity, we ask ourselves whether this option of merging it with the geometry of the graph is adequate. In particular, we are going to (1) manually build a small test to verify the eventual decrease of the scores when there are issues with the connectivity of the reconstructed map. We want to compare what happens when we evaluate one example with poor connectivity using local sampling versus using global sampling.

Another possible concern about the local sampling is the location of the initial seeds. One can think that seeds are uniformly placed along the length of the graph, but this is not true if seen from a perspective of the area that they reach. Take for example the disk created by a seed in an area with high density of edges: there are high chances that within that disk we will find more seeds. The disks formed by these seeds will overlap, hence, the edges that are in high density areas will be sampled more times. On the other hand, edges that are placed in very sparse areas will have less seeds close to each other, thus, there

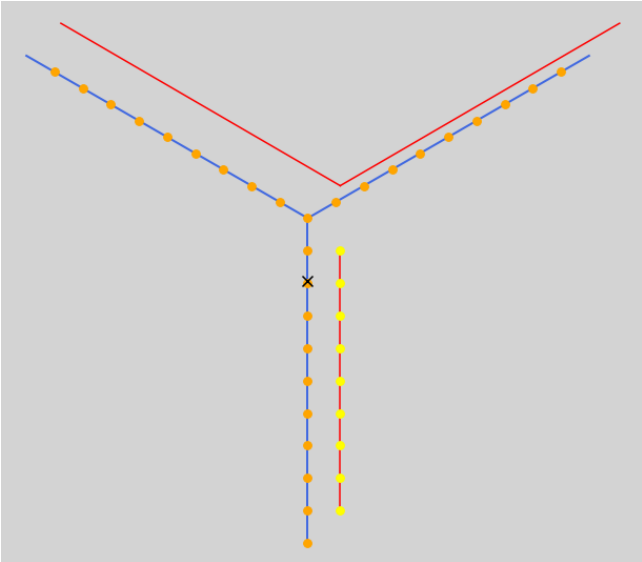


Figure 2: Two manually created maps (blue and red). The cross ("x") in shows the placement of the seed. Orange points are samples in the blue map, yellow points are samples in the red map.

will be less overlapping of disks. We are going to (2) analyze this situation through a predefined example where we emphasize high density areas and low density areas and compare the behaviour of local sampling versus global sampling.

5.1.2 Consistency

Another claim stated in [8] is that global sampling is a *robust* method to compare two maps. However, we know that it adds two extra parameters (*num_evaluations* and *sampling_distance_threshold*) that are not needed in the local implementation. Moreover, the use of seeds introduces two more decisions that may cause variability to the final score: the choice of what to do with seeds that are not close enough to and edge of H and the way to compute the scores once the matching has been done.

It is for this reason that we want to analyze the variance of the final score when we change the value of some parameters. Hopefully we would like to find that the f-score doesn't have too much variability when using different values for the parameters. Otherwise, one could *fine-tune* the parameters of their evaluation metric to achieve the higher available score. This could lead to two situations that are unfavourable to distinguish which are the best models: the first one is that a specific configuration of parameters may cause a significant change in accuracy of the models and eventually change the order of best algorithms. The second one is that if you were to compare your own new model against the previous algorithms in the literature, you could not rely on the scores provided in the previous publications unless they use the very same configuration of parameters that you plan to use (remember that for example in Biagioni's local implementation there are up to 6 different parameters, and several implementation details that are not trivial). This means that you have to obtain the corresponding outputs of the models or even the models themselves to generate the necessary outputs that you plan to compare. It is true that in some publications it is possible to find figures that illustrate the score of the algorithms with various values of a given parameter, but the most common choice is to only change the *match_distance_threshold*.

The first two experiments will compare the robustness of local against global when changing two param-

eters that they have in common that are the (1) *match_distance_threshold* and the (2) *sampling_interval*.

Then we will compute the variance that the local approach has when changing their own parameter that is the (3) *sampling_distance_threshold*.

5.2 Matching experiments:

The second set of experiments will be performed to assess the behaviour of the different matching criteria. We will evaluate three different approaches. The first one will be the *maximum matching* that maximizes the total number of matched pairs, the second one will be the *weighted maximum matching* that generates matches such that the total sum of the weight of each match is maximum. In our case, the weight of each possible match will be *match_distance_threshold* $- ||s^G - s^H||$, where $||s^G - s^H||$ is the distance between the pair of points, one from each map. Intuitively, this prioritizes shorter matches. The third algorithm will be the Greedy matchings, described in Section 1.

5.2.1 Quantitative experiments

As we described in Section 4, the matching process in the graph sampling algorithm should try to relate the areas of the two maps that are meant to be the same, nevertheless, we still need to figure out a way to verify if the respective matchings meet the requirements and properties we are asking for. To achieve this, we have come up with five metrics that are closely related with the properties we are trying to find in the matchings and that will help us to distinguish and rank the different matchings available.

To test the matchings, first we will generate the samples using the global sampling approach (with a *sampling_interval* of 5m) to keep the experiments simple and avoid possible variations that the local sampling may cause. We will compare the three matchings in two different datasets: in Chicago vs the reconstruction of Roadrunner [14] and in Athens vs the reconstruction of Buchin [9]. The maximum matching distance will be set to 50m.

Connected sets: We will define a connected set as a set of consecutive matches. How are we going to define consecutive matches? We will consider that two matches (s_1^G, s_1^H), (s_2^G, s_2^H) are consecutive matches if the samples s_1^G and s_2^G are neighbours in G. The same must happen for the samples in H. Two samples s_1^G and s_2^G are considered as neighbours if, traversing the graph G starting at s_1^G, s_2^G is among the immediate next samples visited. Notice that almost all the samples will have two neighbours, one at each side, but neighbours that are at the end of an edge can have as many neighbours as the number of edges in the intersection. One further restriction is that the two matches (s_1^G, s_1^H), (s_2^G, s_2^H) can't intersect. That is, the line segments that connect each pair of points must not cross. To solve the problem of counting the number of connected sets in an efficient way, we create a graph where each sample is a node, and nodes are connected if they are neighbours. Then, the problem is reduced to counting the number of connected components of this graph.

