An extensive analysis and calibration of the Modular Aggregation Algorithm across three categories of for GNSS trajectories data sources

Marie-Dominique van Damme, Yann Méneroux, Ana-Maria Olteanu-Raimond

Université Gustave Eiffel

LaSTIG

Lab on Geographic Information Science for sustainable development and smart cities

IGN — INSTITUT NATIONAL DE L'INFORMATION GÉOGRAPHIQUE ET FORESTIÈRE

ENSG Géomatique — ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

An extensive analysis and calibration of the Modular Aggregation Algorithm across three categories of for GNSS trajectories data sources

Marie-Dominique Van Damme*, Yann Méneroux, Ana-Maria Olteanu-Raimond

Univ Gustave Eiffel, IGN-ENSG, LASTIG

73 Avenue de Paris

Saint-Mandé

France

25 novembre 2024

* All authors contributed equally to this research.
version 2.0

Table des matières

Table des figures

1 Introduction

This technical report aims to complement the conference paper [Van Damme et al., 2024] by providing additional experiments or further details that could not be included in the paper.

1.1 Data availability

— This extensive analysis is based on the work described in [Van Damme et al., 2024] ;

— The open-source implementation of the MIAA algorithm in the tracklib library [Méneroux and van Damme, 2024] ;

— Multi-sensors and multi-canopy traces acquisition with the ground truth

2 Data and material

Since our goal is to reconstruct the geometry of the path from a given subset of GPS trajectories, disregarding the time dimension, each dataset used in this study contains trajectories following the same route, travelling in the same direction, and having approximately the same start and end points. In this study, the data used consist of a collection of datasets that respect these conditions, aligning with those outlined in [Etienne and Devogele, 2014].

The scenarios of greatest interest for studying the aggregation method are paths under forest cover, due to the noise effect, and those containing a series of sharp handling challenges in aggregation. Moreover, our analysis will be carried out by using both synthetic and real trajectories. Thus three categories of data are considered :

1. Realistic Synthetic GNSS Trajectories : These simulated trajectories provide a controlled environment to test the algorithm's performance under various predefined conditions, allowing us to assess its accuracy and robustness ;
2. Multi-Sensor Trajectories : data collected with different sensors (professional, watched and smartphone applications) to evaluate how well the algorithm handles the heterogeneity of sensor data. Multi-sensors trajectories are acquired by following a repeatable data collection protocol, we defined ;
3. Real world data (i.e. crowdsourced datasets), more specifically three crowdsourced datasets categories (terrain-constrained itinerary, infrastructure-constrained and multi-scale) ; this choice is not strictly metrological but rather qualitative : to investigate graphically how the algorithm performs on real typical cases.

2.1 Synthetic GNSS trajectories

In order to assess the metrological performances of the algorithm (*i.e.* its ability to reconstruct accurately the common path followed by all the individual sample trajectories) it is required to test it on a large array of configurations, with a substantial amount of GPS trajectories. This raises two operational problems :

- First, to compare the estimated trajectory with the real route actually followed by the individual samples, it is required to perform a costly and time-consuming on-the-field survey to measure the ground truth. This is especially problematic since moderately to densely covered forest areas are of interest, where the direct use of precision carrier-phase GNSS with Real-Time Kinematic fast and convenient procedure is often impossible. Therefore, ground truth has to be measured with classical surveying traverse, tied to an absolute reference point, possibly located hundreds of meters away ; spending a full day of topometric survey for each case study is not realistic.
- Secondly, for each case study, individual GNSS trajectories must be collected by walking several times along the ground truth route. Because GNSS error is known to be auto-correlated in time [Roberts, 1993], for a realistic acquisition of data, trajectories must ideally be sampled at different times of day, which increases again drastically the effort needed to collect real data.

To overcome these limitations, it was first decided to proceed to extensive experimentation of the algorithms on simulated GNSS trajectories. This enables to simulate as many case studies as needed, with potentially unlimited number of GNSS trajectories, and a readily available ground truth track to compare the results with.

This methodology however, requires an accurate modeling of auto-correlation error of GNSS trajectories, to avoid non-realistic simulations, as depicted for example in Fig. 1, which can also result in topological errors [Bonin, 2002, Vauglin, 1997].

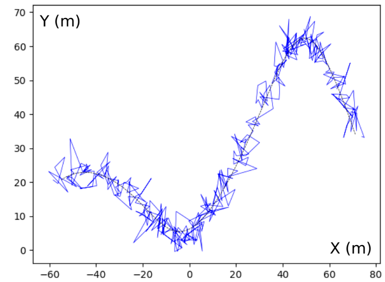


FIGURE 1 – Example of a non-realistic simulation of a GNSS trajectory with a white noise process, completely missing out the true correlation pattern of GNSS measurements.

GNSS errors were then modeled through their covariance function

$$\gamma(s_1, s_2) = \mathrm{Cov}(X(s_1), X(s_2))$$

describing the statistical covariance between positioning errors $X(s_1)$ and $X(s_2)$ at two locations s_1, $s_2 \in \mathbb{R}_+$ (described trough their curvilinear abscissa along the ground truth trajectory).

Further, the error X is supposed to be a second-order stationnary process (hence described only by the difference $s_2 - s_1$), and is modelled to take into account different error components in the GNSS trajectory measurement process, as illustrated on Fig 2.

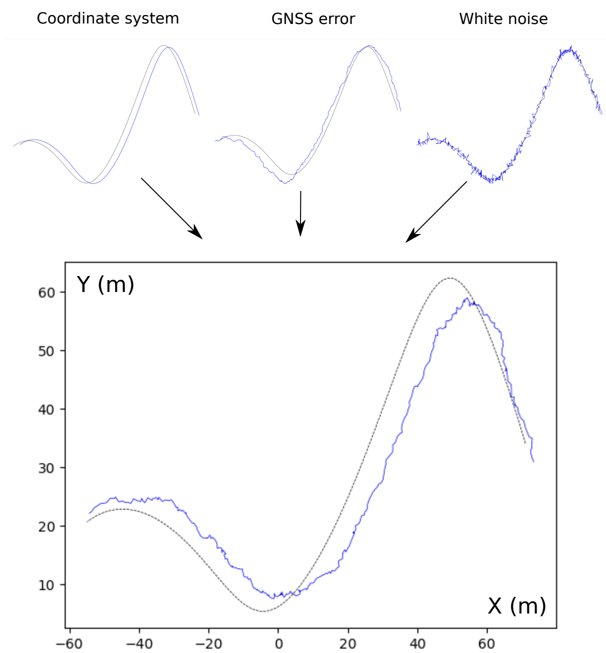


FIGURE 2 – Illustration of the three error components in the GNSS trajectories. From left to right : (1) a long wave-length process describing coordinate system errors, (2) an intermediate wave-length process describing GNSS observation errors (auto-correlated in space and time) and (3) a white noise process (e.g. heat, vibrations, electronic noise).

Generation of GNSS trajectories was done independently on each of the two planimetric components, with a methodology described in [Ripley, 2009] and also employed in [Méneroux et al., 2023] : with a random generator, we sampled n i.i.d. unit-variance and zero-mean gaussian values, compiled in a vector $\mathbf{x}$. It can easily be shown that, for any positive-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$, the random vector $\mathbf{y} = \mathbf{Ax}$ where $\mathbf{A}$ is a Cholesky factor of $\mathbf{\Sigma}$, is a realization of a correlated random vector $\mathbf{Y}$ having covariance matrix Σ. The covariance matrix $\mathbf{\Sigma}$ is formed with $\mathbf{\Sigma}_{ij} = \gamma(s_j - s_i)$ (*Cf.* example in Figure 3).

