Map construction algorithms: a local evaluation through hiking data*

David Duran† Vera Sacristán‡ Rodrigo I. Silveira‡

Abstract

We study five existing map construction algorithms, designed and tested with urban vehicle data in mind, and apply them to hiking trajectories with different terrain characteristics. Our main goal is to better understand the existing strategies and their limitations, in order to shed new light into the current challenges for map construction algorithms.

We carefully analyze the results obtained by each algorithm focusing on the local details of the generated maps. Our analysis includes the characterization of 10 types of common artifacts, which occur in the results of more than one algorithm, and 7 algorithmic-specific artifacts, which are consequences of different algorithmic strategies. This allows us to extract systematic conclusions about the main challenges to fully automatize the construction of maps from trajectory data, to detect the strengths and weaknesses of the potential different strategies, and to suggest possible ways to design higher-quality map construction methods.

We consider that this analysis will be of help for designing new and better methods that perform well in wider and more realistic contexts, not only for road map or hiking reconstruction, but also for other types of trajectory data.

1 Introduction

The massive amount of GPS traces being generated every day, due to the ubiquity of GPS receivers, are giving rise to many geographic data analysis challenges. One of them is *map construction*: the automatic generation of maps from GPS traces. In the last few years, volunteered geographic information platforms, such as OpenStreetMap or Wikimapia,[1] have shown that the potential of user-generated geographic content is enormous, but also require automated tools to handle the vast amounts of data involved, especially in order to produce high-quality results. For this reason, it is not surprising that map construction algorithms have received a lot of attention during the last few years. A recent book on the subject [4] lists over 20 papers on map construction, almost all of them published during the last decade.

In the general formulation of the map construction problem, the input is a set of GPS traces, each consisting of a finite sequence of time-stamped positions (*trajectories*). The output is a representation of the underlying map, usually as an embedded graph. Up to now, most previous work on map construction has focused on urban vehicle data. That is, the trajectories used were generated by motor vehicles on urban street networks, and the goal was to generate a street map. Furthermore, in many cases the data used were generated by fleets of identical vehicles equipped with the same GPS devices. However, it is clear that the map construction problem is interesting for a broader kind of data, since people equipped with all sorts of GPS devices move around the world in many other ways, such as by foot, bicycle, or boat, just to name a few.

The goal of this paper is to better understand the existing map construction algorithms at the local level, their limitations—and what causes them—, in order to come up with new map reconstruction strategies that apply in a wider context. Existing evaluations of map construction methods always

*A preliminary version of this work appeared in [14]

†Universitat Politècnica de Catalunya, Spain. david.duran.rosich@est.fib.upc.edu

‡Departament de Matemàtiques, Universitat Politècnica de Catalunya, and BGSMath, Spain. {vera.sacristan,rodrigo.silveira}@upc.edu. Partially supported by projects MTM2015-63791-R (MINECO/FEDER), Gen. Cat. 2017SGR1640. R.I. S. was also supported by MINECO through the Ramón y Cajal program.

[1] See `http://openstreetmap.org` and `http://wikimapia.org`, respectively.

focus on global quality measures. However, the details of generated maps are also important, and should be correctly represented in any high-quality map. For that reason, our focus is on understanding how existing map construction methods perform at a local level.

It is for this reason that in this work we consider hiking data. Hiking data have also been generated in vast amounts during recent years, and we believe that they have several particular features that make them interesting to understand the current challenges in map construction, regardless of whether the trajectories come from cars, people, or boats. One reason comes from the fact that hiking often takes place in natural environments with varied terrain characteristics (possibly with dense vegetation and rough topography). In some cases, terrain and vegetation give rise to higher GPS error. On hilly terrains, paths can be extremely narrow, have sharp turns, be quite winding, and can cross at all sorts of angles, while on flat agricultural terrains paths behave somehow similarly to urban streets, roads and even highways. Another important characteristic is that trajectories are generated by pedestrians. Thus, while street networks can be considered "hard-constraints", hiking networks are of a softer nature—a car cannot easily leave the street network and drive off-track, but that is much easier and frequent with hikers. In addition, it is not so infrequent to have parts of hiking trails that go through areas in which no proper path exists, such as open or boulder fields. Finally, hiking trajectories have particular characteristics in terms of speed, bidirectionality, and are extremely heterogeneous. Many of these factors affect the hiking trajectories, producing more noise due to higher GPS error, and posing special challenges to map construction algorithms. It should be stressed that many of these challenges also appear when dealing with urban vehicle data, but it may be harder to spot them there. In short, the data we have used extends the one previously considered to evaluate map construction algorithms in several aspect, as they are richer and allow for a better study of the map construction strategies and their limitations.

Related work In the last years there has been dozens of different map construction methods proposed, which take as input a set of trajectories and output a map, often represented as a graph embedded in the Euclidean plane. Ahmed et al. [3, 4] have proposed a classification of map construction methods that distinguishes four different major strategies that existing algorithms follow. The most basic approach is that based on *point clustering* (e.g., [15, 18, 31]), which consists of methods that interpret trajectories as sets of points, and produce maps by clustering those points to obtain streets and their corresponding intersections. In *density-based* algorithms (e.g., [1, 8, 11, 12, 20, 25, 26]) the main idea is to compute a density function over the set of input points (often obtained by sampling the input trajectories). From the density function one can then extract roads (by identifying ridges) and their structure. For instance, a recent algorithm by Wang et al. [28] applies discrete Morse theory to produce a map in this way. Still, density-based methods are based on points, and thus do not treat the input trajectories as continuous objects. *Intersection linking* algorithms (e.g., [16, 19]) follow a two-step approach, by first finding intersection nodes, and then connecting the nodes using the information given by the input trajectories. For instance, Karagiorgou and Pfoser's algorithm [19] identifies intersection nodes by detecting changes in movement and clustering the resulting locations, and then uses portions of trajectories between the identified nodes to form the edges of the map. Following a very different approach we have *incremental track insertion* algorithms (e.g., [5, 10, 22, 24]), which, starting from an empty map, insert the trajectories one by one. How to select the next trajectory to add, and how to decide if it is already represented in the map are important challenges in such methods, for which often map-matching ideas are used [5, 24].

We note that there are plenty of map construction algorithms that do not fit into these four general categories, and many that combine ideas from more than one. For instance, the method proposed in CrowdAtlas [30] combines map matching techniques, clustering and intersection linking ideas. CellNet [21] is essentially an intersection linking algorithm, but it uses a rather elaborate density-based method to detect the splits that will form the intersections in the map. A recent method proposed by authors of this paper is based on subtrajectory clustering [9], following an approach inverse to that of intersection linking methods: first edges are detected by bundling trajectories together, and then the intersections between them are computed. Methods based on topology tools such as Morse theory, have also been proposed [29, 13]. For a more complete treatment of map constructions methods (albeit up to 2015), we refer to the book by Ahmed et al. [4].

Despite the large number of map construction methods proposed, few comparisons between the different methods have been carried out. There are several reasons that explain this: the code of

most algorithms is not publicly available, there is no standard benchmark data to use (each new algorithm is usually evaluated on its own data), and there is no consensus on what aspects of a map should be evaluated to determine its quality. An important exception in this regard is the work by Ahmed et al. [3], which presented the first cross-comparison of seven map construction algorithms [5, 8, 10, 12, 15, 17, 19], evaluated for four different urban data sets. The accuracy of each generated map was compared to that of a ground-truth map, using four different distance measures: Directed Hausdorff distance [6], path-based distance [2], a distance measure based on shortest paths [19], and a graph-sampling-based map comparison measure [8]. It is important to note that all the distance measures considered by Ahmed et al. [3] are global measure, since they assign a single score to each generated map. All the data used for the study, including re-implementations of several of the algorithms, was made publicly available [23]. From their experiments, Ahmed et al. [3] observed certain general behavior of the different algorithms that is worth summarizing here. Density-based methods like [8, 12] produce maps with fewer vertices and edges, missing streets that don't have enough trajectories through them. Incremental algorithms [5, 10] have issues clustering tracks and often have multiple paths that represent the same road. In terms of accuracy, measured by the path-based and Directed Hausdorff distances, the methods obtaining the best results were [8, 12, 19]. At the same time, it is observed that higher accuracy is often associated with worse coverage: methods like that of Davies et al. [12] include more accurate paths, but at the same time miss many others. The authors observe that the method by Karagiorgou and Pfoser [19] often achieves a good balance of both aspects. An insightful conclusion of Ahmed et al. [3] is that no algorithm seems to be good at both accuracy and coverage.

In this paper, we build on top of [3], in order to evaluate different map construction algorithms but from a local point of view, through the use of hiking trajectory data. The artifacts that we identify reflect issues of the different methods in both accuracy and coverage, but at the local level.

Goals of this work

- Analyze locally the result of map construction methods. Existing analysis and comparisons are based on global quality measures. However, global measurements do not explain how the resulting maps look at the local level, in detail. We aim at understanding how well the produced maps represent the ground truth at the local level, something that has a major impact in the perceived quality of the resulting map.
- Understand causes of poor maps. Global measures help indicate that a resulting map is poor, but not the reasons for that. As part of our local analysis, we want to understand not only where algorithms go wrong, but why: whether this is caused by erroneous assumptions on the data, algorithmic decisions or both.
- Identify challenges for high-quality map construction. In order to obtain methods that produce better maps at all levels, we need to pinpoint the main challenges that make current methods perform poorly. This will be useful to help devise new strategies that can overcome these difficulties.

Contributions We designed an experimental study of five existing map construction algorithms, which are a subset of the seven algorithms evaluated by Ahmed et al. [3].[2] We chose four different terrain areas of varied geographic characteristics, ranging from mostly flat rural areas (similar to many urban landscapes) to mountainous areas with dense forest. For each terrain we considered a set of user-generated hiking GPS trajectories downloaded from trajectory-sharing website Wikiloc,[3] each with a total number of nodes between 38,000 and 288,000. An important consequence of the nature of our data set is that it is extremely heterogeneous in aspects like length, speed, sampling rate or GPS error.

Before running the existing algorithms on our hiking data, we had to understand and set the (often many) parameters used by each algorithm. This task, which required more effort than expected, implied analyzing each algorithm in detail, to understand the meaning and relevance of each parameter,

[2] Unfortunately, we were not able to include two of the seven algorithms analyzed in [3]. This was due to not having the code available in one case [17], and to technical issues when attempting to run the algorithm in the other case [8].