The number of connected sets gives us a lot of information. As stated in Section 4, we want a matching that relates both maps in a continuous way, ideally we are not just connecting points of two maps, but connecting whole areas of both maps eventually composed of several edges. Measuring the number of connected sets can give us an idea if the matching algorithm tries to look for this correspondences of sections between both maps. We could even look for the size of these connected sets: we could take the mean to not only see how many sets we have but also how big are them, or we could count the proportion of sets with a size bigger than i.e. 5 matches. However, in our experiments we are only going to retrieve

the number of connected sets, and we are aiming for a low number of connected sets, since it will be an indicator that the algorithm matches whole sections.

We provide an example where the connected sets of a matching between two maps are differentiated using different colors for each set.

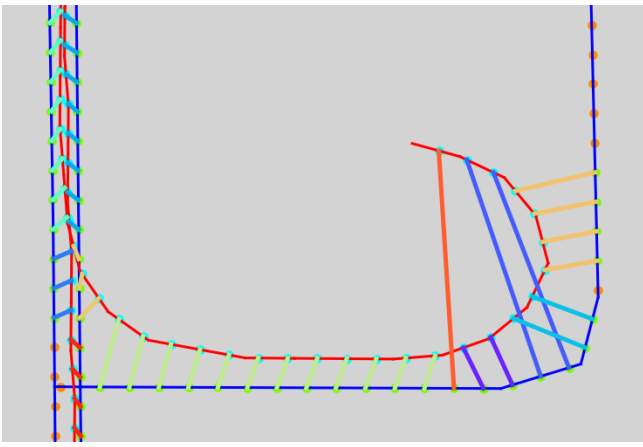


Figure 3: Two maps, one in blue and the other in red. Connected sets are colored in a different color each.

Number of intersections: This metric counts the total number of crosses among all the matches. For two matches (s_1^G, s_1^H), (s_2^G, s_2^H), we will say that they intersect if the line segments between (s_1^G, s_1^H) and (s_2^G, s_2^H) intersect. Recall that we are talking about segments, not lines.

A good matching aims to relate two maps with the smallest possible effort. Given Theorem 5.1, which states that intersections will produce greater distances between the pairs of points, it is logical to think that having few intersections will mean that our matching algorithm is able to connect points in a way that it creates matchings that have minimum length, which can be a good way to justify that our algorithm matches points using the smallest possible effort.

Theorem 5.1. *Given the points (s_1^G, s_2^G) belonging to G and the points(s_1^H, s_2^H) belonging to H, the two 1-to-1 matches from G to H such that the sum of their lengths is the shortest one will not intersect.*

We will prove it by contradiction and using theorem 5.2

Theorem 5.2. *The sum of two sides of a triangle is larger than the third side. (Triangle Inequalty [21])*

Proof. Given the points (s_1^G, s_2^G) and (s_1^H, s_2^H) and two 1-to-1 matches that intersect between these two sets, these two matches can be thought as the diagonals of the quadrilateral that wraps the four points. Given Theorem 5.2, we know that the sum of both diagonals will be greater than the sum of two opposite sides.

Mean length: We have seen that counting the number of intersections is closely related with the length of the matches. Moreover, we believe that a short distance between the two points of a match means a shorter *displacement* of the maps in order to become the same map. So a good way to evaluate if a matching algorithm focuses on minimizing the *displacement* between two maps is by computing the mean length of a match. This metric is much more straight-forward than the previous one where we counted the number of intersections. With this metric we will know the average distance between two matched points,

which not only gives a look at how well the map that you are evaluating is constructed but also will serve as a comparison metric between matches algorithms to determine which is the most tightened one.

Neighbour rank: We are going to define the neighbour rank of two samples (s^G, s^H) as the minimum k such that s^H is between the *k–nearest neighbour* of s^G. This metric has a similar purpose than the metric *number of intersections* and the metric *mean length*: we want to know if our matching algorithm is prioritizing shorter matches over larger matches. However, as the *number of intersections* does, we are not explicitly taking into account the distance of the match, we are evaluating the proximity of the matches looking at their degree of neighbourhood. One can assume that if a match has discarded a high quantity of samples that are closer to the actual matched point, chances are that these two points do not belong to the same coordinate of the map, and our matching algorithm has been too tolerant in the matching. In fact, this is probably what Biagioni had in mind when he added the restriction of the 10–*nearest neighbour* to his matching algorithm. In the section of results (Section 6) we will find if this captured by this metric.

Maximum (bottleneck) distance: Finally we will take into account a metric that searches the distance of the longest match that has been done. This metric will give us an upper bound of the matches produced by an algorithm in a particular pair of maps. It will help us to identify if a particular matching algorithm is really *ambitious* and tries to match at all costs or if on the other hand the algorithm gives a little bit of slack to the *match_distance_threshold* and produces matches that are less close to this threshold.

5.2.2 Qualitative

The metrics described in the previous section will be really useful to obtain insights about the general properties of the matching algorithms and also to be able to compare numerically each matching. However, we are also interested in understanding the behaviour of these algorithms in a specific context. In particular, we are going to generate two small tests that contain really simple shapes and features that could be found in a map and evaluate them using the different matching algorithms. From that, and based on the arguments described in Section 4, we are going to point out what is the correct behaviour of a matching algorithm in that particular situation and which are the advantages and disadvantages that we can observe in the tests.

6. Results

In this section we are going to explain the results obtained in the experiments described in section 5.