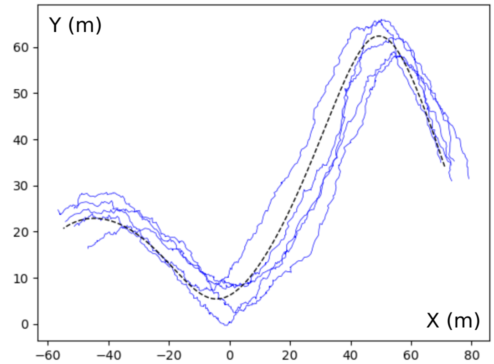


FIGURE 3 – Example of 5 synthetic GNSS trajectories (blue) generated on a common ground truth track (dashed line).

Application : In [Van Damme et al., 2024], the used methodology is the following : for each case study, a reference track is simulated (or extracted from an existing topographic database) and is considered as the ground truth track from which all GNSS trajectories are simulated. The error between the estimated and ground truth track is then evaluated, which in turn, enables to assess the sensitivity of the algorithm to all its parameters. Similar methodologies have been used for example by [Biljecki et al., 2015] and [Zhang and Yang, 2015].

In our experimentation, trajectories have been generated with a noise generated with a 100 m range Gaussian Process and with a 50 cm-amplitude for referencement error, completed by a noise generated with a 5 m-amplitude exponential covariance process (GPS error) ([Grejner-Brzezinska et al., 2005]) with a 50 m correlation scope and completed by 1 m white noise process (vibrations, electronic noise, etc.).

Which translates to the following code using the *Tracklib* library ([Méneroux and van Damme, 2024]) :

```
# Generate a track
tkl.seed(123)
base_lacets = tkl.generate(0.4, dt=10)
chemin = tkl.noise(base_lacets, 20, tkl.SincKernel(20),
                    direction=tkl.MODE_DIRECTION_ORTHO)[::3]
chemin = chemin[80:250]
chemin.scale(10)

# Generate a collection of five identical tracks
N = 5
tracks3 = tkl.core.TrackCollection([chemin]*N)

# Generate noise to get realistic GNSS trajectory
tracks3.noise(0.5, tkl.GaussianKernel(100))
tracks3.noise(5, tkl.ExponentialKernel(50))
tracks3.noise(1, tkl.DiracKernel())

# Visualisation
plt.plot(chemin.getX(), chemin.getY(), color="black", linestyle='--', linewidth=1)
for track in tracks3:
    plt.plot(track.getX(), track.getY(), color="royalblue", linestyle='-', linewidth=1)
```

2.2 Multi-sensors and multi-canopy traces acquisition

To asses the impact of canopy and the sensors on precision accuracy, we defined and implemented a data collection protocol.

First, based on the literature, we identified three types of canopy (i.e. open area, moderate coverage, and heavy coverage). Additionally, we delineated five types of sensors (i.e. mobile phone equipped with Visio-Rando application, Polar GPS device, Garmin GPS device, Keymaze device and professional Ublox GPS sensor chip).

Second, for each type of canopy, the following data collection protocol was defined :

- ∼ Identification of areas without spatial constraints (e.g. bridge, stream, unobstructed)
- ∼ Identification within a rectangle of an Origin/Destination route with moderate winding and approximate length of 300 m.
- ∼ Collect five round trips following the route exactly.

Third, the field work was done by two of the authors of this paper. The placement of the sensors is also relevant according to the literature [Blunck et al., 2011]. To limit this effect, the GPS watches were worn on the wrist, the professional GPS devices were carried in a bag with the antenna positioned externally, and the mobile phones were held in the hand. Data collection has been carried out during the summer season. In total, for the three types of canopy and five sensors, 150 trajectories are collected.

Application : In [Van Damme et al., 2024], our experiments focus only on dense forest 50 trajectories collected with five sensors are shown in Figure 4.

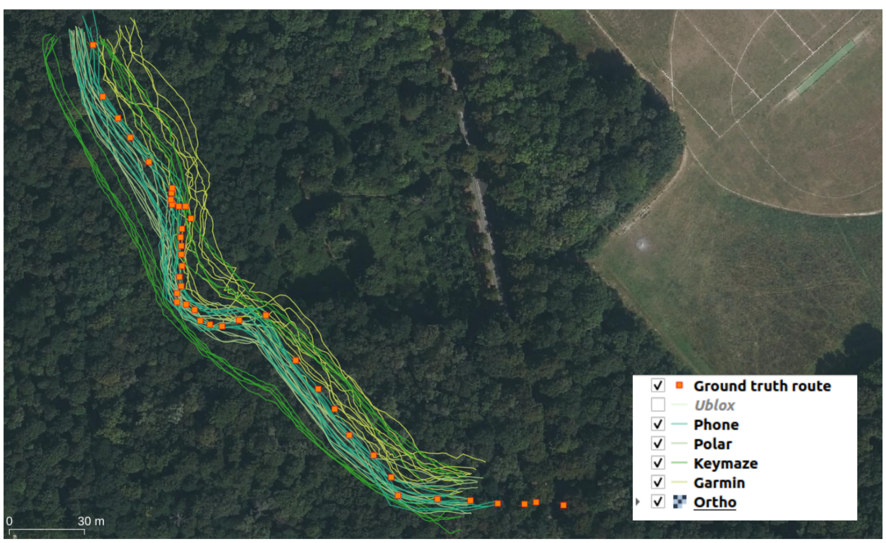


FIGURE 4 – Set of 50 trajectories collected with five sensors in dense forest and ground truth route.

The ground truth route is obtained through a topometric survey conducted by a group of students as part of a topometric project, under the supervision of the authors of this paper and their teachers [Calloch et al., 2024]. Absolute positioning is conducted through GNSS differential static positioning of a set of reference points located in open-sky conditions, between 300 and 500 meters from the surveyed route. Topometric determination of the route geometry is performed with surveying traverse. The output ground truth is sampled with about 42 points (*i.e.* about 1 point every 7 meters) with an absolute positioning accuracy of 5 mm in each 3D axis (1σ) (*Cf.* Figure 4).

2.3 Crowdsourced traces

To validate the results obtained with synthetic GNSS tracks and data acquired according to the proposed protocol, we have chosen to test the algorithms on crowdsourced trajectories, as these will be used to derive pressure and route frequency indicators for the end-users.

To this end, our third experiment was carried out considering tracks downloaded from Visorando [1] and from Wikiloc [2], websites offering donwloading tracks published online by contributors. The GPS sensors used are therefore unknown, as well as data changes, context conditions.

In this experiment, we focused on hiking activity in the mountain area of the Pralognan Valley in the French Alps. Specifically, from all the trajectories, we selected some of them having specific and challenging configurations and spatial constraints : terrain constraints (e.g. ridge, river), infrastructure support (*e.g.* switchback trail) and variations of the shape of the path (*e.g.* a straight line followed by a series of twists and turns with varying distances between them). Note that, in a dataset, all the selected trajectories have the same origin/-destination (*Cf.* Figure 5). Note that, traces may have been previously filtered via a simplification algorithm like Douglas-Peucker. Indeed, all geometries offer the same characteristics like vertex of bends, turns in route. If this was not the case, traces would be more dissimilar in shape.