[3] `http://www.wikiloc.com`

and to adjust it to obtain the best possible results for each data set. A side result of this work is a systematic and detailed analysis of all the parameters that influence each algorithm, their meaning, and how they interact and behave, and the issues arising in their adjustment to specific data sets. Given that many other algorithms use similar parameters, our analysis can be useful for other methods as well.

We present an analysis of the generated maps produced by each algorithm on each data set, including a thorough discussion on the most important artifacts detected, in order to pinpoint the major issues found by the different methods. In contrast to the comparisons performed by Ahmed et al. [3], which were quantitative and global, our study is both qualitative and quantitative, and local, focusing on the details of the generated maps. Our analysis includes the characterization of 10 types of common artifacts, which have been observed in at least two of the algorithms, and 7 algorithmic-specific artifacts. Based on that, we identify different challenges that affect several of the algorithms, often caused by (combinations of) algorithm design decisions, characteristics of the terrain or the trajectory data. Many of the artifacts identified are not exclusive to hiking data, and can also appear in other setting such as urban vehicle data.

We consider that our analysis sheds new light on the current challenges for map construction algorithms, and will be of help for designing new and better methods that perform well in wider and more realistic contexts. It is important to stress that our goal is not to determine which algorithm performs best, but to identify the weak points of each algorithm, their causes, and possible solutions. For this reason we finish with a summary of what we consider the main lessons learned from this analysis, and a series of suggestions that we consider that new algorithms for map construction should take into account.

We note that in a shorter and preliminary version of this work [14], we described the experiments that gave rise to this study. While the preliminary version mostly focused on the description of the experimental setup and a set of 26 artifacts identified, but without any analysis of their cause or nature. In contrast, in this work we have classified and grouped the artifacts into 10+7 artifacts, and carried out a refined analysis of them. For each of the ten common artifacts we propose an objective way to measure the extent to which the artifacts is present for the different generated maps, together with a deeper analysis of their causes, related challenges, and possible ways to tackle them in future map construction methods.

2 Terrains, paths, and data

In order to run our experiments, we have used four sets of trajectory data from four different geographic areas. Each trajectory consists of a GPS track generated by a pedestrian carrying a GPS receiver when hiking in the corresponding area.

2.1 Terrains and paths

Four different geographic areas in Catalonia have been chosen, which are popular for hiking: *Delta del Llobregat* (Delta), *Aiguamolls del Baix Empordà* (Aiguamolls), *Garraf* (Garraf), and *Turó de l'Home* (Montseny). They are illustrated in Figure 1.

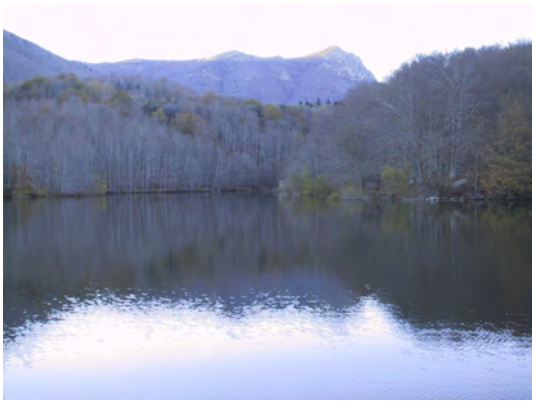
The four areas cover a diversity of characteristics that can be found in hiking terrains. *Delta* and *Aiguamolls* are rather flat, while the *Garraf* and *Montseny* areas are part of mountain ranges that have highest points at 593 m and 1712 m, respectively. As for the vegetation, *Garraf* vegetation consists mostly of dense shrubland, typically of about one meter in height, while a large part of the area in *Montseny* is covered by dense woods. *Delta* and *Aiguamolls* are mostly agricultural lands and have little vegetation close to the walking paths.

Comparison with urban streets, roads and highways. In flat zones, paths have some topographic similarities with urban streets, roads and highways: paths are wide and straight, and intersect at right angles, especially in agricultural areas. Notice, though, that *free walking* areas such as open fields or beaches are very different in this sense: paths disappear and pedestrians can walk anywhere in these portions of the terrains. On the other hand, the relief of mountainous areas makes them very different from (flat) urban areas: paths and trails are narrow and winding, and they can be connected to each other forming very small angles. In some cases, hilly terrains can also have portions

(a) Cropland in *Delta*.

(b) *Aiguamolls*: cropland, river and beach.

(c) Characteristic small bushes in *Garraf*.



(d) *Montseny*, view of the Santa Fe reservoir.

Figure 1: Views of the terrains chosen. Source: Wikimedia Commons.

where pedestrians do not follow paths, either because there is no path to follow, as in open or boulder fields, or paths are obstructed by obstacles such as puddles, fallen trees, or rocks, forcing hikers to go off-track. In addition, when the ground is flat, paths are only distinguishable through their latitude and longitude (like streets, roads and highways) while in a hilly context two paths can be very similar in latitude and longitude and substantially different in elevation. Detailed information on the main characteristics of the different contexts is given in Table 1.

Location	**Area (km^2)**	**Min. path sep. (m)**	**Path width (m)**	
			Min	Max
URBAN				
Athens large	12×14	35	5	25
Athens small	2.6×6	60	5	16
Berlin	6×6	55	6	22
Chicago	7×4.5	54	6	20
FLAT				
Delta	5×2.8	8	2	5
Aiguamolls	9.6×5.9	13	2	6
HILLY				
Garraf	6.7×4.5	15	1	5
Montseny	8.2×4.7	10	1	5

Table 1: Area covered and main characteristics of each data set. Area is expressed in terms of east-west and north-south distances. Minimum path separation and path width have been empirically estimated using Google Earth satellite imagery.

In the following we describe in detail the precise areas studied in this work, see also Table 2.

		Latitude	Longitude	North	East
FLAT					
Delta	min (SW)	41.286377	2.082439	4570954 m	423166 m
	max (NE)	41.311783	2.141361	4573724 m	428128 m
Aiguamolls	min (SW)	41.988487	3.085601	4648501 m	507090 m
	max (NE)	42.041505	3.202024	4654404 m	516720 m
HILLY					
Montseny	min (SW)	41.752525	2.418508	4622463 m	451656 m
	max (NE)	41.795680	2.516904	4627203 m	459863 m
Garraf	min (SW)	41.243725	1.861402	4566438 m	404594 m
	max (NE)	41.285209	1.940710	4570959 m	411296 m

Table 2: Rectangular windows for the terrains considered, both in the WGS 84 cordinate system and UTM projection (zone 31T).

Delta The region covered is defined by a rectangular window of 5 km by 2.8 km. The terrain is at sea level and does not have any relevant slope. In this area we can find the mouth of the Llobregat river. As a consequence, there are several irrigation canals and crops, which give rise to straight paths that intersect perpendicularly most of the times. It is worth mentioning that in this area there are also parallel paths at very short distance from each other (15–20 m).

Aiguamolls. The area studied is defined by a rectangular window of 9.6 km by 5.9 km. This region is quite similar to *Delta* due to the presence of the mouth of river Ter. Again, fields are abundant, paths tend to be straight and to intersect perpendicularly. The terrain is at sea level and does not have important elevation changes. Nevertheless, this area does not have parallel paths at a short distance. Finally, it is important to notice that this area includes part of a sand beach: a strip of about 50–70 m of width along the sea.

Garraf. The area considered is defined by a rectangular window of 6.7 km by 4.5 km. *Garraf* is a massif located in the Catalan coastal mountain range. Its heights reach almost 600 m, and hills have calcareous soil, with caves and shafts. Vegetation is scarce, mostly formed by herbs and small shrubs. Like in *Montseny*, winding paths intersect in acute angles. The area hosts a few coastal urban developments. Most of the trails start at one of them, which has a train station.

Montseny. The area studied is defined by a rectangular window of 8.2 km by 4.7 km located in the natural park *Montseny*. The topography of this zone is typical of mountain hiking areas in which no climbing is required to reach the summit. As such, footpaths can be quite narrow, trails often have acute turns and acute intersection angles to help gain altitude. Except at the highest parts of the massif, hillsides are covered by dense vegetation, mainly beeches and fir trees, among other typical trees of Central European forests. The area considered includes the highest peak (1712 m) and its surroundings.

2.2 Data

In order to run our experiments, we have obtained four different sets of hiking trajectories, one for each of the selected areas. Trajectories have been obtained from the website Wikiloc[4], which currently offers over 10 million trajectories from outdoor activities, shared by over 4 million users worldwide. Trajectories are uploaded by users, who individually obtain them while hiking, biking, and the like. Consequently, Wikiloc data is very heterogeneous. In order to reduce their heterogeneity, we have only used trajectories classified by the users as "hiking" or "walking". Figure 2 shows the trajectories used for each of the four hiking areas. Each trajectory consists of one GPX file [27]: an XML file

[4] `http://www.wikiloc.com`

(a) *Delta* (b) *Aiguamolls*

(c) *Garraf* (d) *Montseny*

Figure 2: General view of the four trajectory sets.

containing a sequence of tracked points. Each point entry consists of latitude and longitude (WGS84), elevation (in meters), and a timestamp.

Comparison with previously used trajectories. For the purpose of map construction, certain aspects of the input trajectories that are particularly relevant are the data set complexity, sampling rates, and GPS error. In the following we discuss the main differences on these aspects for the different trajectory sets, summarized in Table 3.

Area	**Traj.** #	**Nodes** #	**Length** [km]	**Speed** [km/h]	**Sampling rate** [sec.]	**GPS Error** [m]
URBAN						
Athens large	120	72 439	11 214	20.27 (66.74%)	30.00 (0.00%)	12
Athens small	129	2 840	389	18.92 (59.43%)	30.00 (0.00%)	12
Berlin	27 189	192 223	36 036	24.13 (37.40%)	40.41 (38.70%)	15
Chicago	889	118 360	2 563	33.46 (29.68%)	2.88 (42.45%)	16
FLAT						
Delta	161	38 029	364	4.71 (40.09%)	10.15 (55.25%)	8
Aiguamolls	101	46 116	363	4.89 (26.32%)	7.29 (79.06%)	5
HILLY						
Garraf	630	288 472	2 348	4.32 (39.16%)	8.63 (63.97%)	15
Montseny	101	128 181	776	3.24 (48.65%)	8.56 (77.33%)	22

Table 3: Trajectory data used. Values between parentheses indicate relative standard deviation. GPS errors were estimated empirically using Google Earth satellite imagery.