6.1 Sampling experiments:

6.1.1 Connectivity

The first experiment that we are going to perform is (1) the evaluation of the behaviour of the local sampling in a predefined test. In Figure 4 we show the same example as in Figure 2 but this time with 50 seeds. Each circle is associated to a seed, and the color indicates the F-score of the seed. In this example we can ignore *sampling_distance_threshold* since it is larger than the graph itself. We can see that the seeds in the vertical edge have the lowest score since they are associated with the shortest edge of the red map. On the other hand, seeds in the top edges have a higher score. We can appreciate that seeds right

in the middle of the graph have the highest score since we are sampling both connected components of the red map. The precision of this example is almost 1, 0.983 precisely. However, the recall is much lower (0.556) although both maps are really similar and occupy almost the same region of the space. In fact, this recall can be computed by hand: we know that 2/3 of the seeds in the blue map (the ones placed in the two upper edges) will have a recall of 2/3 since we will only cover 2/3 of the red map. In contrast, 1/3 of the seeds will have a recall of 1/3. Hence, $Recall = \frac{2}{3} \cdot \frac{2}{3} + \frac{1}{3} \cdot \frac{1}{3} = \frac{5}{9} \approx 0.555$. When evaluating this example using global sampling we obtain a precision of 0.964 and a recall of 0.931. This example already shows that local sampling goes way beyond than the standard definition of recall (ratio of ground truth covered), and it depends on the connectivity of the graph. In particular, we can see that it depends heavily on where the disconnection is. If we have a disconnection in a central node, recall is going to be heavily punished. If the disconnection is in a isolated region we will barely notice it. Of course, it also depends on the *sampling_distance_threshold*, since this parameter will define the neighbourhood that we are going to sample.

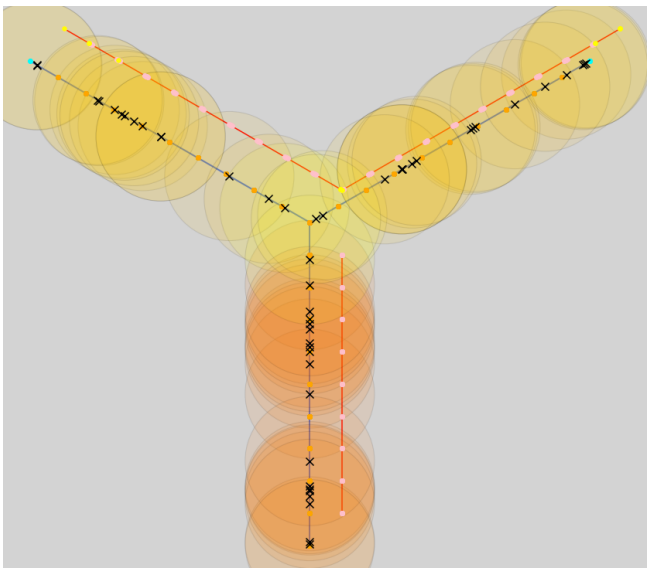
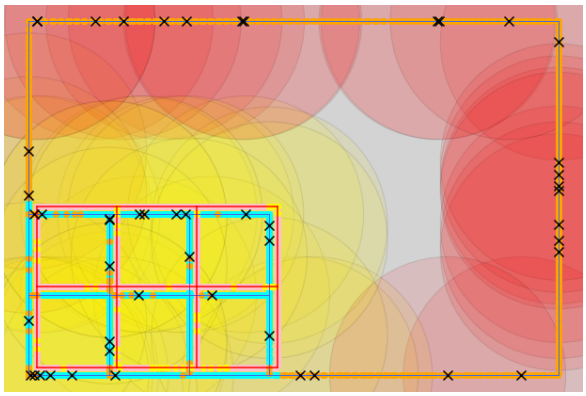


Figure 4: Two manually created maps (blue and red). The crosses ("x") show the placement of the seeds. Orange points are samples in the blue map, pink points are samples in the red map. The disks illustrate the F-score of each seed, the brighter (yellow) the higher the score.

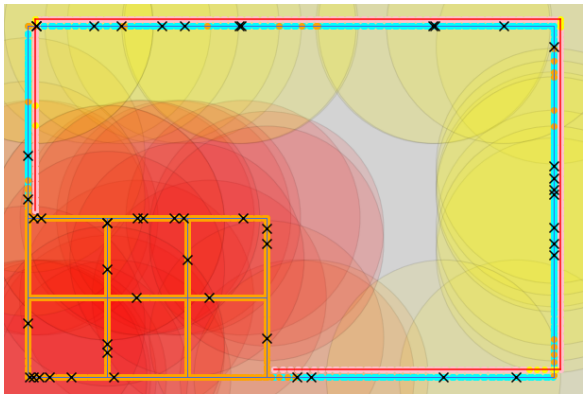
Density issues (2) are evaluated in Figure 5 and Table 1. Among the three figures in Figure 5, the only thing that changes is the position of the *map 2*. In Figure 5a, *map 2* is over the grid (the dense zone), in Figure 5b the reconstructed map is over the big cycle (the sparse zone) and in Figure 5c the reconstructed map is distributed among the dense area and the sparse area, in particular it is placed over two of the six squares of the dense area. In the plots, the crosses represent the seeds and each seed has its corresponding disk of radius *sampling_distance_threshold*. The color of the disk is associated to the F-score of the particular seed. The brighter (yellow), the higher the value of the Fscore of the seed.