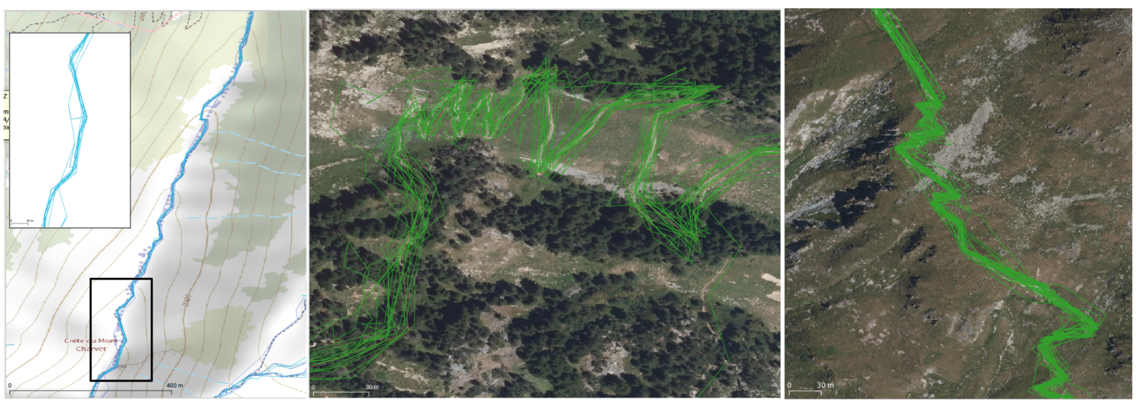


Figure 5 – Trajectories in three differents contexts with spatial constraint : (a) ridge (terrain constraint), (b) series of sharp (infrastructure constraint), (c) heterogeneous trail shape (scale variation)

As the reviewers suggested in [**?**], considering a broader approach based on real world data would be interesting. This could be done by generalizing the first context : a route with terrain constraints is a corridor, covered by an acceptable number of traces, for which we can guarantee they have actually followed the path. For example, in addition to ridge paths, constrained route segments can include trails along the mountainside, trail sections bordered by fences or rock walls. This can be addressed by defining a protocol to extract such trajectories, involving running spatial and topological queries to identify such areas, and then cross-referencing them with traces dataset. To ensure the representativeness of such constraint contexts, we use crowdsourced data issued from collaborative platforms. Eventually, the geometry of the road section stemming from an accurate topographic database is used as the ground truth. Alternatively, ground truth dataset may be collected directly on the field with topometric surveying. The algorithm will be run on all extracted constrained route segments, providing a quasi-metrological validation on real data. Although this approach is relevant, we could not further develop it due to the paper length constraints, so it will be explored in future works.

1. `www.visorando.com`
2. `www.wikiloc.com`

3 Formalization of the original algorithm

This section formalizes the algorithm proposed [Etienne and Devogele, 2014] lacking in a mathematical description to better describe its properties.

Considering a set of trajectories, where each trajectory is defined as an n ordered points :
$\mathcal{X} = (\mathbf{x}_i)_{i=1..n}$, with $\mathbf{x}_i \in \mathbb{R}^2$ and GPS records in a 2D space.

Let d be a distance (Euclidian, Manhattan, etc.) between those points :
$d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}_+$.
Note that heights and timestamps for example, could also be considered with $\mathbf{x}_i,\ \mathbf{y}_i \in \mathbb{R}^3$ or $\mathbb{R}^4$, and defining d accordingly to measure distance between three-dimensional timestamped records.

An **aggregated trajectory**, noted $\mathcal{AX}$, is defined as the best geometric representation of a set of trajectories $\mathcal{X}$ following exactly the same route defined from an origin to a destination :
$\mathcal{AX} = (\overline{xj})$, j=1..m,
where $\overline{xj}$ represents the aggregated points of matched points for the j-th point in the master trajectory.

We defined an **accurate aggregated trajectory**, noted $\mathcal{AAX}$, an aggregated trajectory that optimizes a quality criterion $\mathcal{Q}$, with respect to the ground truth $\mathcal{G}$ (unknown). More formally, $\mathcal{AAX}$ optimizes $E[\mathcal{Q}]$, where E is the expected value of the quantity $\mathcal{Q}$. The main challenge is to define $\mathcal{Q}$, and then to compute it in an approximate and satisfactory way, which is our focus.

One might say, a trace $\mathbf{T}_1$ is a good partial representation of $\mathbf{T}_2$ if $\mathbf{Q}_{T2 \rightarrow T1}$ is minimal with $\mathcal{Q}$ being the square root of the mean of the squared distances between each point in $\mathbf{T}_1$ and its closest neighbor on $\mathbf{T}_2$.

Thus, the quality of the trace $\mathcal{AAX}$ can then be evaluated from the average of the partial qualities :
$\mathcal{AAX} = (\mathbf{Q}_{\mathcal{AAX} \rightarrow G} + \mathbf{Q}_{G \rightarrow \mathcal{AAX}}) \div 2$, with $\mathcal{G}$ the ground truth.

3.1 Algorithm initialization and termination

As mentioned before, this algorithm uses an iterative refinement approach that improve the existing solution at each step, continuously performing multiple matches. And each iteration is composed of three steps : trajectory matching, representative selection on trajectory sections and aggregation of representative points.

But first algorithm initialization consists of choosing a first trajectory, will be called the master trajectory, $\mathcal{R} = (\mathbf{r}_j)_{j=1..m}$. This process involves defining a set of reference positions $((\mathbf{r}_j)_{j=1..m})$, which are all the vertices of the master polyline. Selecting a master trajectory, to start the algorithm, avoids having to match all the trajectories in pairs.

And finally, the iterative process stops when the difference between the aggregated trace and the master trace is below a minimum threshold.

3.2 First step : trajectory matching

The points of each trajectory will be matched with the master trajectory. To be more precise, a matching between the trajectory $\mathcal{X}$ and the master trajectory $\mathcal{R}$ is a bipartite graph $(\mathcal{X}, \mathcal{R}, \mu)$ with no isolated vertex and where μ are the set of matching links. Note that the connectivity constraint imposes that each point in $\mathcal{X}$ is linked to at least one points in $\mathcal{R}$, and reciprocally. We denote by $\mathcal{M}(n, m)$ the set of all possible matching between two trajectories containing n and m points respectively.

An **ordered matching** between trajectories $\mathcal{X}$ and $\mathcal{R}$ is a matching with the additional constraint that there should be no pair of *crossing* links, *i.e.* that we cannot find two links $(\mathbf{x}_i, \mathbf{r}_j)$ and $(\mathbf{x}_k, \mathbf{r}_l)$ with $i > j$ and $k < l$. We denote by $\mathcal{M}^+(n, m)$ the set of all possible ordered matchings between two trajectories containing n and

m points respectively. Note that the definition of an *ordered matching* imposes that $(\mathbf{x}_1, \mathbf{r}_1)$ and $(\mathbf{x}_n, \mathbf{r}_m)$, *i.e.* start and end points of trajectories $\mathcal{X}$ and $\mathcal{R}$ should be matched together respectively.

Matching algorithms based on dynamic programming approach (*Dynamic Time Warping matching* (DTW), *Discrete Frchet distance*) are used to calculate ordered matching $\mu \in \mathcal{M}^+(n,m)$. The L_p-norm optimal dynamic time warping matching between the trajectory $\mathcal{X}$ and the master trajectory $\mathcal{R}$ is given by :

$$DTW_p(\mathcal{X},\mathcal{R}) \in \underset{\mu \in \mathcal{M}^+(|\mathcal{X}|,|\mathcal{R}|)}{\operatorname{argmin}} \sum_{i=1}^{|\mu|} d(\mathbf{x}_{\mu_i(1)}, \mathbf{r}_{\mu_i(2)})^p$$

where, for any index $i \in [1, |\mu|]$, the point of index $\mu_i(1)$ of trajectory $\mathcal{X}$ is linked to the point of index $\mu_i(2)$ of $\mathcal{R}$.

Note that the discrete Frchet distance (the method used by [Etienne and Devogele, 2014]) is similar, but not identical to DTW, since Discrete Frchet distance seeks to minimize the maximum of distances between the two curves, whereas DTW aims at minimizing the sum of these distances. In fact, the discrete DTW distance given in the formula above degenerates to Frchet distance when p grows to the infinity.

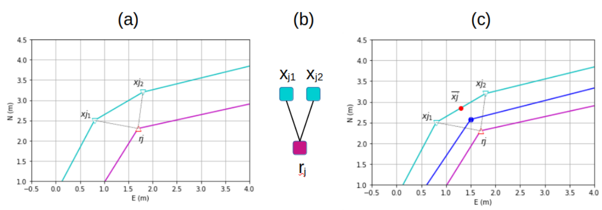


FIGURE 6 – (a) A matching link between trajectory $\mathcal{X}$ and master trajectory $\mathcal{R}$. (b) A connected component of the bipartite graph associated with the matching. (c) representative and aggregation positions for a reference position of the master trajectory $\mathcal{R}$.