To put our data sets into perspective, in this section we also compare our hiking trajectory data

to the urban vehicle trajectories used in [3], consisting of four data sets: *Athens* (data from school buses, in two sizes: *small* and *large*), *Berlin* (data from taxi rides), and *Chicago* (data from university shuttle buses).

In terms of complexity, our hiking data is comparable to that of previous work on urban vehicle data: the total numbers of nodes in each set have very similar ranges, and the number of trajectories is also comparable. The only exception in this respect is the *Berlin* data set, where the number of trajectories is substantially larger (although trajectories have substantially fewer nodes). As expected, the overall distances traveled are much larger in all vehicle data sets, as also is mean speed, when compared to hiking data.

An important aspect for many algorithms is the angular differences at each trajectory point. We expected hiking data to contain a larger number of sharp turns than urban data sets, but, contrary to our intuition, that was not the case. As seen in Figure 3, the third quantile on our hiking data is lower than *Athens* and *Berlin* urban data. The data in *Chicago* had been already heavily preprocessed, hence showing an anomalous low angular difference. However, Figure 4 shows that in all data sets there is a negative correlation between the angular difference and the distance between consecutive points. Figure 4 also shows that the distance between consecutive points moves within 0 m and 25 m

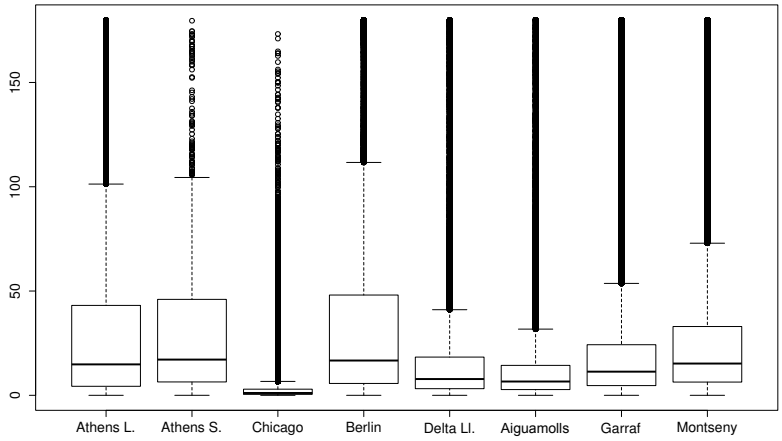


Figure 3: Angular differences in trajectory points. 0° means completely straight whereas 180° is a complete turn-back.

on the hiking data sets, whereas in the urban data sets they reach 460 m in *Athens*. This is due to the difference between speeds, and it will have its consequences on the algorithms results as trajectory points are more densely distributed. Finally, from the figure it is evident that the data from *Chicago* and *Berlin* have been preprocessed to remove small inter-point distances.

Regarding sampling rates, the *Athens* and *Berlin* data sets have an average of one sample every 30 and 40.41 seconds, respectively, while the remaining trajectory sets have higher sampling rates, with averages between 2.88 and 10.15 seconds. It is also worth mentioning that standard deviation is equal to zero in *Athens*, is about 40% for *Berlin* and *Chicago*, and increases to 55–79% in the hiking data we have worked with. Map reconstruction must objectively be more difficult within this context.

Finally, and partially related to the previous observation about standard deviation in sampling rates, it is important to notice that GPS errors in *Garraf* and *Montseny* trajectories tend to be larger than in the urban data sets, since GPS signal can be blocked or distorted by the mountains, as well as by the forests in the case of *Montseny* [32].

3 Methodology

We selected five existing algorithms (we will refer to them by the initials of their authors: AW, CK, DBH, ES, KP), described in detail in Section 4, and used the implementations publicly available

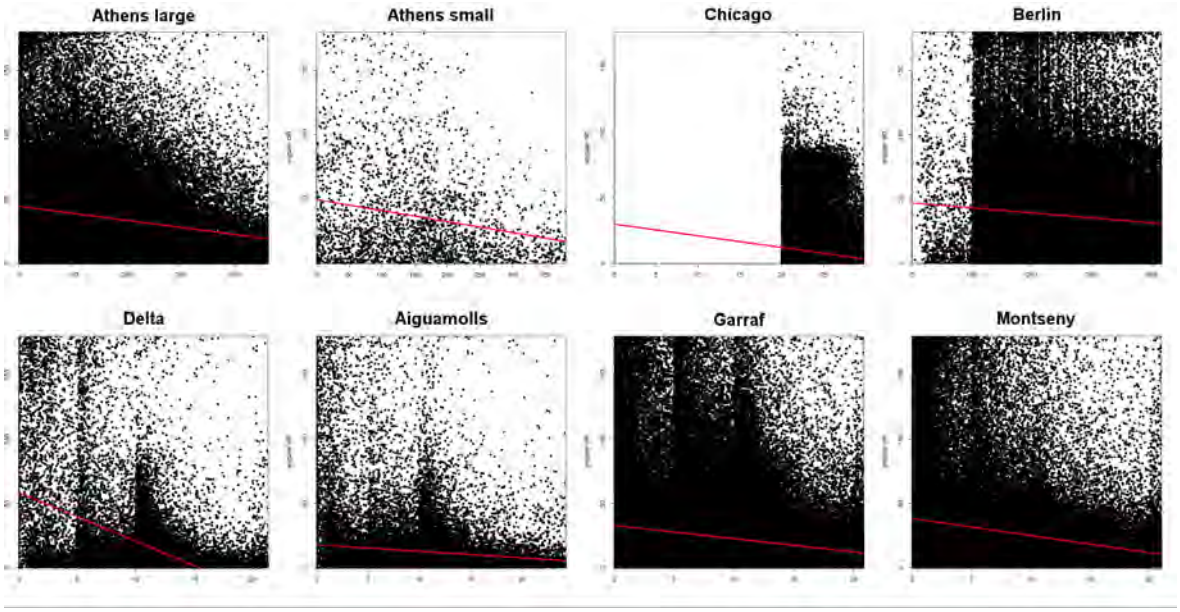


Figure 4: Angular differences in trajectory points and their distance to the following point within the trajectory. Regression lines appear in red. The horizontal axis indicates distances (percentiles $0\% - 95\%$). The vertical axis the angular difference ($0° - 180°$). Angular difference of $0°$ means completely straight whereas $180°$ is a complete turn-back. To avoid outliers, only the first 95% of distances are considered.

at [23]. It is important to note that two of the implementations (CK and DBH) were not the original ones by the authors of the algorithms, but re-implementations by the authors of [3].

The trajectories for each data set were preprocessed to guarantee a minimum spatial and temporal consistency, as described below.

The parameters of each algorithm were set for each of the four data sets, based on the terrain and trajectory characteristics of each. This is described in detail in Section 4.1.

Each algorithm was executed for each of the four data set, obtaining a total of 20 maps. Each map was inspected visually in an overlay with two other layers: the input trajectories, and the Google Earth satellite and aerial images. During this inspection we identified a large number of areas where the generated map was visually erroneous, in relation to the trajectories, aerial images, and official cartographic maps. We grouped and classified these areas into 17 groups, based on the visual appearance of the phenomenon (e.g., shortcut in a zig-zagging path), that we call *visual artifacts*. That wide variety and representativeness of the artifacts identified allowed us to perform a thorough analysis of the behavior of algorithms. For each visual artifact, we proposed a measure and analyzed its possible causes in terms of four factors: terrain features, path characteristics, aspects of trajectories, and strategy of the algorithm. The results of this analysis are presented in Section 5.

3.1 Preprocessing of trajectory data

Before using the trajectories as input to the algorithms under comparison, we have preprocessed the data in order to guarantee their consistency. The preprocessing addressed three aspects.

1. **Time consistency.** Points that are consecutive in a trajectory should have consecutive increasing time stamps. However, this was not always the case: approximately 10% of the trajectories had significant backtracks in time, ranging from days to years, probably due to malfunctioning of the GPS receiver. Figure 5 shows three examples. Since these deviations didn't follow any clear pattern, we decided to discard these trajectories. Another anomaly detected in time stamps was consecutive points with identical time stamps; trajectories with this anomaly were kept.

2. **Format consistency.** Wikiloc uses the default GPX format [27] of GPS trajectories. We have projected them into UTM (zone 31T), ignoring the elevation, since the considered algorithms

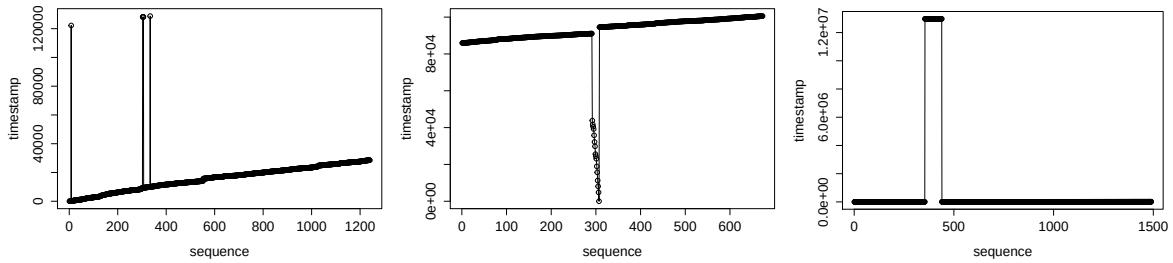


Figure 5: Examples of time stamps error examples. The horizontal axis follows the order of the sequence of points in the trajectory, the vertical axis shows the time tamps, translated to zero.

only use x, y-coordinates.

3. **Geometric extent.** Trajectories have been segmented as to eliminate the portions external to the rectangular windows described in Section 2.1.

3.2 Local versus global evaluations

Our method of evaluation is qualitative at the phase of defining and detecting each artifact, and quantitative when it comes to evaluating to what extent a certain artifact is present in the output of each algorithm. The goal of our experiments was to detect the most important deficiencies of current algorithms, at a local level, in reconstructing maps from data, and to investigate their causes. In order to do that, we carefully analyzed the resulting maps in search of their limitations. Arguably, a weak point of this methodology is that the definition and detection of artifacts is qualitative and somewhat subjective.