Table 1 shows the different scores obtained for each example using three different approaches described in Section 3.3. In this example we are not interested in the precision since it is almost 1 in all the cases, instead we are interested in the recall. Looking at the GEO evaluation, we can see that the recall value is always around 0.5. In fact, the examples have been built in a way that the reconstructed map has half the length of the ground truth in all the cases. If we look closely at the scores provided by TOPO, we can see

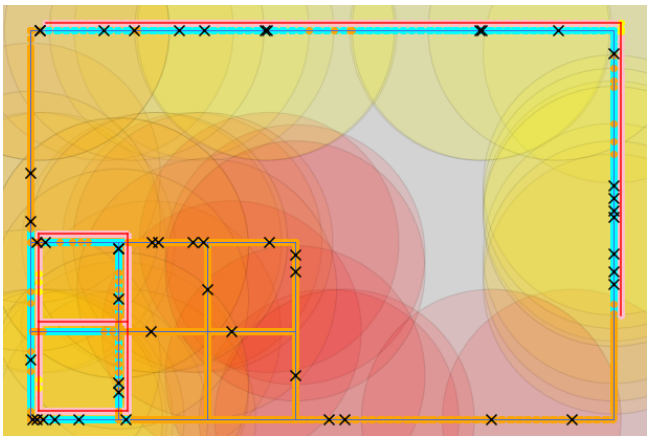
that TOPO (agg.) boosts the recall in *Dense* 5a and drops it in *Sparse* 5b. This is because the seeds in the dense areas have roughly two times the samples of the seeds in the sparse areas. Aggregating the seeds causes that those seeds with more samples (in dense areas) will have a higher weight. Note than in *Mix* 5c, this effect is compensated. On the other hand, TOPO (mean) has a lower although noticeable variance in *Sparse* 5b and *Mix* 5c, going from 0.451 to 0.571 in two different examples that have the same length of reconstructed map. While it is true that by taking the mean of each seed we are hiding a bit if a seed has a lot of samples or just a few, the fact that we can't control the overlapping of the disks adds variability to the final score. It is interesting to note that these examples don't even have connectivity issues, a thing that shows that the local sampling is a method that has more variance than GEO anyway.



(a) Dense



(b) Sparse



(c) Mix

Figure 5: Example illustrating the behaviour of local sampling in three different scenarios. Two maps are compared, *map 1* with blue edges and *map 2* with red edges. 50 seeds are shown with a "x". Each seed has a disk of radius *sampling_distance_threshold* associated. The brighter the color (yellow), the more accurate the region of the reconstructed map. A sample matched in *map 1* is shown in cyan and a sample in *map 2* is shown in pink. Sampling distance has been set to $5m$, *match_distance_threshold* $= 15m$ and *sampling_distance_threshold* $= 75m$

In the examples above we have shown that although TOPO does take into account the connectivity of a map, it introduces a lot of variance that makes the interpretability difficult. On the other hand, with GEO we have a clear knowledge of what precision and recall means. Their meanings, the ratio of correct

Chicago	Dense			Sparse			Mix		
Algorithms	precision	recall	F	precision	recall	F	precision	recall	F
GEO	0.988	0.493	0.658	0.988	0.493	0.658	0.994	0.493	0.659
TOPO (mean)	0.971	0.512	0.67	0.986	0.451	0.619	0.995	0.571	0.726
TOPO (agg.)	0.97	0.698	0.812	0.982	0.27	0.424	0.995	0.508	0.673

Table 1: Precision, recall and F score of the three examples in Figure 5 computed using three different approaches.

predictions and the ratio of covered ground truth respectively, are respected.

From Figure 5 we can also deduce that as the name indicates, the local approach is useful to have a perspective of what areas of the map are the ones that should be improved and what areas are the ones that have a high accuracy. Following this intuition, we can build a heatmap as the one in Figure 6 where if we drastically increment the number of seeds, we can obtain an almost continuous evaluation of the map in a local environment.

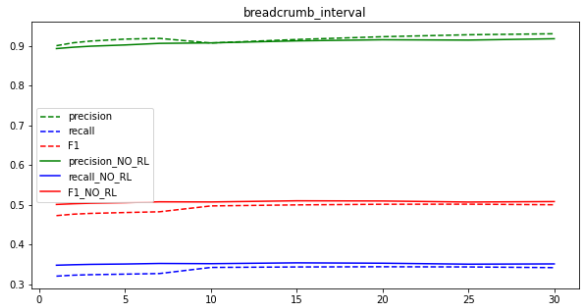
Figure 6: Heatmap of Athens small. 5000 seeds used on the reconstructed map with a radius of 75m. Each one is colored according to its F-score. The brighter the color (yellow), the more accurate the region of the reconstructed map.

6.1.2 Consistency

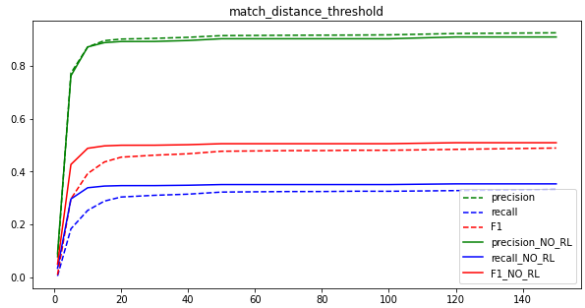
We are presenting the results of the variation of the scores when changing the parameters of the evaluation metric in Figure 7. In Figure 7a we plot the precision, recall and F-score of Chicago vs Roadrunner

using different values for the parameter (1) *sampling_interval*. We have differentiated the global approach (continuous line) and the local approach (dashed line). We can observe that the local approach exhibits a little bit more of variability, specially around 5m and 15m. In Figure 7c we also see a comparison between GEO and TOPO but this time with different values of the parameter (2) *sampling_distance_threshold*. We are able to see that when using the local implementation, recall takes a little bit of extra time to converge than in the global implementation.

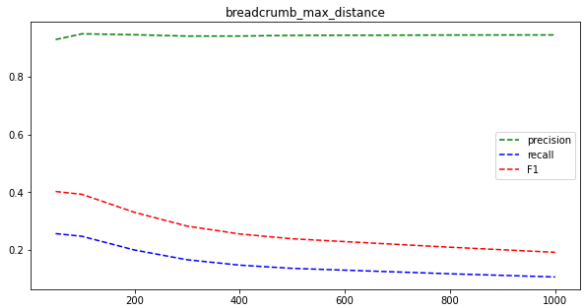
In Figure 7c we can observe that the recall decreases to half when going from 100m to 1000m. This results make sense, since the bigger the disk, the bigger the amount of ground truth that we are going to include in our evaluation.