A connected component of the bipartite graph ($\mathcal{X}$, $\mathcal{R}$, μ) corresponds exactly to one or more matching links with ni positions of $\mathcal{X}$ and mj positions of $\mathcal{R}$. For example, in Figure 6-a and 6-b, the j-th connected component is composed of vertex (rj_1) from $\mathcal{R}$, vertices $(xi_1$ et $xi_2)$ and the two associated vertex.

3.3 Second step : representative selection on trajectory sections

At the end of matching process, positions of trajectory $\mathcal{X}$ can be linked to many points of $\mathcal{R}$ and reciprocally. Thus, the second step of algorithm iteration is to choose for each connected component previously established, a representative position of each group of vertices from the trajectory $\mathcal{X}$: $\overline{xj}$. This ensures that the final aggregation is not too much influenced. If the segment of the trajectory is composed of several vertices, the representative is calculated from all the positions of the segment, otherwise the single position is chosen. [Etienne and Devogele, 2014], take the center of gravity of the segment. In Figure 6-c, the red circle $\overline{xj}$ represents the center of gravity of the vertices xi_1 and xi_2).

3.4 Third step : aggregation of representative points

At this point, all positions of the master trajectory $(\mathbf{r}_j)_{j=1..m}$ have a unique homologous point $\overline{xj}$ from each trajectory $\mathcal{X}$. The positions of the new merged trajectory are calculated from the median of matching points. The median aggregation operator is better suited to handle outliers. [Etienne and Devogele, 2014] add a constraint in this step of the algorithm : each position aggregated must also be part of existing trajectory positions, so as not to be located in an unlikely place.

At last, we iterate using the aggregated trajectory as the new master trajectory. The algorithm **stops** when the distance between two subsequent estimation of the aggregated trajectory is below a predefined threshold. The distance used is the pointwise L_2 distance.

4 MIAA : a Modular and Iterative Aggregation Algorithm

Knowing the diversity of similarity and aggregation method for trajectories, we adapt the existing algorithm proposed in [Etienne and Devogele, 2014] by transforming it into a modular one. Modularity has substantial advantages such as flexibility (*i.e.*a measure can be easily replaced by a another), scalability (*i.e.* new measures can be easily added), and more relevant to the goal of this study, experiment testing (*i.e.* study the behavior and the influence of measures and parameters in different contexts and with different data).

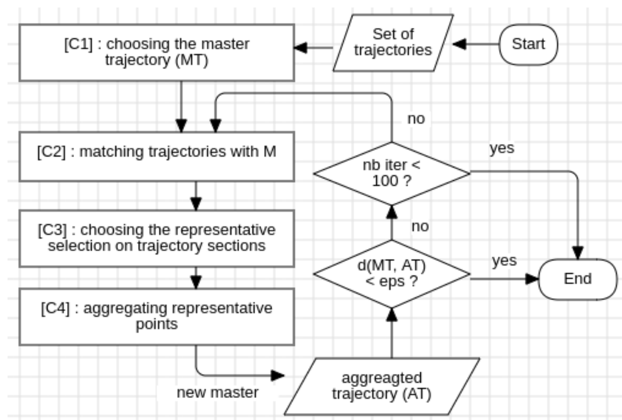


FIGURE 7 – The four components of modular and iterative aggregation algorithm for GNSS trajectories.

Figure 7 illustrates the proposed Modular and Iterative Aggregation Algorithm (MIAA) for GNSS trajectories which is composed by four components with new options. Thus, this proposal will create many variants of the algorithm. Note that, each component corresponds to a step of the algorithm.

The open-source implementation of the MIAA algorithm have been integrated into the Tracklib library ([Méneroux and van Damme, 2024]).

4.1 Component 1 - choosing the master trajectory

This component is representing the first step of the workflow. We propose three options (*Cf.* Figure 8) to get the master trajectory. The first, already proposed in [Etienne and Devogele, 2014], selects as master, the trajectory whose length is closest to the median of the lengths of all trajectories to be aggregated. We add, to this initial option, two new options : (1) the master trajectory is one that minimizes the sum of distances to other trajectories and (2) the master trajectory is randomly selected from the set of trajectories to be aggregated. These features have the advantages to study the influence of the choice of the master trajectory on the final result.

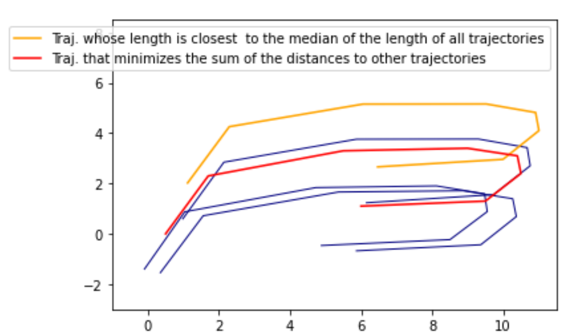


FIGURE 8 – Heuristics for choosing a trajectory *master*

4.2 Component 2 - matching trajectories with the master trajectory

This component is linked to the second step of the algorithms and contains different measures to compute the distance between two trajectories. Among the many similarity measures, we considered four measures : in addition to the matching based on the discrete Frechet distance, we add as new variants, two methods parameterized with L_p-norm ($p \in 1, 2$) and the nearest neighbour matching.

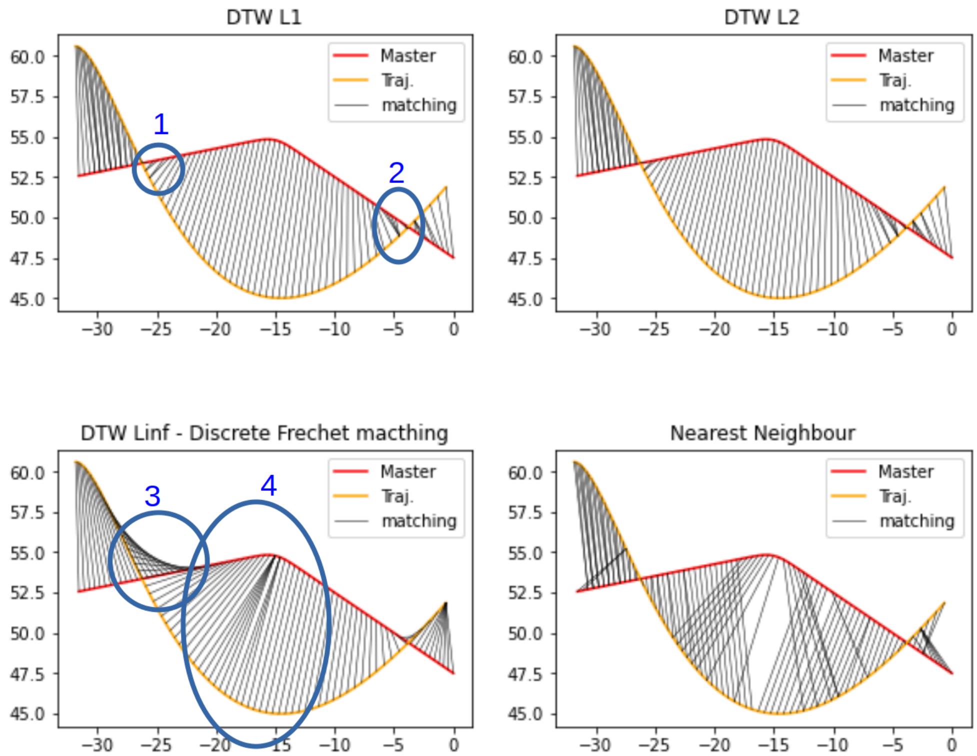


FIGURE 9 – Heuristics for choosing a similarity measure for matching trajectories

Clearly, the Nearest Neighbour distance differs from the others because it does not preserve the order of position in matching between the two trajectories, as described in the Figure 9 (bottom right image).