In theory, a better way to do so would have been to measure the quality of the maps with respect to their corresponding ground-truth using some quantitative measure. As mentioned in the introduction, several of these measures have been proposed and used. For instance, four measures are used in [3]. The *directed Hausdorff distance* [6] measures the maximum distance between a point on the ground truth and one on the constructed map. The *path-based distance* [2] measures the maximum Fréchet distance between one path in the ground truth and one on the constructed map. In the *shortest-path based distance* [19], the comparison between maps is restricted to a selection of shortest paths connecting vertices identified on both maps, using discrete Fréchet and average vertical distance to compare each pair of paths. More different is the *graph-sampling based distance* [7], which measures random samples of the same location on each map, and compares the number of matched points on each side. This has the goal of estimating the similarity between the geometry and topology of the maps at the chosen locations.

However, we have not followed this quantitative approach for several reasons.

Firstly, quality measures are numeric indicators that, though useful for comparing different methods, do not give information on the concrete problems encountered by the algorithm in reproducing the exact map, which is what we are interested in in this work. Moreover, measures like the ones mentioned above focus on particular aspects of the final map (e.g., proximity between the points forming the map, or preservation of shortest paths), which are not the main focus for us.

Secondly, and most important, quality measures are global indicators, while our goal is to locally understand the behavior of the algorithms.

Finally, all proposed quantitative measures require reliable ground truth data sets [3], which not always exist for hiking data sets, as a large number of paths are missing in the available maps. See Figure 6 for an example.

For all these reasons, our analysis starts necessarily by a qualitative identification of visual artifacts. Once the different common artifacts are identified, for each artifact we have defined a local ad-hoc measure quantifying to what extent a generated map presents that artifact. The measures are local in the sense that they are taken over a local window of the map. They are ad-hoc in the sense that their definition depends upon the artifact under study although, except for one of them, they essentially reduce to (windowed) length or area ratios between the generated map and the corresponding ground truth. This allows to objectively compare the output of each algorithm for the region where the artifact is illustrated.

Figure 6: Example showing the lack of a reliable ground truth in *Montseny*. Left: the large amount of trajectories departing from the center of the image towards the right suggests that there must be a path from there. However, this is not apparent in the aerial image (center), and neither is present in the largest-scale available topographic map (right).

4 Algorithms

In this section we present in more detail the algorithms that have been compared. The selection consists of five out of the seven algorithms compared by Ahmed at al. [3].[5] After some general remarks about the types of parameters used in map construction algorithms, we focus on the details of each of the algorithms evaluated. For each of them we provide an outline of the method, mention the original parameters used, and discuss how these parameters were adjusted for our hiking data sets in order to obtain the best possible results for each method. For the sake of conciseness, and given its very technical nature, part of this discussion is deferred to the supplementary material in Appendix A. However, we should emphasize that understanding the effect of each parameter is crucial to obtain the best possible results from each method, and thus to provide a fair comparison between them. In addition, we believe that this information will be valuable for future users of map construction algorithms that will have to adjust these or similar algorithms to their own data.

4.1 Parameterization

All the compared algorithms require that some parameters are adjusted to each specific data set. Among the different algorithms we were able to establish several groups of parameters whose reasoning are similar and, thus, were parameterized in a similar way. In this section we provide a conceptual overview of the main types of parameters that the algorithms use. A detailed description of all the parameters that each algorithm uses is presented in Sections 4.2–4.6.

Spatial proximity-related parameters Almost all the analyzed algorithms use to some extent a distance threshold indicating which (portions of) trajectories are close enough to be considered samples of the same underlying path. The value of this threshold is usually adjusted according to the maximum width of the paths (*max_width*), the expected (in a Gaussian distribution) or maximum GPS error of the trajectories (*gps_error*), and the minimum separation between different paths (*min_sep*). One notable exception is KP, whose proximity parameter relates to the distance between turns of the same intersection. Table 4 summarizes which terrain and data features each algorithm uses for its spatial proximity-related parameters.

Algorithm	Parameters	*max_width*	*min_sep*	*gps_error*
AW	ε	✗	✓	✓
CK	M, σ_1, k	✓	✓	✓
DBH	σ	✓	✓	✗
ES	θ	✓	✗	✓
KP	d_t	✗	✗	✗

Table 4: Terrain and data features used by the different algorithms' spatial proximity parameters.

[5] The reasons not to include two of the algorithms were: implementation not available for one of them, and technical issues when trying to reproduce previous experiments in the other case.

Angular parameters The algorithms CK, ES, and KP use the angular difference of the trajectory points around intersection areas in order to distinguish to which path each trajectory belongs. Algorithm KP further extends this approach by detecting turns according to angular difference. Moreover, CK also considers the angular difference in order to prevent merging orthogonal trajectories.

Reliability parameters Reliability parameters are thresholds indicating how much sampled a path must be in order to be included in the final map. They are user-defined in the sense that they try to capture the concept of reliability of the generated map, something heavily dependent on the purpose of the final application. The algorithms including a reliability parameter are CK (*min_vol*) and DBH (*threshold*).

Segmentation parameters Some algorithms segment trajectories whenever they detect a discontinuity in time or space between samples of the same trajectory, see Table 5. It should be noted that although not specified in their respective papers, the original code of KP and AW, also includes such preprocessing.

Although one could see similarities between the reliability and the segmentation parameters, the latter depend much more on the trajectories rather than on the final application. We have adjusted each parameter using its original value and the ratio between the mean distance (or elapsed time) between trajectory points of the original data sets and each of our data sets.

Algorithm	*temporal discontinuities*	*spatial discontinuities*
AW	✓	✗
CK	✓	✓
DBH	✗	✗
ES	✗	✗
KP	✗	✓

Table 5: Segmenting criteria used by each algorithm.

Although most of the parameters used by the algorithms lie in one of the aforementioned groups, some of them are specific to each algorithm and, thus, their value cannot be easily generalized across different algorithms. Some examples are termination criteria or ad-hoc approaches only used in one of the algorithms, like the maximum permitted speed in turns used by KP. We discuss these more specific parameters in the following sections.

4.2 Algorithm AW

The algorithm by Ahmed and Wenk [5] is incremental: starting from an empty graph, it adds the trajectories one by one, in two steps. In the first step, the portions of the current trajectory that are present in the current partial map are identified. This *partial map-matching* is done using a variant of the Fréchet distance. In the second step, the portions that have not been matched are inserted into the current map, adding new vertices and edges as needed.[6]

Parameters A remarkable feature of this algorithm is that it uses only one parameter, ε, plus a temporal parameter for segmenting erroneous trajectories. The meaning of ε is related to the minimum separation between different roads, and the minimum distance between two intersections. This is formalized in [5] with the following assumptions on the ground truth: (i) any road in a map has a portion that is *well-separated* from the others, meaning that its distance to all other roads is at least 3ε; (ii) if two roads become closer than 3ε at some point, then they must share a vertex. Two assumptions on the input trajectories are also made: (iii) each input trajectory is within Fréchet distance $\varepsilon/2$ from some road in the ground truth; (iv) all input trajectories sample an acyclic path in the ground truth.

In practice, the value of ε determines the distance threshold to consider that (portions of) two trajectories correspond to the same or different roads. The effect of this parameter is illustrated in

[6] In the original paper [5] a third step is described, where identified portions can be adjusted. However, the implementation made public ignores this third phase.

Data set	ε [m]	Data set	ε [m]
Athens large	180	Delta	15
Athens small	90	Aiguamolls	25
Berlin*	82.5, 170	Garraf	35
Chicago	80	Montseny	40

Table 6: Different ε chosen values for the AW algorithm. The values for city data sets are from [3]. *Value 82.5 was used in [5], while 170 was used in [3], for the same data set.

Appendix A.1 (see Figure 9). However, the assumptions made on ε are not satisfied by any of our four hiking data sets, so setting this parameter is not straight-forward. The values of ε chosen are summarized in Table 6. See Appendix A.1 for the justification of these values.

4.3 Algorithm CK

The algorithm by Cao and Krumm [10] is an incremental insertion algorithm that has two previous phases: a *preprocessing* phase and a *clarification* phase.

The *preprocessing* phase consists in segmenting erroneous trajectories, and reducing over-sampled straight portions of trajectories while preserving samples that lie on portions with high curvature.

The *clarification* phase intents to reduce the effects of noise in the input trajectories, by using a force-directed algorithm that cleans up the initial data. The clarification phase simulates physical attraction and repelling forces between trajectory nodes and segments, so that points that correspond to the same route get close together, while those of different routes move further away. More precisely, each trajectory segment exerts an attraction force towards all other nodes that locally have a similar heading to the edge. The attraction force is modeled as a inverted Gaussian function, thus the strength of the force decreases exponentially with distance. In addition, each node exerts a conservative force that attracts the node to its original position, implemented as a spring force, thus the force strength increases linearly with distance.

The second phase of the algorithm inserts incrementally each of the clarified trajectories, starting from an empty map. The incremental insertion is based on inserting trajectory nodes one by one. If the current node has a local direction similar to that of an existing edge that is close enough, it is not inserted. Otherwise, it is inserted into the map, adding connections to other existing vertices depending on some connectivity criteria.

Parameters The CK algorithm uses several parameters for the preprocessing, the clarification process, and the incremental algorithm.

In the preprocessing, temporal and spatial gaps within a trajectory that are greater than two thresholds t and d_1 are eliminated. Moreover, consecutive sample points within a trajectory that are too close to each other are removed, based on two other thresholds: one for all sample points (d_2) and another one (d_3) used when the angular difference is smaller than a parameter α.

The two types of forces used in the clarification phase need to be carefully adjusted to produce the desired effect. This is done by tuning several parameters. The attraction forces are parameterized by two values: M and σ_1, while the spring force depends on one parameter k. Finding values for these parameters is a delicate task, and requires in turn finding values for three parameters from the data: average number of trajectories on a path (N), expected GPS error (σ_2), and an intermediate value between the maximum path width and the minimum separation between two different paths in the terrain (t_d). Some of these parameters have some more or less direct translation. For instance, σ_1 and σ_2 are related to the GPS error and the width of roads, while N has to do with the expected number of trajectories that go through the same route. According to Cao and Krumm [10], the values should be chosen so that the force of one edge attracts all vertices with similar direction within a certain distance, and not those further apart. However, as described in Appendix A.2, for our data sets it was not possible to find values that fit this description, thus we had to set them empirically after a careful study of each data set.