(a) Scores of GEO and TOPO with varying *sampling_interval*



(b) Scores of GEO and TOPO with varying *match_distance_threshold*



(c) Scores TOPO with varying *sampling_distance_threshold*

Figure 7: Plots showing the variance in the scores of GEO vs TOPO when varying the *sampling_interval* in 7a and the *match_distance_threshold* in 7b. Plot 7c shows the variance in the scores of TOPO when changing the *sampling_distance_threshold*. The maps evaluated were Chicago with Roadrunner using 15m as *match_distance_threshold* = 15m, *match_distance_threshold* = 75m, *sampling_interval* = 5, *num_evaluations*=500

6.2 Matching experiments:

6.2.1 Quantitative

Tables 2 and 3 show the results of the different metrics described in Section 5 about some properties of the matching. Luckily, the behaviour of the different matchings is preserved regardless the dataset, so in

the following paragraphs we will not mention which table are we referring to unless we point some specific value.

Chicago	Without bearing			With bearing		
Metrics	MM	WMM	Greedy	MM	WMM	Greedy
#matches	3,649	3,516	3,433	3,571	3,475	3,405
#sets	1,319	214	173	1,271	186	127
#intersections	8,204	0	19	8,379	37	26
Mean length	45.39	4.32	3.67	45.54	4.18	3.65
Mean neighbour	24.8	1.51	1.16	24.57	1.45	1.13
Variance neighbour	125.99	4.84	0.74	121.43	5.20	0.56
Longest matching	49.99	49.23	49.23	49.99	49.23	49.23

Table 2: Metrics of three matchings with two variations comparing Roadrunner vs Chicago cropped.

Athens small	Without bearing			With bearing		
Metrics	MM	WMM	Greedy	MM	WMM	Greedy
#matches	5,325	5,260	5,172	5,256	5,152	5,014
#sets	2,540	514	497	1,890	305	267
#intersections	9,552	0	57	8,774	152	70
Mean length	44.46	9.46	9.14	45.01	10.01	9.40
Mean neighbour	18.11	1.93	1.56	19.02	1.88	1.37
Variance neighbour	66.98	5.89	2	64.32	7.51	1.19
Longest matching	49.99	49.96	49.96	49.99	49.58	49.96

Table 3: Metrics of three matchings with two variations evaluated on Roadrunner vs Athens small.

At first glance, the biggest insight that can be found is the disproportion between the results of MM and the results of WMM or Greedy. Despite the fact that MM only has around 4% more matches than the other two algorithms, the number of sets, intersections, the mean length and the mean neighbour is way higher than the other two metrics. This is not surprising, and shows the main difference that WMM and Greedy have with respect to MM: they care about the length of the matchings. With MM we are not enforcing any requirements on the length of the matching, thus, it is predictable that it will have long matches. In fact, although the difference is really subtle it is interesting to see that MM is right on the edge of the limit of the matching distance since what MM prioritizes is the number of matchings, while WMM and Greedy leave a bit more of space.

Another surprising fact is that WMM has 0 intersections when implemented without bearing. This was both promising and unexpected, but it has an answer. As stated in Theorem 5.1, no intersections will be produced if we are aiming for the smallest sum of the length of all the matchings, and this is exactly what we are doing with the WMM: the weight of each possible match is inversely proportional to the length of the match, so if we want to maximize the overall weight we will enforce overall short matches that will not intersect. Of course, when adding the bearing we stop considering some of the samples nearby, then, it is possible that we produce intersections between incompatible samples.

One more interesting fact is the similarity between WMM and Greedy in all the metrics. However, it is also plausible that the Greedy implementation is more restrictive than WMM. The fact that WMM

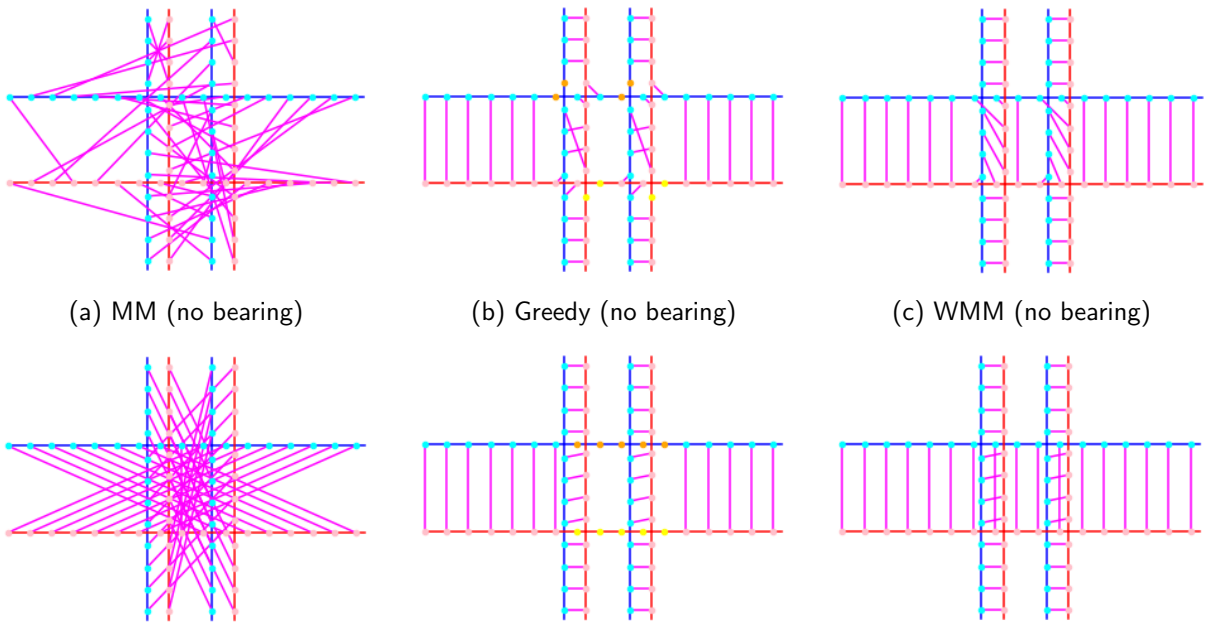
doesn't have a neighbour constraint, causes that two samples that remain unmatched that are closer than *match_distance_threshold* will be matched regardless of their position. This can be clearly seen in the variance of the mean neighbour, WMM has around 5 times more variance than Greedy. Not matching points that are outside the 10–*nearest neighbours* definitely helps to have a control over all the metrics defined above.