As mentioned in the section Formalization of the original algorithm , Discrete Frchet distance seeks to minimize the maximum of distances between the two curves. Once the maximum distance link is fixed, the other matching links are placed in the remaining spots. In Figure 9, the matching links are either grouped for distant points (Circled area 4) or create a large offset (Circled area 3).

DTW, both L_1 and L_2, aims at minimizing the sum of these distances,the matching links are therefore spaced more evenly, as shown in the the top 2 images in the Figure 9. The difference between the L_1 and L_2 distances in matching is minimal. The L_2 distance is the straight-line distance, while the L_1 distance follows the x-axis and y-axis. In Figure 9, the difference between these two distances becomes significant when the links between the points are close to the diagonals of the two axis (Circled area 1 and circle area 2).

4.3 Component 3 - choosing the representative position of each homologous points

As mentioned in the second step of the iteration of the aggregation algorithm, it is possible that multiple points on a trajectory are matched to the same point in the reference trajectory. Then it is required to reduce these points to a single representative position. This can be done with different methods : center of gravity, position with median time, position furthest from the master trajectory.

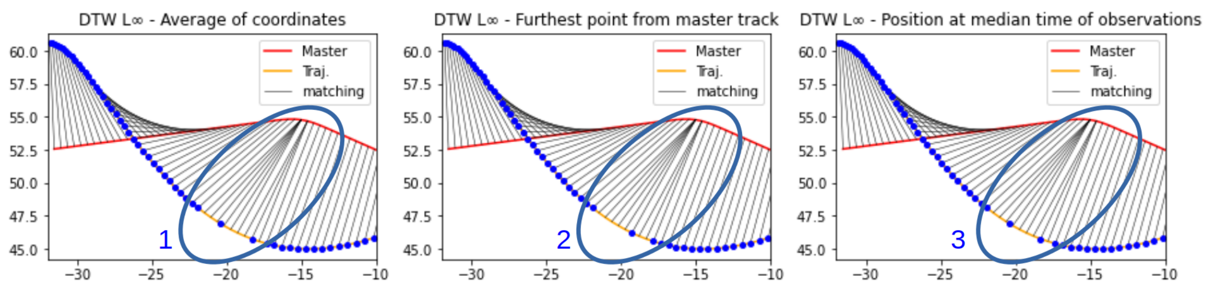


FIGURE 10 – Heuristics for options in [C3] for DTW L_∞

With the Discrete Frchet distance (L_∞) in C2, Figure 10 describe the three options. Unsurprisingly, the higher the link cardinality, the more this option will influence the position of the representative. The choices of the center of gravity and the position furthest from the master trajectory (Circled area 2) correspond to a spatial strategy, unlike the option of the position with median time. In Figure 10, since the points of the track are spaced at fixed time intervals, the difference between option 1 (Circled area 1) and option 3 (Circled area 3) is the same.

4.4 Component 4 - aggregating the representative position

So, for each reference position $(\mathbf{r}_j)_{j=1..m}$ on the master trajectory $\mathcal{R}$, we have a representative $\overline{xj}$ on each trajectory $\mathcal{R}$. Using an aggregation operator, a unique aggregating position can represent each reference position $\mathbf{r}_j$. We propose four aggregate function to calculate aggregating position : the marginal median ([Etienne and Devogele, 2014]), the geometric median, the L_2 mean and the L_∞ which is the center of the minimum covering circle. In this component, we can add a constraint with the four approaches : whether or not to anchor the aggregated position to an existing position in the dataset's trajectories as explained in step 3 of 3.

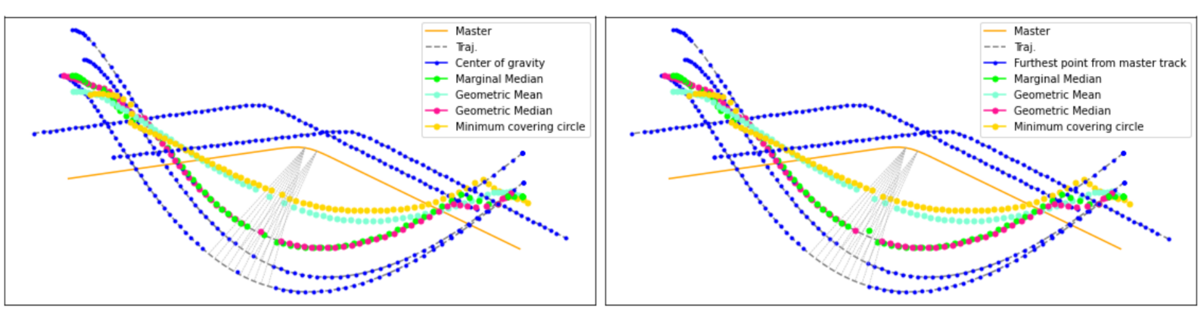


FIGURE 11 – Heuristics for options in [C4] for DTW L_∞

This step is the one that determines the final position of the points in the track. In Figure 11, we can observe that the marginal median suggests a point among the positions of the representatives of C3, whereas the geometric mean proposes a new position for the trajectory, which can therefore be influenced by outliers.

5 Extensive analysis and calibration

5.1 Termination of the algorithm

It should be mentioned that, to our knowledge, the termination of the algorithm has not yet been studied in [Etienne and Devogele, 2014]. MIAA algorithm is an iterative algorithm, and there is no guarantee that it will converge from an IT point of view. For example, at the end of an iteration, you can return to a previous step (the merged trajectory corresponds to a previous master trajectory), therefore iterations are entering a cycle. We demonstrate this potential drawback with a counter-example described below.

For example, consider these two trajectories shown in Figure 12-a :

$\mathcal{X} = <(68.0, 20.0), (69.0, 22.0), (69.0, 24.0), (67.0, 25.0)>$

$\mathcal{Y} = <(71.0, 14.0), (71.0, 16.0), (72.0, 19.0), (70.0, 22.0)>$

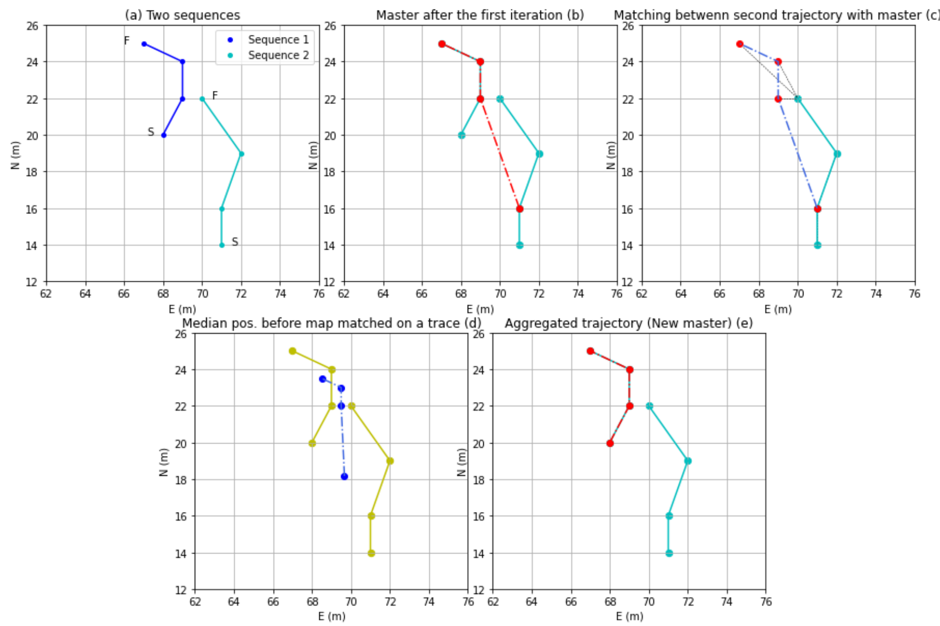


FIGURE 12 – Counter-example showing a case of non-convergence

The selected configuration is : for the master trajectory, is one that minimizes the sum of distances to other trajectories, the DTW-L1 has been chosen for matching trajectories, the center of gravity is taken for the representative position of each homologous and to aggregate the representative position, median aggregator is used to ultimately be map matched over a position of the trajectories.