Data set	Data			Segm.		Redund.			Clarification			Insertion			
	N	σ_2	t_d	d_1	t	d_2	d_3	α	M	σ_1	k	d_4	β	v	h
Original	20	5.0	25.0	100	10	10	30	10	1.0	5.0	0.005	20	45	3	5
Delta	20	2.7	6.5	159	102	1.9	5.8	10	1.0	2.7	0.05	5	45	3	5
Aiguamolls	5	1.7	9.5	130	73	2.0	6.1	10	1.0	3.0	0.02	6	45	3	5
Garraf	80	5.0	10.0	135	86	1.8	5.4	10	1.0	5.0	0.025	5	45	3	5
Montseny	30	7.3	7.5	100	86	1.3	4.0	10	1.0	7.3	0.025	5	45	3	5

Table 7: Values for each parameter used in CK. Angular values are measured in degrees, distance values are given in meters.

Finally, the incremental graph generation algorithm is governed by four parameters: (i) a distance threshold d_4 and (ii) an angular threshold β used to decide when a node should be inserted into the current map; a (iii) *minimum volume* v: a reliability threshold to exclude from the final map edges with few trajectory representatives, and a (iv) *maximum detour* h, which determines the maximum number of extra hops that are allowed in the map before a direct (existing) connection to a node is inserted.

The values chosen are summarized in Table 7. See Appendix A.2 for their justification.

4.4 Algorithm DBH

The algorithm by Davies et al. [12] applies image-processing techniques to generate a map in four main steps. Based on a rectangular grid that covers all the map area, a 2D histogram is built, measuring the total length of trajectory edges that go through each cell. The value in each cell gives an estimate of the likelihood of having a road passing through the cell. A Gaussian blur filter is then applied to the histogram, to remove small gaps due to GPS noise. In a second step, road positions are computed by binarizing the histogram using a global threshold value. Then, polygon boundaries for the road's region are extracted from the binary image, using a contour follower. The third step consists in producing the road centerlines from the extracted polygons by computing the straight skeleton of each polygon. Due to noise and other inaccuracies that can create cavities in the road polygons, the resulting skeletons may have a large number of short edges (*hairs*). Therefore, a final procedure is applied to filter all skeleton edges shorter than some *shaving* threshold. However, it should be noted that the available implementation performs the *shaving* in a different way, different from that proposed by Davies et al. [12], thus so we opted to ignore this step. The fourth step of the algorithm determines the direction of travel permitted along each road. We omit this part as it is not relevant to our setting.

Parameters The algorithm uses three parameters. (i) The grid *cell size* determines the resolution of the map, and has a high impact in the final result and running time of the algorithm. (ii) The Gaussian filter used to remove small gaps depends on a parameter σ (kernel size, the Gaussian σ is automatically computed), and (iii) a *mask threshold* is used to decide how to binarize the blurred histogram. (iv) Finally, in the original paper a *minimum road length* value is used to decide, in the third step, when skeleton segments are too short, and thus should be removed. However, this was not part of the implementation provided.

The parameter values chosen are summarized in Table 8. See Appendix A.3 for their justification.

4.5 Algorithm ES

The algorithm by Edelkamp and Schrödl [15] is based on point clustering. A first initial set of uniformly distributed point samples is taken from trajectory edges (keeping their original heading), guaranteeing a minimum sample density (namely, such that every trajectory vertex has a point sample within distance $d_{\max}$, for $d_{\max}$ a parameter—see below). Next each point in the sample is taken as an initial cluster of points Each cluster is iteratively refined by considering other trajectories passing

Data set	*cell_size* (m)	σ	*mask_threshold*
Original	2	17	100
Delta	1	5	35
Aiguamolls	1	7	15
Garraf	0.5	7	15
Montseny	0.5	5	15

Table 8: Chosen values for the parameters of DBH.

close to its center, and by adjusting the position and heading of the center of the cluster based on the new points. The procedure is very similar to the k-means clustering method.

After refining the clusters of points, the cluster centers become the nodes of the final map. Then, edges between pairs of map nodes are added if at least one trajectory is associated to both clusters. This results in a first approximate map where each edge represents the bundle of (portions of) trajectories that consecutively belong to both clusters (map vertices) forming the edge. Notice that these bundles may contain portions of trajectories that belong to different lanes of the same road. In the implementation available, this approximate map is the final result. However, in the original paper [15] a further refinement phase is proposed.

Parameters The algorithm uses three parameters: (i) a distance threshold $d_{\max}$ that sets the maximum distance between the initial clusters, (ii) a cluster bearing tolerance δ that determines when two trajectory segments are to be considered parallel, and (iii) an intra-cluster distance tolerance θ used to decide when a point belongs to an existing cluster (i.e., it is considered part of the cluster when the distance between the point and the cluster center is less than θ).

The role of parameters δ and θ is particularly important, because they provide the membership test for a trajectory point to belong to a cluster. Specifically, given a cluster defined by its center point and its heading, the trajectories that belong to the cluster are those that fulfill the following three conditions: i) the trajectory intersects the line l that is orthogonal to the heading and passes through the cluster center; ii) the heading of the trajectory differs from the cluster heading by at most δ; iii) the intersection point between l and the trajectory is at distance at most θ from the cluster center.

The parameter values chosen are summarized in Table 9. See Appendix A.4 for their justification.

Data set	$d_{\max}$ (m)	δ (°)	θ (m)
Original [15]	50	45	20
Urban data sets [3]	-	45	50
Delta	20	45	8
Aiguamolls	20	45	8
Garraf	20	45	10
Montseny	20	45	12

Table 9: Values for the parameters used for ES.

4.6 Algorithm KP

The algorithm by Karagiorgou and Pfoser [19] first detects intersection points and then connects them to each other, based on the input trajectories. The description that follows is approximate and omits many details, which can be found in [19].

In an initial preprocessing, KP segments trajectories using a spatial criterion.[7] Then the main algorithm proceeds in four steps. First, potential turning points are identified based on changes in

[7] We note, however, that this is what is done in the implementation, but the criterion described in the original paper [19] is temporal.

direction and speed. These turning points are then grouped based on distance and type of turn (a discretized model with eight types of turns is used). These groups are clustered, based on a distance parameter, and the centroid of each cluster gives rise to one intersection point. In a second step, intersection nodes are connected by edges obtained from the input trajectories. This is done by "compacting" the trajectories that contribute to turns through the clusters associated with the intersection nodes, and finally producing a polygonal representation of the links between intersection nodes. The third phase consists in fitting to the generated map the remaining trajectories (those that do not contribute with turning points), which can generate adjustments in the polygonal representation of the edges. Finally, the last step eliminates occurrences of *triangular intersections*, a particular type of artifact that consists of a sort-of shortcut at certain turns.

Parameters The algorithm uses several parameters. For the input preprocessing, (i) a spatial threshold to segment the input trajectories. The turn clustering and intersection extraction steps rely on four parameters: (ii) an angular difference threshold and (iii) a mean "turning speed" to identify turns, (iv) a distance threshold to group turning points, and (v) another distance threshold to cluster the grouped turns into intersection nodes. In addition, (vi) a direction threshold is used for the network extraction phase (steps two and three above). The values chosen are summarized in Table 10. See Appendix A.5 for the justification of these values.

Parameter	**Value used**
(i) *Max inter-node distance*	$1000 \cdot \frac{\textit{mean trajectory inter-node distance}}{\textit{mean original trajectory inter-node distance}}$
(ii) *Angular difference*	30º (*Aiguamolls*), 50º (*Delta*), 70º (*Garraf*, *Montseny*)
(iii) *Mean speed*	∞
(iv) *Turn clustering threshold*	25m (*Aiguamolls*), 10m (*Delta*), 30m (*Garraf*, *Montseny*)
(v) *Intersection clustering threshold*	$0.5 \cdot$ *value of (iv)*
(vi) *Direction threshold*	45°

Table 10: Fixed parameter values used for the KP algorithm. The values of parameters (ii) and (iv) were determined empirically for each data set.

4.7 Main differences between algorithmic strategies

Having presented the key ideas behind each algorithm, it is worth pointing out briefly the main differences between the methods, in addition to the differences already discussed from the point of view of their parameters in Section 4.1.

Following the strategy classification in [3, 4], we have that AW and CK use the paradigm of incremental construction. An important difference between them is that what AW adds at every step is (part of) an original trajectory, making it a pure incremental algorithm. On the one hand, this gives a great advantage to the method every time that it chooses to add good trajectories, because it can allow the method to be more robust in front of high noise or in difficult situations. On the other hand, choosing to insert a bad trajectory can affect very negatively the whole resulting map, since once inserted, it will be part of the final map. In contrast, CK adds clarified trajectories. The force-directed clarification process can be seen as a type of clustering, thus CK falls also into the clustering category (although it does not cluster points, but trajectories). The idea of inserting clarified trajectories, which are expected to be more reliable than individual original tracks is very interesting, since—if the clarification method works well—it should prevent many of the issues that incremental algorithms suffer due to picking bad trajectories to insert in the map.

ES is mainly a point clustering method, but one should take into account that the initial points used are sampled from the trajectory edges. Thus, in a way, it is also a density-based method, and the parameter used to decide the distance between samples will also play an important role.

KP is an intersection linking method, thus the criteria used to decide when a vertex is part of an intersection (in this case, based on direction and speed changes) is fundamental. KP is also, to a lesser extent, a point clustering method, since the turns detected are clustered to produce the final intersection nodes.

Finally, DBH is a density-based algorithm, and as such the whole success of the method will depend heavily in choosing an appropriate grid cell size. This method is the most different one from the five analyzed, since it follows a completely different approach. As such, it can be expected to have issues in very different situations than the other four methods.

5 Experimental results

The experiments consisted in running each of the five algorithms on each of the four hiking data sets, with the parameters derived in Section 4. Appendix C includes images of the maps generated by each algorithm, together with the input trajectories.

5.1 General observations

A first observation is that the generated maps for hiking data are about ten times more complex than those for urban areas, with respect to number of vertices per km^2 covered, see Tables 11 and 12. The table also includes information on the running time of all the algorithms on the different data sets.[8]

Generated map	**Vertices #**	**Edges #**	**Length [km]**	**Exec. time [min]**
Athens L. - AW	7067	7960	1358	-
Athens L. - KP	6584	5280	252	-
Athens S. - AW	344	378	35	-
Athens S. - CK	20	14	3	-
Athens S. - DBH	209	227	2	-
Athens S. - ES	526	1037	197	-
Athens S. - KP	660	637	35	-
Berlin - AW	1322	1567	164	-
Berlin - KP	2542	2262	161	-
Chicago - AW	1195	1286	34	-
Chicago - CK	2092	2948	78	-
Chicago - DBH	1277	1310	14	-
Chicago - ES	828	1247	83	-
Chicago - KP	596	558	26	-

Table 11: Complexity of generated maps. Data for urban setting from [3].