Finally, we will talk about the improvement of the number of sets and the mean neighbour when implementing the bearing option. It is interesting to see how in Athens small the number of sets is reduced almost to half when implementing the bearing. This definitely helps to improve the continuity of the matches and to not merge edges that belong to different roads. The fact that the mean neighbour is slightly reduced is also a good indicator that matching samples that belong to edges with different bearing is not a good idea and causes a domino effect that eventually produces matches with further neighbours.

6.2.2 Qualitative

In this section we have manually created two scenarios to show the behaviour of three different matchings without and with bearing.

Figure 8 is comparing the blue map against the red map. They are both composed by one horizontal line and two vertical lines. Note than the horizontal lines of both maps are more separated between them than the vertical lines and that both maps have the same number of samples.



(a) MM (no bearing) (b) Greedy (no bearing) (c) WMM (no bearing)

(d) MM (with bearing) (e) Greedy (with bearing) (f) WMM (with bearing)

Figure 8: Example illustrating three different matching rules, without and with bearing. Two maps are compared, *map 1* with blue edges and *map 2* with red edges. A pair of matched samples is shown with a magenta segment between a sample in *map 1* (cyan) and a sample in *map 2* (pink). Unmatched samples in *map 1* are represented in orange, unmatched samples in *map 2* are represented in yellow. Sampling distance has been set to $5m$, and *match_distance_threshold* $= 50m$.

As it happened in the previous section, the biggest difference is also between MM and the other

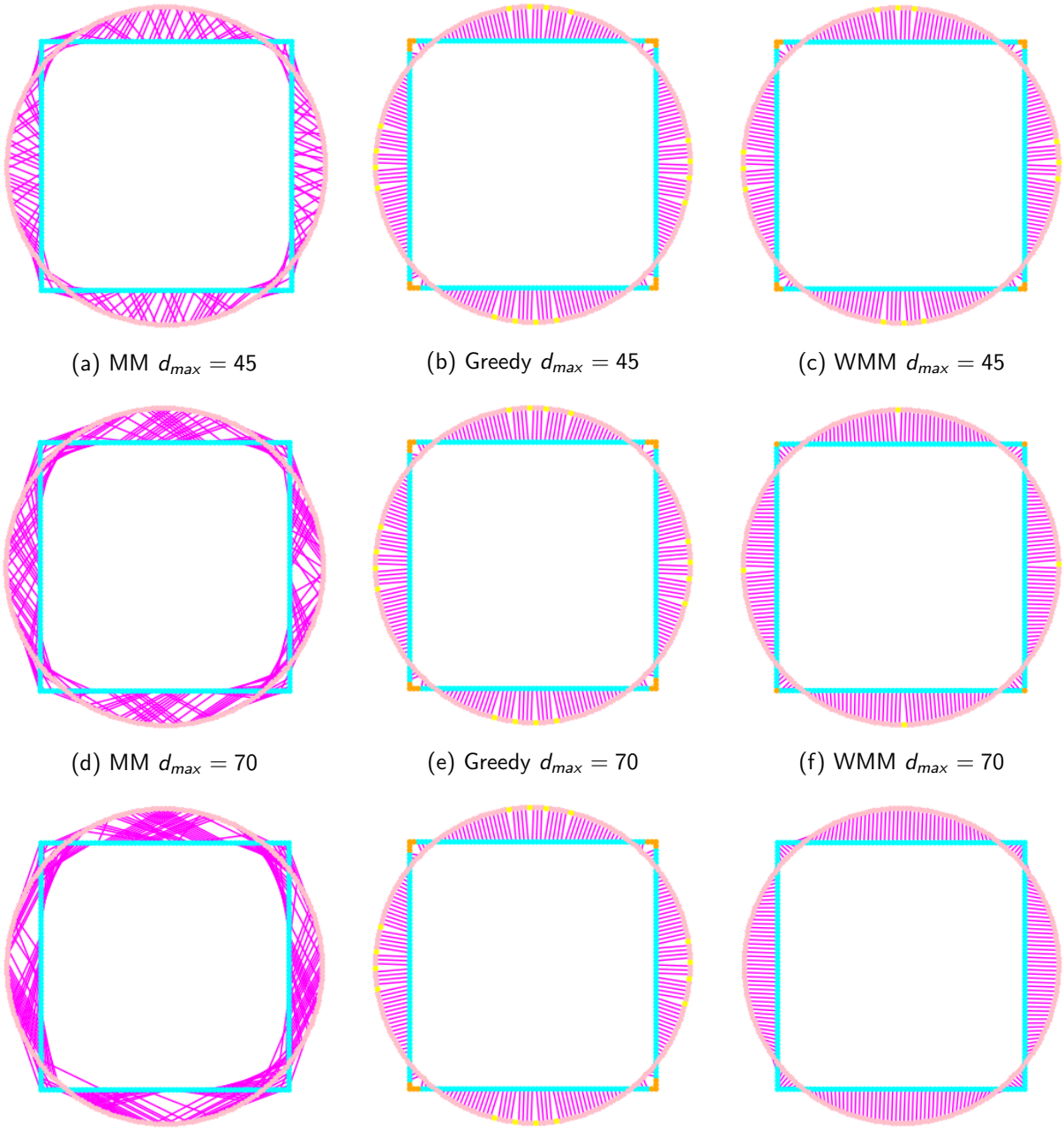
algorithms. In MM (no bearing) (Figure 8a), one could say that samples are matched randomly, and in MM (with bearing) (Figure 8d) there seems to be some kind of pattern but anyway the number of intersections is really high. We do know that this is because MM algorithm doesn't look for the shortest matchings. We can also see that the Greedy leaves some points unmatched. This is because in the dense area of the center we reach the 10-nearest neighbours constraint. This behaviour is a bit unfortunate since it is a bit odd to match the samples in the both ends of the horizontal lines and not to do it in the central area. Finally the WMM (with bearing) shows an ideal behaviour: it matches consistently the horizontal lines and the two vertical lines. We can see how this particular matching creates only 3 connected sets, the optimal in this case. In the WMM (no bearing) we can observe the behaviour that this algorithm has when there is no bearing: no intersections are produced. However, the output in this case is not desirable, there is nothing wrong in producing intersections between matches that do not belong to the same edge. In this figure one could argue that here MM has multiple solutions and that one of them is the one produced by WMM (with bearing) (Figure 8f), so if we would be able to find it, the behaviour of this algorithm would be more coherent with our requirements. While this is true, this involves a second step where we have to find the shortest solution among the optimal solution. Moreover, one could make a test case where there is only a single optimal solution that produces long and forced matches.