In this configuration, the first iteration of the MIAA algorithm returns the aggregated trajectory represented by the red line in Figure 12-b. Now let us take a closer look at the second iteration. The matching of the trajectories with the master trajectory produces a single matching link (one vertex of $\mathcal{X}_2$ with one vertex of $\mathcal{R}$) except for the first reference position, we have a matching link 1-3 : (r_1, $x2_1$, $x2_2$, $x2_3$) (Figure 12-c) (corresponds to these three points of the second trajectory x_1 (71.0,14.0), $x2_2$ (71.0,16.0) and $x2_3$ (72.0,19.0)). Consequently the center of gravity of these three points is $\overline{x2_1}$ (214/3, 49/3) that is to say, the representative position associated to r_1 (Figure 12-d).

With anchor constraint, we need to match the center of gravity $\overline{x2_1}$ with one of the four vertex : r_1, $x2_1$, $x2_2$, $x2_3$, respecting this order (IT implementation). There are two candidates positions with the same distance r_1 and $x2_3$ (6.1388...8), so the winner is r_1 (first in the list).

At the end of the second iteration, the aggregated trajectory corresponds to the master trajectory of the first iteration. We are locked in a two-iteration cycle.

5.2 Algorithm calibration : further details

This section reproduces the online notebook on the Tracklib documentation website "Aggregated Trajectory : position errors and shape deviation estimation". It is inspired by the work presented in [Van Damme et al., 2024] and also provides some pieces of *Python* code from Tracklib library ([Méneroux and van Damme, 2024]) documentation [3].

We will examine the aggregation of trajectories using two matching distances : the Frchet distance and the DTW-L2 distance, and thus see its ability to reconstruct accurately the common path followed by all the individual sample trajectories by comparing the position errors and the shape deviation.

Our experiment will be conducted in four steps :

1. Step 1 : creating a synthetically reference track considered as the ground truth track,
2. Step 2 : creating a set of simulated tracks from the reference track,
3. Step 3 : computing the aggregation track from the set previously constructed (step 2),
4. Step 4 : the error between the estimated and ground truth track is then evaluated.

5.2.1 Step 1 : Create reference tracks

For this experimentation, we will examine the trajectory aggregation process for different types of paths to be reconstructed. In the context of mountain hiking, we identified three characteristic path shapes : nearly straight segments, moderately sinuosity road segments, and a zigzagging path composed of a series of switchbacks. To maintain shape consistency throughout the entire road segment, the generated trajectories have a length of approximately 300 meters.

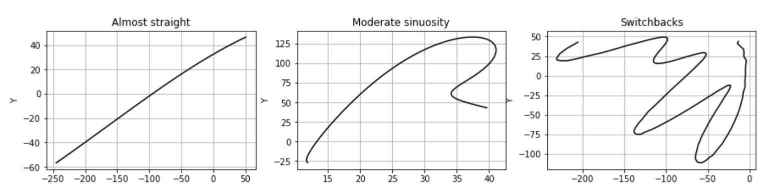
This three commonly mountain path shapes correspond to the reference tracks.

```
tkl.seed(123)
# ----------------------------------------------------------
# Generate the path 'Almost straight'
sentier1 = tkl.generate(0.5, dt=10)[::3]
sentier1.scale(5)

# ----------------------------------------------------------
# Generate the path 'Moderate sinuosity'
sentier2 = tkl.generate(0.1, dt=10)[::3]

# ----------------------------------------------------------
# Generate the path 'Switchbacks'
base_lacets = tkl.generate(0.4, dt=10)
sentier3 = tkl.noise(base_lacets, 20, tkl.SincKernel(20),
direction=tkl.MODE_DIRECTION_ORTHO)[::3]
sentier3.scale(4)

# ----------------------------------------------------------
SHAPES = ['Almost straight', 'Moderate sinuosity', 'Switchbacks']
sentiers = [sentier1, sentier2, sentier3]
```



3. https ://tracklib.readthedocs.io/en/latest/usecase/AggregatedTrajectory.html

5.2.2 Step 2 : Create sets of simulated GNSS trajectories

Now that we have our three reference trajectories, we want to create three sets of N simulated GNSS trajectories from them. To generate realistic noise, we used an approach described in [Ripley, 2009] and also employed in [Méneroux et al., 2023] :

with a random generator, we sampled N i.i.d. unit-variance and zero-mean gaussian values, compiled in a vector $\mathbf{x}$. It can easily be shown that, for any positive-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$, the random vector $\mathbf{y} = \mathbf{Ax}$ where $\mathbf{A}$ is a Cholesky factor of $\mathbf{\Sigma}$, is a realization of a correlated random vector $\mathbf{Y}$ having covariance matrix Σ. The covariance matrix $\mathbf{\Sigma}$ is formed with a (stationary) covariance kernel with three parameters :

— The **type of kernel** : exponential, gaussian, and triangular models are used.

— The **amplitude** of noise : is between 0 and 5 meters, as it is quite uncommon to find building databases with more than 5 m error amplitude. If necessary, the output tables could be extended to handle large errors.

— The correlation **scope** of the noise which roughly speaking describes how far apart two errors would remain correlated (in both amplitude and direction) : between 1 m (white noise) and 1000 m (global translation).

In tracklib, you have to create a Kernel and use the noise method on a track.

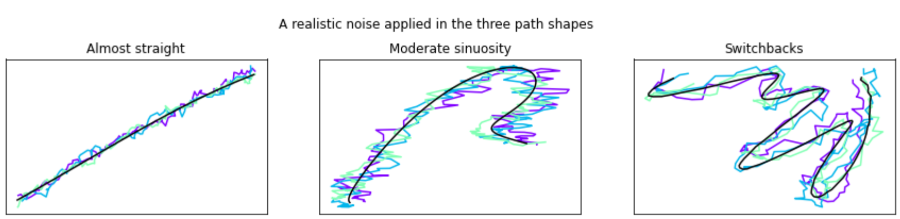
Trajectories have been generated with a 5 m-amplitude exponential covariance process, completed by a 1 m white noise process, a 50 cm range Gaussian Process for referencement error.

```
N = 20

# Generate 'Almost straight'
tracks1 = tkl.core.TrackCollection([sentier1]*N)
tracks1.noise(5, tkl.ExponentialKernel(1))
tracks1.noise(1, tkl.GaussianKernel(0.5))

# Generate 'Moderate sinuosity'
tracks2 = tkl.core.TrackCollection([sentier2]*N)
tracks2.noise(5, tkl.ExponentialKernel(1))
tracks2.noise(1, tkl.GaussianKernel(0.5))

# Generate 'Switchbacks'
tracks3 = tkl.core.TrackCollection([sentier3]*N)
tracks3.noise(5, tkl.ExponentialKernel(1))
tracks3.noise(1, tkl.GaussianKernel(0.5))
```



5.2.3 Step 3 : Compute aggregated trajectory

We have two evaluations to do : compute the aggregated trajectory with Frechet distance and with L_2 distance.

Each one is conducted as follows : we generate N' random noisy traces, with $N' \in [1, E]$ and $E \leq N$. For each generation, we are going to compare the agregated trajectory against the reference track (*i.e.* the ground truth).