A second general observation is that the different data sets considered produce, as expected, contrasting results for the different algorithms. Indeed, as Figure 7 shows, trajectories on hilly and densely covered areas contain more noise than those in flat terrains, and they are more winding and run closer to each other. This translates into harder instances for the algorithms, as evidenced by results like the ones shown in the figure.

[8]The computer used was equipped with an Intel i5-2500K CPU and 8GB DDR3 Synchronous 1600 MHz memory.

Generated map	Vertices #	Edges #	Length [km]	Exec. time [min]
Delta - AW	2362	2459	395	1.26
Delta - CK	2667	2436	1810	40.55
Delta - DBH	10229	10197	45	3.19
Delta - ES	1028	1756	11029	2.00
Delta - KP	6787	4817	446	73.33
Aiguamolls - AW	13454	13516	2179	7.17
Aiguamolls - CK	10621	5308	2208	11.22
Aiguamolls - DBH	39786	39206	121	61.92
Aiguamolls - ES	4147	4918	40849	1.88
Aiguamolls - KP	21690	21810	1990	45.77
Garraf - AW	7827	7898	2005	24.88
Garraf - CK	13565	8172	4345	950.93
Garraf - DBH	88009	87162	363	245.69
Garraf - ES	5295	9320	30763	51.81
Garraf - KP	36487	36574	2229	1710.83
Montseny - AW	8893	8940	1721	15.42
Montseny - CK	19323	11625	2809	146.30
Montseny - DBH	83025	81783	214	214.81
Montseny - ES	4610	7774	9661	17.91
Montseny - KP	24329	24492	4478	351.62

Table 12: Complexity of generated maps in hiking setting.

(a) AW [5]

(b) KP [19]

Figure 7: Figure showing how noise in trajectory data, typical in mountain areas, affects the quality of the generated maps. Input trajectories shown blue, generated map in red. The first column is from *Aiguamolls*, the second is from *Montseny*. Images from Google Earth.

5.2 Detailed description of the artifacts found

We have carefully studied the resulting maps together with the input trajectories, the official local topographic maps available from the Geological and Cartographic Institute of Catalonia (ICGC), and the Google Earth imagery. Based on these data sources, we have detected and classified situations that were visually incorrect, with respect to the available ground truth (topographic maps and aerial photographs). We present a detailed analysis of the most relevant artifacts identified in our experiments. Some of the artifacts are common to the maps generated by several of the algorithms, while some others are specific to one single algorithm, because they are the result of some particular design decision. We present the two groups of artifacts in the following two sections. The color convention used in all the figures in both sections is as follows:

- Fine blue lines are the input trajectories.
- Thick lines are edges of the generated maps (output of the algorithms). The correspondence between colors and algorithms is as follows:
  - Red: AW
  - Yellow: CK
  - Green: DBH
  - Cyan: ES
  - Orange: KP
  - Dotted: ground truth

5.2.1 Common artifacts

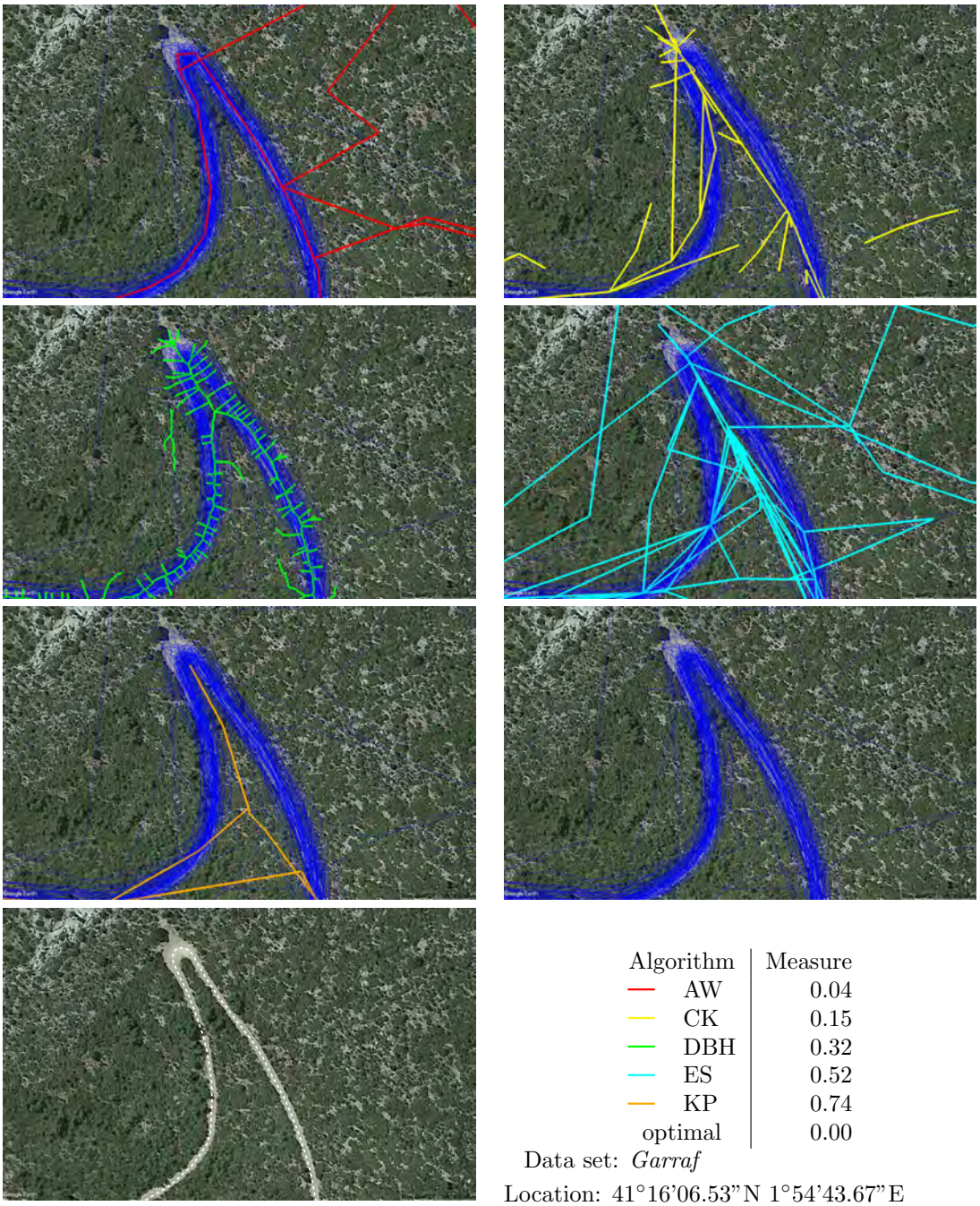
In this section we present an analysis of the main artifacts identified that are common to more than one of the output maps of the algorithms. For each of these artifacts, we present a concrete example for one of our data sets, including:

- A concise description of the artifact.
- A local measure on how much the artifact is present in the current example for each algorithm. As already mentioned in Section 3.2, each local measure is defined ad-hoc to each artifact, since none of the global measures used so far in the literature is appropriate for the local analysis we need to perform.
- Output images for the five algorithms, on top of aerial image.
- Image of input trajectories on top of aerial image.
- Image of ground truth on top of aerial image.
- Measurements for the five algorithms and ground truth (optimal) value.
- A brief explanation of possible causes of the artifact.

Artifact [$C1$]: Merged narrow curves

Description: Some algorithms tend to produce a y-shaped map at narrow curves.

Measure: Distance between ground truth's maximum-curvature turn point and output map bifurcation point, divided by distance between ground truth maximum-curvature turn point and the midpoint of the two "first" locations where all maps found the two portions of the path away from the curve. The measure value for optimal map reconstruction is 0. The measure tends to 1 as the merged portion of the curve increases. Values above 1 cannot occur.