In Figure 9, we observe the three matchings with bearing evaluated in the very same example with three different *match_distance_threshold*, 45, 70, and 90. Both maps have the same number of samples.

The main takeaway of Figure 9 is that while MM is able to match all the samples of both maps already with *match_distance_threshold* = 45m (Figure 9a), both Greedy and WMM allow for some unmatched points that punish the metrics. However, the interesting part comes when we increase the *match_distance_threshold*. While WMM progressively matches more points until matching everything at *match_distance_threshold* = 90, the Greedy algorithm has the same output for all three values. Due to the hard constraint of the 10-nearest neighbours, this algorithm reaches a limit (when all your 10-nearest neighbours are matched) that doesn't allow to scale well when increasing the *match_distance_threshold*. The fact that WMM is sensible to this parameter is a good feature, since *match_distance_threshold* is what tells us if two points are close enough to be matched. For example, a distance of 40m with a threshold of 45 meters may seem like a very long match, whereas a distance of 40 meters with a threshold of 90 meters is a shorter match in perspective. We need our algorithms to be scalable with respect to *match_distance_threshold*. Furthermore, one more aspect to take into consideration is the "shortest-first rule" of the Greedy. This produces that the Greedy algorithm always produces the same pairs. In the case that we wouldn't have the 10-neighbour constraint, in this example with *match_distance_threshold*=90 we would have matched the points of the vertex of the square with the points of the top of the circle, creating and incredibly long match. On the other hand, when WMM matches all the points of this example it produces a very nice relation between the square and the circle.

7. Conclusions

In this work we describe the different approaches taken to compare two maps when using the Graph Sampling method. We focus on the differences between the local sampling and the global sampling: the first one involves several parameters, making its implementation and the reproducibility a little tricky. Moreover, we question its robustness proving that the scores can highly vary when using different sets of parameters and that care must be taken when dealing with maps with different densities since the areas where disks overlap the most can have an influence in the score. Although it is true that this metric does



(a) MM $d_{max} = 45$ (b) Greedy $d_{max} = 45$ (c) WMM $d_{max} = 45$

(d) MM $d_{max} = 70$ (e) Greedy $d_{max} = 70$ (f) WMM $d_{max} = 70$

(g) MM $d_{max} = 90$ (h) Greedy $d_{max} = 90$ (i) WMM $d_{max} = 90$

Figure 9: Example illustrating three different matching rules with three different *match_distance_threshold*. Two maps are compared, *map 1* with blue edges and *map 2* with red edges. A pair of matched samples is shown with a magenta segment between a sample in *map 1* (cyan) and a sample in *map 2* (pink). Unmatched samples in *map 1* are represented in orange, unmatched samples in *map 2* are represented in yellow. Sampling distance has been set to $5m$. The first row computed with *match_distance_threshold* = 45m, the second with *match_distance_threshold* = 70m, the third with *match_distance_threshold* = 90m.

take into account the connectivity of the map, it does it at the cost of losing interpretability. F. Bastani *et al.* show these same concerns in [6], saying that "*TOPO tends to assign higher scores to noisier maps, and thus don't correlate well with the usability of an inferred map. Additionally, the metrics make it difficult to reason about the cause of a low or high score*". It must be said that this metric is a really good tool to measure the accuracy of the map in a local scale as shown in Figure 6.

On the other hand, the global approach improves the variance introduced in the local approach. Moreover, it uses fewer parameters and avoids a lot small details in its implementation. On a negative note, we have to remember that the global implementation does not take into account the connectivity of the graph. However, summarizing the comparison of two maps in a single number is a bit optimistic and it is a good practice to test a new algorithm in a set of different evaluation metrics. Having this in mind, we think that the global graph sampling approach is a good metric to obtain an initial comparison between two maps.

Another important aspect presented in this report is the importance of matchings. Although it is a topic that has been overlooked in a lot of papers, we explain why it is important to choose a matching that is coherent with the requirements of a fair comparison between two maps. We show that the greedy algorithm does a good job although the constraint of the 10–*nearest neighbours* doesn't allow to scale the *match_distance_threshold* and may yield to generalization problems if we work with different *sampling_interval* (5 meters is the default) or in other contexts rather than maps where the neighbour constraing may be too selective. We present a *weighted maximum matching* algorithm that is mathematically well defined and exhibits a similar or even better behaviour than the greedy matching at the cost of being computationally expensive.