This code compute an aggregation track for a set of 10 trajectories :

```
FRECHET = {'Almost straight': tkl.TrackCollection(),
  'Moderate sinuosity': tkl.TrackCollection(), 'Switchbacks': tkl.TrackCollection()}
DTW = {'Almost straight': tkl.TrackCollection(),
  'Moderate sinuosity': tkl.TrackCollection(), 'Switchbacks': tkl.TrackCollection()}

represent_method = tkl.MODE_REP_BARYCENTRE
aggmeth = tkl.MODE_AGG_MEDIAN
cstt = False
master = tkl.MODE_MASTER_MEDIAN_LEN
itermax = 25

# create set of 10 trajectories
TAB = set()
while len(TAB) <= 10:
    n = randint(0, N-1)
    TAB.add(n)

sets1 = tkl.TrackCollection()
for idx in TAB:
    sets1.addTrack(tracks1[idx])

# Frechet
p = float('inf')
mode = tkl.MODE_MATCHING_FRECHET
central1 = tkl.fusion(sets1, master=master, dim=2, mode=mode, p=p,
        represent_method=represent_method, agg_method=aggmeth, constraint=cstt)
FRECHET['Almost straight'].addTrack(central1)
...

# DTW
p = 2
mode = tkl.MODE_MATCHING_DTW
central1 = tkl.fusion(sets1, master=master, dim=2, mode=mode, p=p,
      represent_method=represent_method,  agg_method=aggmeth, constraint=cstt)
DTW['Almost straight'].addTrack(central1)
...
```

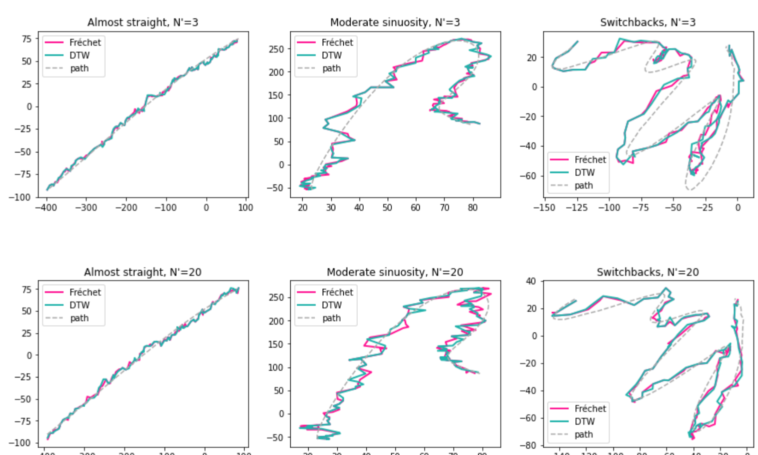


FIGURE 13 – Aggregated trajectories for two samples of 3 and 20 trajectories and computed with Discrete Frchet distance (in pink) and DTW-L_2 distance (in blue)

5.2.4 Step 4 : Evaluate error between the aggregated and ground truth track

Position error measurement

We compute the distance $pointwiseL_2$ (*Root Mean Square Error*) by finely resampling both the aggregated and ground truth trajectories from 120 to 1,000 points, ensuring one point every 50 cm over 500 meters.

```
def rmse(central, sentier):

    central.resample(npts=1000, mode=1)
    sentier.resample(npts=1000, mode=1)

    # compute the distance NearestNeighbour
    m = min(sentier.size(), central.size())
    return tkl.compare(central[0:m], sentier[0:m], tkl.MODE_COMPARISON_POINTWISE, p=2)

rmse(FRECHET['Almost straight'][s], sentier1)
...
rmse(DTW['Switchbacks'][s], sentier3)
```

Which results in the following graphically :

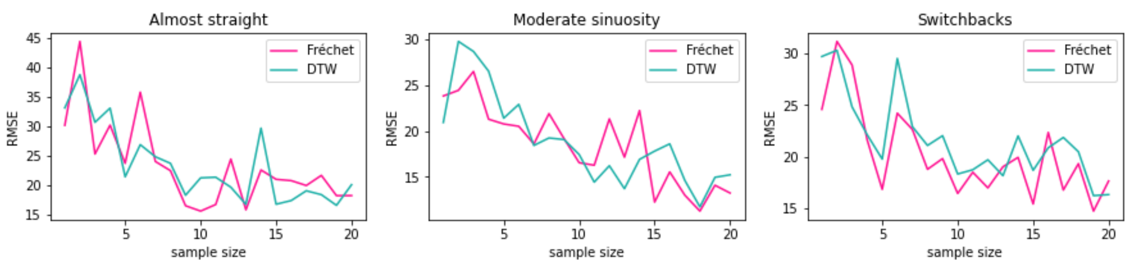


FIGURE 14 – Position error measurement for each samples of N trajectories generated

From the perspective of positional accuracy, neither matching with the Frchet distance nor matching with the DTW distance prevails : the two curves have the same shape, and the differences, regardless of the number of trajectories in the sample, are minimal.

Shape deviation measurement

Let's start by aligning, with a geometric affine transformation, the aggregated track with the reference track to abstract away from position issues. Then, we finely resample both the aggregated and ground truth trajectories from 120 to 1,000 points, ensuring one point every 50 cm over 500 meters. Finally, we compute the distance *NearestNeighbour* for all positions to estimate shape deviation.

```
def shapeDeviationMeasure(central, sentier):

    # Align the aggregated track with the reference track
    tkl.mapping.mapOn(central, sentier, verbose=False)

    # resample to 1000 points
    central.resample(npts=1000, mode=1)
    sentier.resample(npts=1000, mode=1)

    # compute the distance NearestNeighbour
    return tkl.compare(central, sentier, tkl.MODE_COMPARISON_NN, p=2)

shapeDeviationMeasure(FRECHET['Almost straight'][s], sentier1)
...
shapeDeviationMeasure(DTW['Switchbacks'][s], sentier3)
```

Which results in the following graphically :

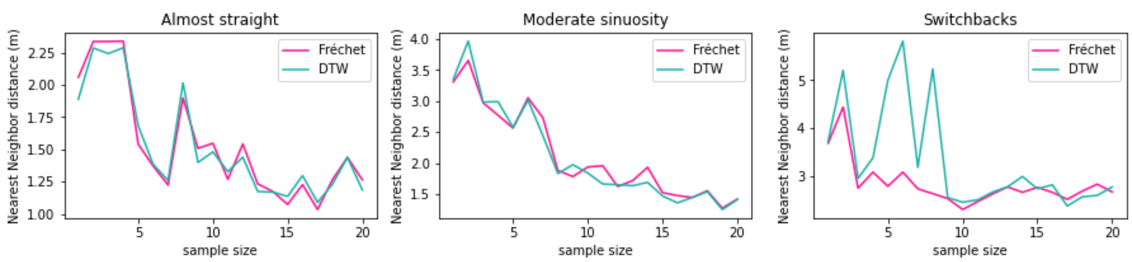


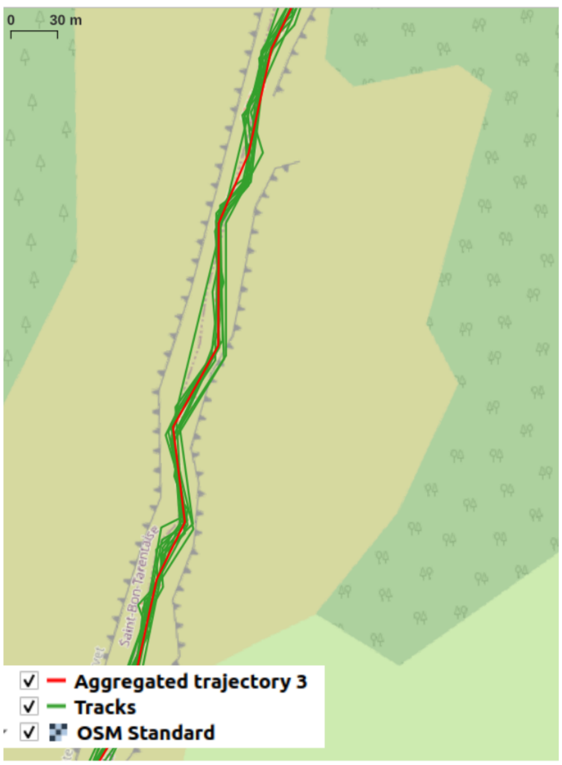
FIGURE 15 – Shape deviation measurement for each samples of N trajectories generated

The Figure 15 still shows a close similarity between the two curves : shape deviation measurement are quite close together and their profiles are quite similar.