Algorithm	Measure
AW	0.04
CK	0.15
DBH	0.32
ES	0.52
KP	0.74
optimal	0.00

Data set: *Garraf*

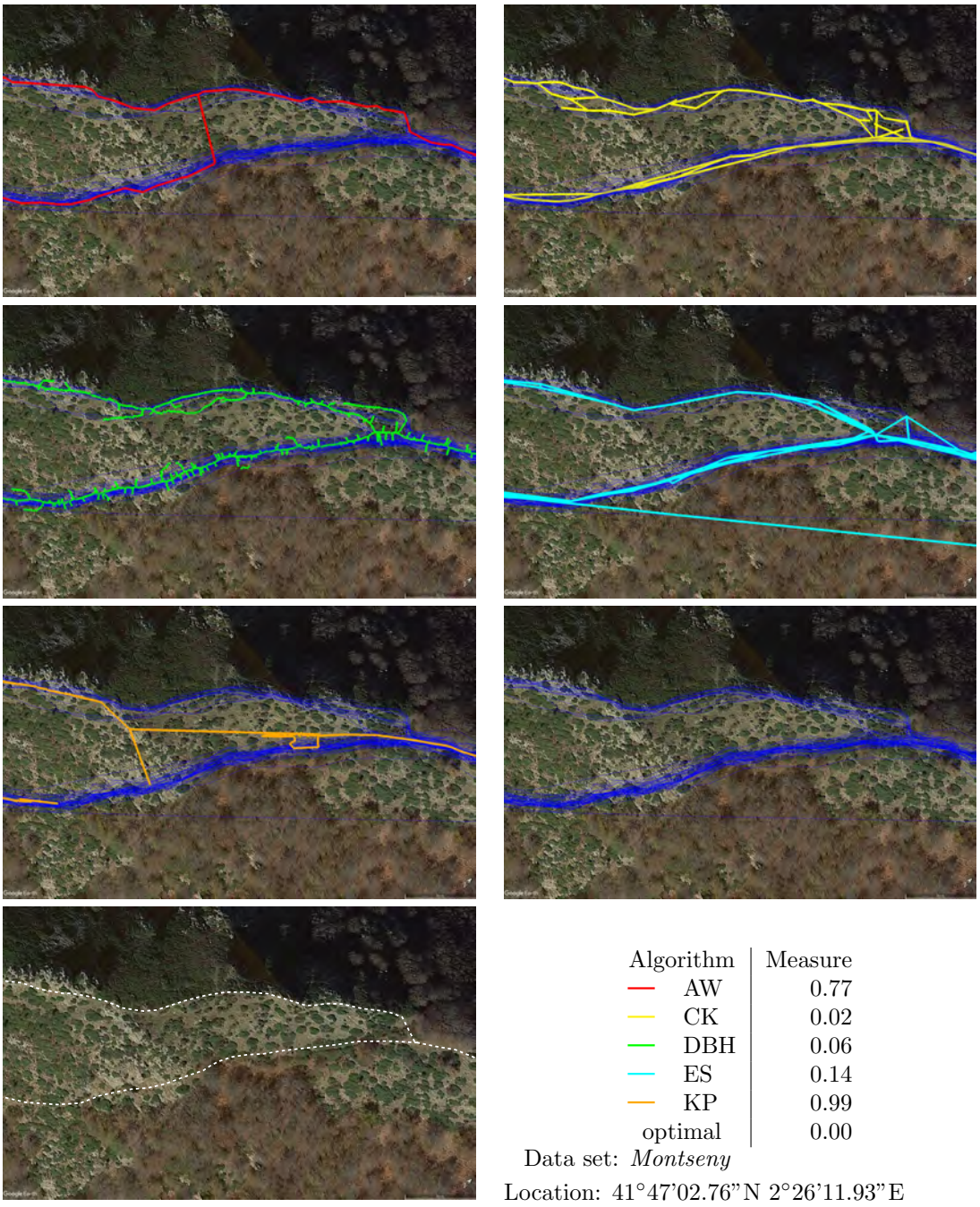
Location: 41°16'06.53"N 1°54'43.67"E

Causes: The worst results with respect to this artifact were exhibited by KP, ES and DBH. KP produces an excessive number of intersections, due to zig-zag combined with noise. ES identifies a portion of the zig-zag as parallel lanes. For DBH, the problem is caused by the underlying grid structure and the lack of direction information.

Artifact [$C2$]: Shortcuts at intersections

Description: At bifurcations with a small angle, some algorithms produce shortcuts.

Measure: Distance between ground truth bifurcation and found one, divided by distance between ground truth bifurcation and the midpoint of the two first locations where all output maps found the two paths. The measure value for optimal map reconstruction is 0. The measure tends to 1 as the found bifurcation point gets further from the real one. Values above 1 cannot occur.



Algorithm	Measure
AW	0.77
CK	0.02
DBH	0.06
ES	0.14
KP	0.99
optimal	0.00

Data set: *Montseny*

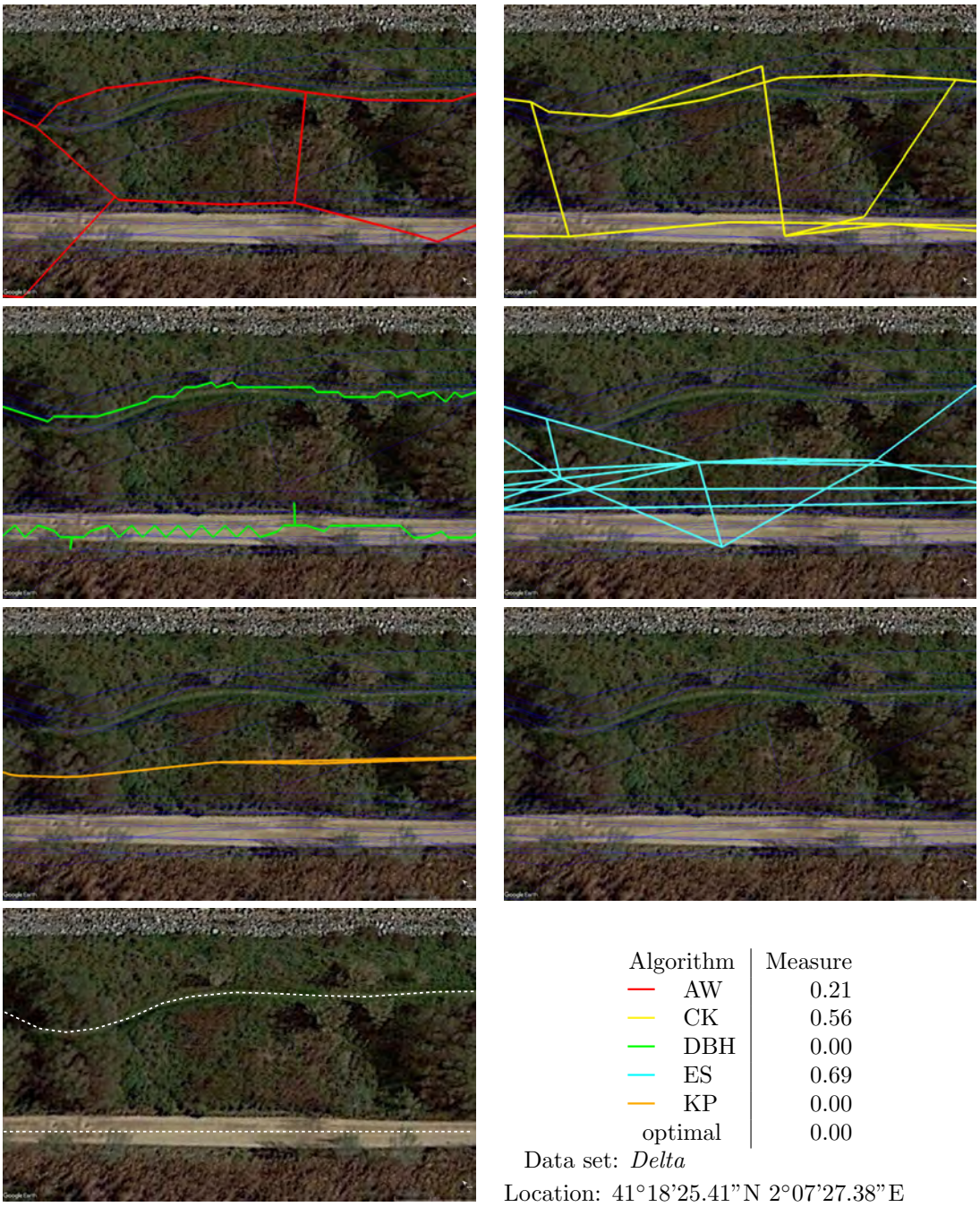
Location: 41°47'02.76"N 2°26'11.93"E

Causes: Algorithms KP and AW produce the worst results in this case. KP creates an excessive number of detected intersections that end up producing a distorted position for the bifurcation, due to the zig-zag combined with noise. AW misinterprets small angles, postponing the bifurcation until both paths become far enough with respect to the parameter ε.

Artifact [$C3$]: Artificial bridges

Description: Some algorithms produce nonexistent connections (*bridges*) between parallel paths.

Measure: Length of bridges divided by total length of upper and lower envelopes of all output edges. Bridges are defined as the union of all edges in shortest paths between a vertex in the upper envelope and a vertex in the lower envelope of the output map, excluding all edges belonging to the envelopes. The measure value for optimal map reconstruction is 0. The measure increases as the output map includes further and longer bridges.



Algorithm	Measure
AW	0.21
CK	0.56
DBH	0.00
ES	0.69
KP	0.00
optimal	0.00

Data set: *Delta*

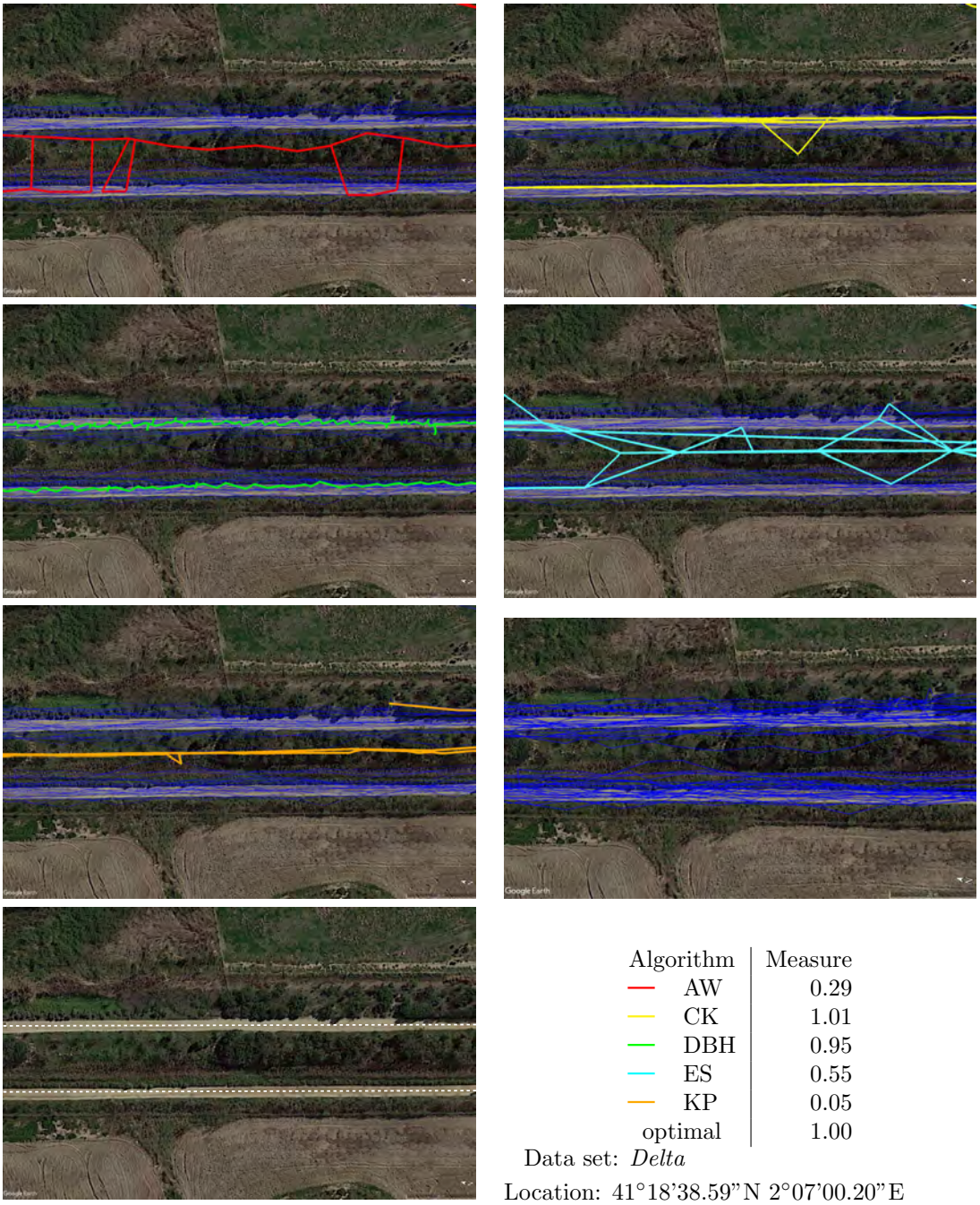
Location: 41°18'25.41"N 2°07'27.38"E

Causes: Close distance between parallel paths combined with noise makes ES, CK and AW exhibit the worst behaviors in this case. ES identifies trajectories partially belonging to parallel paths as belonging to parallel lanes of the same road, merging them. Due to the noise, several of the bottom trajectories are "attracted" by the top trajectories in CK. The final connecting step of AW connects pieces of the paths, since they become further than the specified threshold.