Overall, this report tries to shed some light over the evaluation of a reconstructed map when using the graph sampling technique. As mentioned in the Introduction 1, back in 2012 the lack of a quantitative metric made it hard for the map construction community to compare and rank their algorithms. Nowadays several techniques have emerged, being the graph sampling technique one of the most popular. However, the lack of a unified version complicates the interpretability of the different scores presented in the publications and forces the researchers to do their own evaluations of all the algorithms they want to compare. To this end, along with the other researchers involved in this task we make available an implementation of the global graph sampling technique in Github (https://github.com/Erfanh1995/GraphSamplingToolkit) as well as a Visualizer (see Figure 10) and a map cropper algorithm in order to ease the job of evaluating a map and consolidate a common tool and implementation that hopefully will lead to more fair comparisons between the different publications related to map construction.

References

[1] M. Ahmed, B.T. Fasy, K.S. Hickmann, and C. Wenk. Path-based distance for street map comparison. *ACM Trans. Spatial Algorithms and Systems*, page 28 pages, 2015.

[2] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms. *Geoinformatica*, 19(3):601–632, 2015.

[3] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. *Map Construction Algorithms*. Springer International Publishing, first edition, 2015.

[4] H.A. Akitaya, M. Buchin, B. Kilgus, S. Sijben, and C. Wenk. Distance measures for embedded graphs. *Computational Geometry: Theory and Applications*, page 101743, 2021.

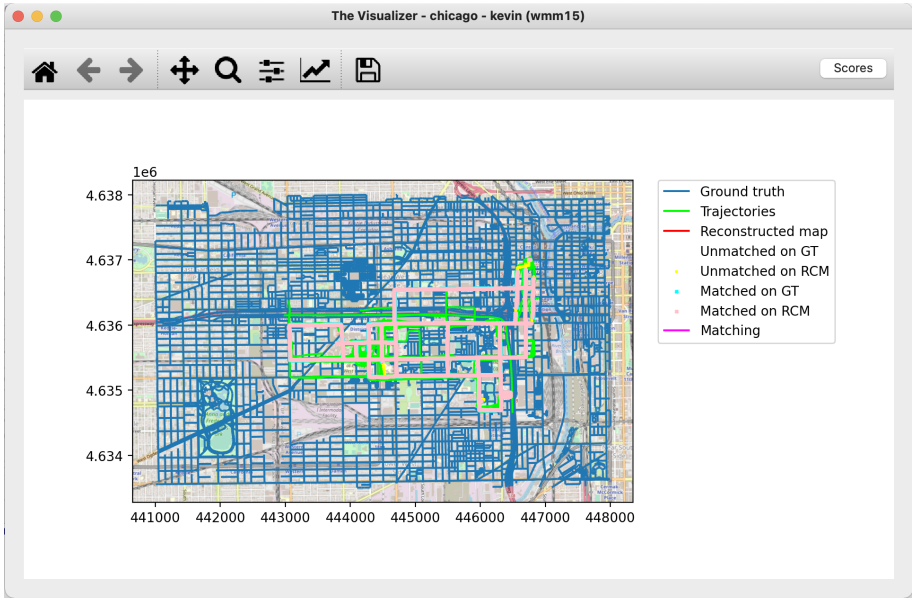


Figure 10: The Visualizer software allows to show the result of the Graph Sampling technique. Aside of the ground truth map and the reconstructed map, we can show or hide the matched/unmatched breadcrumbs, the matched pairs and the input trajectories.

[5] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, pages 262–283, 2003.

[6] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D.J. DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4720–4728. IEEE Computer Society, 2018.

[7] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2291:61–71, 2012.

[8] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proc. 20th ACM SIGSPATIAL GIS*, pages 79–88, 2012.

[9] K. Buchin, M. Buchin, J. Gudmundsson, J. Hendriks, E. Hosseini Sereshgi, V. Sacristan, R.I. Silveira, F. Staals, and C. Wenk. Improved map construction using subtrajectory clustering. In *Proc. 4th ACM SIGSPATIAL LocalRec Workshop*, pages 5:1–5:4, 2020.

[10] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas. City-scale map creation and updating using gps collections. In *Proc./ 22nd ACM SIGKDD*, page 1465–1474, 2016.

[11] O. Cheong, J. Gudmundsson, H. Kim, D. Schymura, and F. Stehn. Measuring the similarity of geometric graphs. In *International Symposium on Experimental Algorithms*, pages 101–112. Springer, 2009.

[12] A. Van Etten. City-scale road extraction from satellite imagery v2: Road speeds and travel times. In *IEEE Winter Conference on Applications of Computer Vision, WACV CO, USA, March 1-5, 2020*, pages 1775–1784. IEEE, 2020.

[13] P. Fang and C. Wenk. The fréchet distance for plane graphs. In *European Workshop on Computational Geometry*, pages 62:1–62:5, 2021.

[14] S. He, F. Bastani, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, and S. Madden. Roadrunner: improving the precision of road network inference from GPS trajectories. In *Proc. 26th ACM SIGSPATIAL GIS*, pages 3–12, 2018.

[15] S. He, F. Bastani, S. Jagwani, M. Alizadeh, H. Balakrishnan, S. Chawla, M.M̃. Elshrif, S. Madden, and M.A. Sadeghi. Sat2graph: Road graph extraction through graph-tensor encoding. In *Proc. 16th European Conference on Computer Vision*, volume 12369 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2020.

[16] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proc. 20th ACM SIGSPATIAL GIS*, pages 89–98, 2012.

[17] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *Proc. 18th ACM SIGKDD*, pages 669–677, 2012.

[18] Pam M.S. N. "metaevaluation," in psychologydictionary.org, 2013.

[19] R. Stanojevic, S. Abbar, S. Thirumuruganathan, S. Chawla, F. Filali, and A. Aleimat. Robust road map inference through network alignment of trajectories. In *Proceedings of the 2018 SIAM International Conference on Data Mining, CA, USA*, pages 135–143. SIAM, 2018.

[20] J. Tang, M. Deng, J. Huang, H. Liu, and X. Chen. An automatic method for detection and update of additive changes in road network with GPS trajectory data. *ISPRS Int. J. Geo Inf.*, 8(9):411, 2019.

[21] Eric W. Weisstein. "triangle inequality." from mathworld–a wolfram web resource.