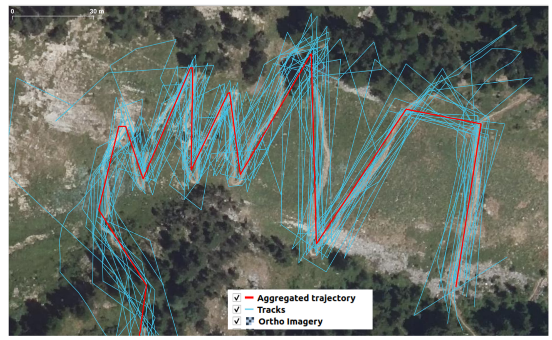
5.3 Crowd-sourced trajectories dataset

Here we present the results obtained on real data. As already mentioned, the analyse is not strictly "metrological" but rather qualitative, a graphic investigation. We define a first context : a ridge which characterises terrain constraints (Figure 16a), a switchback trail which characterises an infrastructure support (Figure 16b) and a straight line followed by a series of twists and turns with varying distances between them which characterises variations of the shape of the path (Figure 16c).

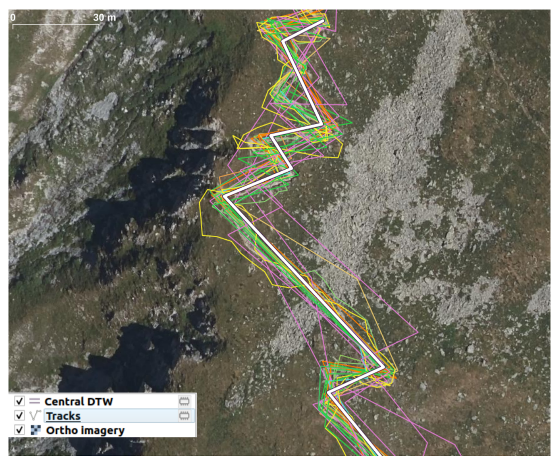
Note that, traces may have been previously filtered via a simplification algorithm like Douglas-Peucker. Indeed, all geometries offer the same characteristics like vertex of bends, turns in route. If this was not the case, traces would be more dissimilar in shape.



(a) Aggregated trajectory closely follows the ridge path



(b) Aggregated trajectory in red follow the most popular route



(c) median operation effect in a series of bends with a Crowdsourced dataset

Figure 16 – Aggregated trajectory result on Crowd-sourced dataset

Aggregated trajectory produced by the MIAA algorithm (red line in Figure 16b and white line in Figure 16c) detects the road shape very well, all the bends have been detected. In the context of mountain hiking, we can see the aggregated trajectory deviates from the road, which may make it possible to quantify the off-tracking. In Figure 16a, we can observe that some turns are not detected, as the geometry of the aggregated trajectory tends to generalize and lose certain inflection points.

5.4 Perspective : performance under extreme conditions

Concerning high noise amplitude, we deliberately chose a low one (5 meters) in our experiments to adapt to advancements in GNSS technological (new constellations, new signals, improvements in receiver electronics). However, we have not studied the case of extreme noise conditions, and we could relaunch computations with extreme values, on the same synthetic GNSS trajectories datasets, with an amplitude of the exponential covariance process from 5 meters (*Cf.* [Lee et al., 2023]) to 30 meters (as mentioned in the literature for errors in heavy coverage canopy areas in [Piedallu and Gégout, 2005]).

Additionally, for outlier traces, particularly off-roads, we conducted empirical tests on the multi-sensors/multi-canopy traces dataset. With few outliers, our tests confirmed there was no impact on the aggregated trajectory, demonstrating its robustness as mentioned in [Etienne and Devogele, 2014]. The algorithm computes, for each matched points in the master trajectory, the robust median positions between all trajectories. Thus, we decided not to further investigate robustness, in order to focus on its modularity. We could study the performance of the algorithm with many off-roads in the sample traces to be merged. For example, the robustness analysis would consist in including a proportion p of off-roads in a given number of trajectories, and determine how the quality of the aggregated trajectory decreases as p increases. In our opinion, with the effect of the median, results should not be dramatically modified, at least up to p=25%. If needed, we can run the test and add comments with a figure depicting the aggregated trajectory quality as a function of p.

Références

[Biljecki et al., 2015] Biljecki, F., Heuvelink, G. B., Ledoux, H., and Stoter, J. (2015). Propagation of positional error in 3d gis : estimation of the solar irradiation of building roofs. *International Journal of Geographical Information Science*, 29(12) :2269–2294.

[Blunck et al., 2011] Blunck, H., Kjærgaard, M. B., and Toftegaard, T. S. (2011). Sensing and classifying impairments of gps reception on mobile devices. In Lyons, K., Hightower, J., and Huang, E. M., editors, *Pervasive Computing*, pages 350–367, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bonin, 2002] Bonin, O. (2002). *Modèle d'erreurs dans une base de données géographiques et grandes déviations pour des sommes pondérées : application à l'estimation d'erreurs sur un temps de parcours*. PhD thesis, Paris 6.

[Calloch et al., 2024] Calloch, P., Labb, B., and Peseux, L. (2024). Établissement d'une méthode low-cost de vérité terrain pour l'étude de la précision de traces gnss de randonneurs en forêt. Technical report, École nationale des sciences géographiques (ENSG-Géomatique).

[Etienne and Devogele, 2014] Etienne, L. and Devogele, T. (2014). Trajectoires médianes. In *14 ème conférence Extraction et Gestion des Connaissances, Ateliers fouille de données spatiales et temporelles & construction, enrichissement et exploitation de ressources géographiques pour l'analyse de données*, Rennes, France.

[Grejner-Brzezinska et al., 2005] Grejner-Brzezinska, D., Toth, C., and Yi, Y. (2005). On improving navigation accuracy of gps/ins systems. *Photogrammetric engineering & remote sensing*, 71(4) :377–389.

[Lee et al., 2023] Lee, T., Bettinger, P., Merry, K., and Cieszewski, C. (2023). The effects of nearby trees on the positional accuracy of gnss receivers in a forest environment. *PLOS ONE*, 18(3) :1–20.

[Méneroux et al., 2023] Méneroux, Y., Maidaneh Abdi, I., Le Guilcher, A., and Olteanu-Raimond, A.-M. (2023). Is the radial distance really a distance ? an analysis of its properties and interest for the matching of polygon features. *International Journal of Geographical Information Science*, 37(2) :438–475.

[Méneroux and van Damme, 2024] Méneroux, Y. and van Damme, M.-D. (2024). Tracklib : a python library with a variety of tools, operators and functions to manipulate GPS trajectories.

[Piedallu and Gégout, 2005] Piedallu, C. and Gégout, J.-C. (2005). Effects of forest environment and survey protocol on gps accuracy. *Photogrammetric Engineering & Remote Sensing*, 71(9) :1071–1078.

[Ripley, 2009] Ripley, B. D. (2009). *Stochastic simulation*. John Wiley & Sons.

[Roberts, 1993] Roberts, W. D. S. (1993). *GPS time correlation and its implication for precise navigation*. PhD thesis, Newcastle University.

[Van Damme et al., 2024] Van Damme, M.-D., Méneroux, Y., and Olteanu-Raimond, A.-M. (2024). A metrological analysis of a modular and iterative aggregation algorithm of gnss trajectories. In *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '24, page 633636, New York, NY, USA. Association for Computing Machinery.

[Vauglin, 1997] Vauglin, F. (1997). *Modèles statistiques des imprécisions géométriques des objets géographiques linéaires*. PhD thesis, Université de Marne-la-Vallée (1991-2019).

[Zhang and Yang, 2015] Zhang, Y. and Yang, Y. (2015). Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187(1) :95–112.