Artifact [$C4$]: Merged parallel paths

Description: Parallel paths at close distance are partially merged.

Measure: Area enclosed by the upper and lower envelopes of all edges in the output map, divided by the analogous area in ground truth. The measure value for optimal map reconstruction is 1. The measure tends to 0 as the merged portions increase.



Algorithm	Measure
AW	0.29
CK	1.01
DBH	0.95
ES	0.55
KP	0.05
optimal	1.00

Data set: *Delta*

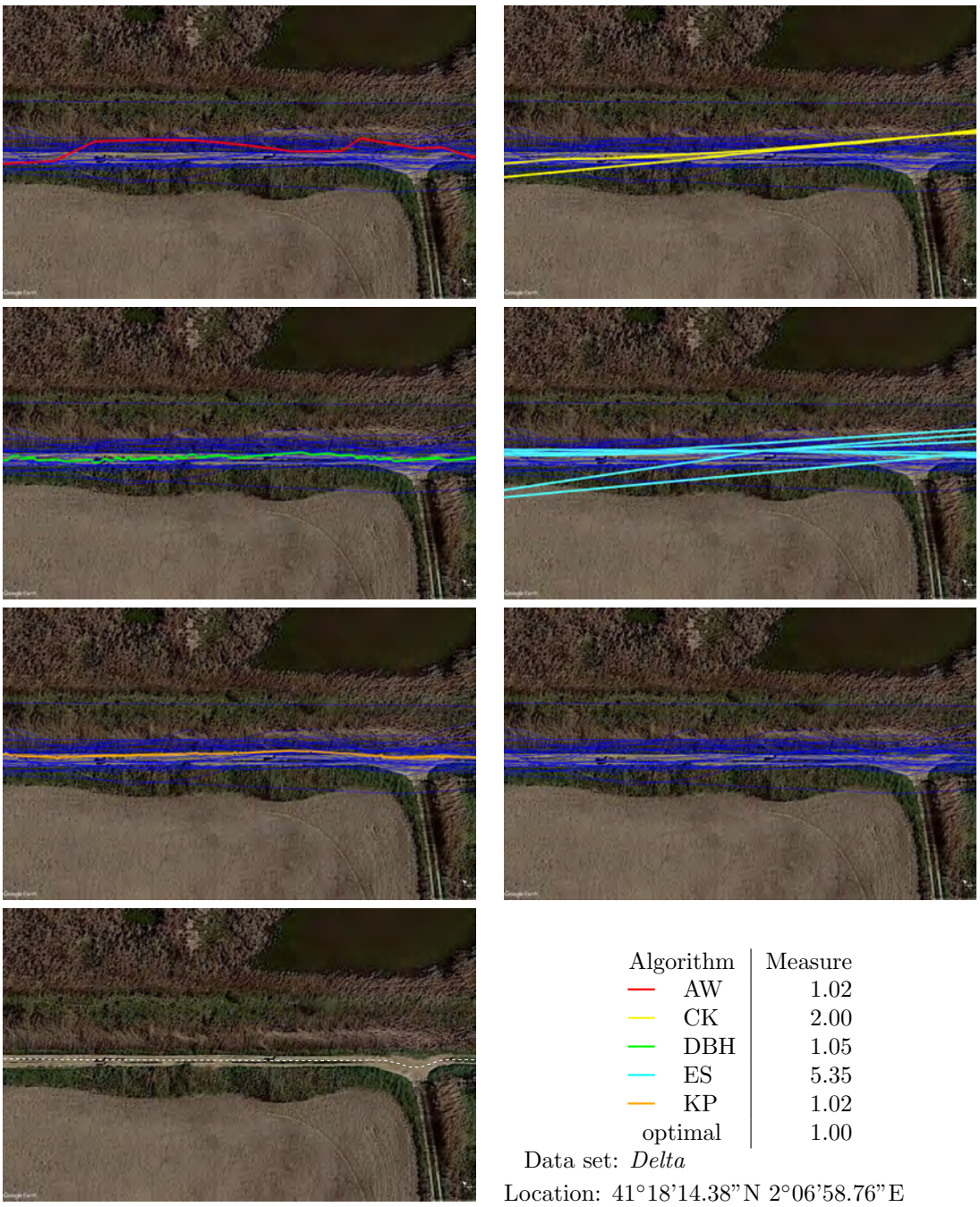
Location: 41°18'38.59"N 2°07'00.20"E

Causes: The worst results are produced by KP, AW and ES. KP detects too many vertices and ends up connecting clusters of turns that coincide on both paths. An incremental strategy, combined with noise, produces the merging in AW. ES identifies parts of parallel trajectories as parallel lanes of the same path, merging them. Notice that the value above 1 of CK is to be considered optimal from the viewpoint of this artifact, as it indicates that no merging occurred.

Artifact [$C5$]: Duplicated paths

Description: In the presence of several trajectories along one path, some algorithms generate duplicate paths.

Measure: Length of output map, divided by length of ground truth. The measure value for optimal map reconstruction is 1. The measure increases as the output map includes further copies of (portions of) the path.



Algorithm	Measure
AW	1.02
CK	2.00
DBH	1.05
ES	5.35
KP	1.02
optimal	1.00

Data set: *Delta*

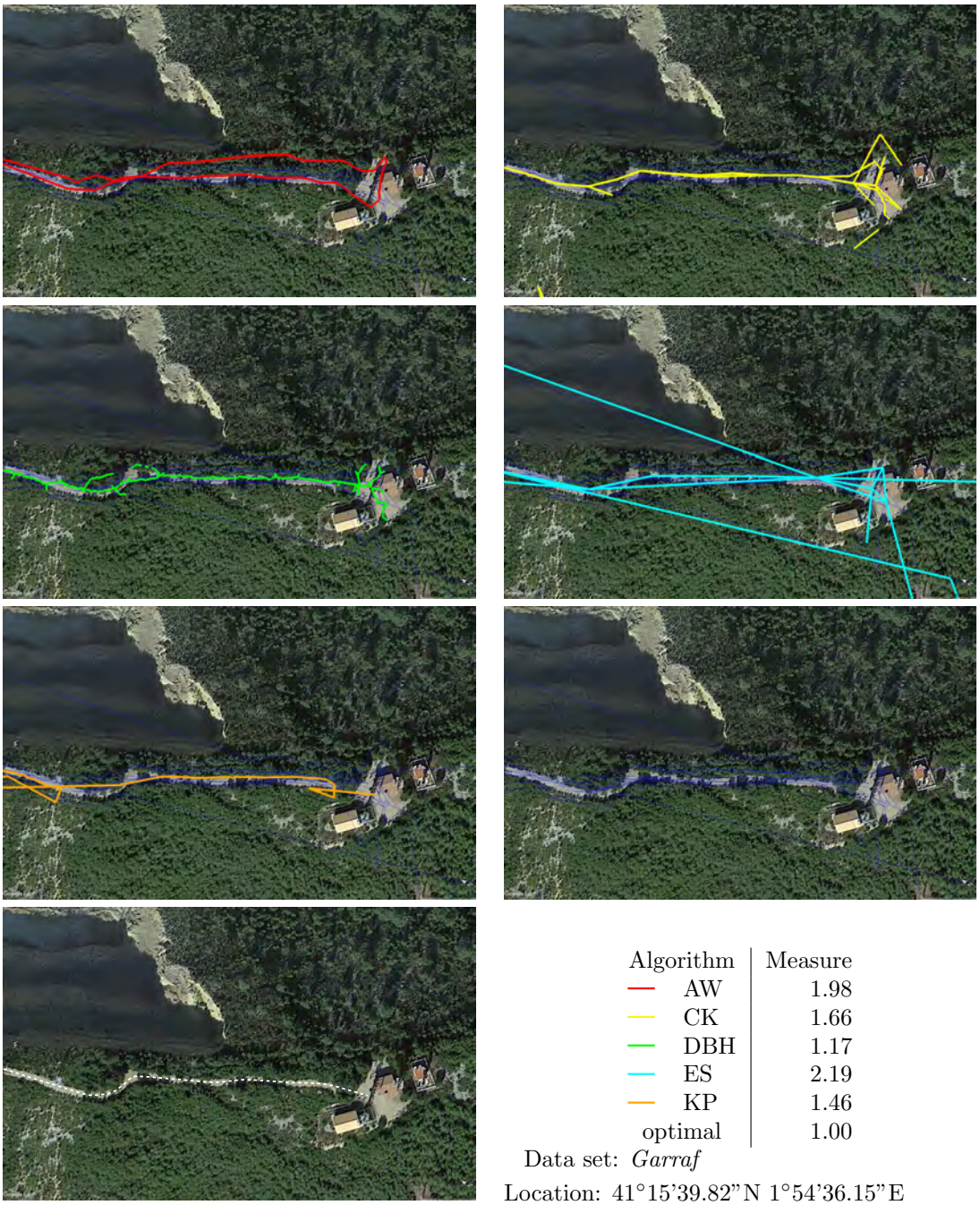
Location: 41°18'14.38"N 2°06'58.76"E

Causes: The worst outputs in this case are those of ES and CK. ES produces erroneous connections between the (too many) cluster centers detected. The disparity of sampling rates makes the clarification phase of CK perform poorly, failing at merging some duplicate paths.

Artifact [$C6$]: Duplicated back-and-forth paths

Description: In the presence of back-and-forth trajectories along one single path, some algorithms duplicate portions of the path.

Measure: Length of output map, divided by length of ground truth. The measure value for optimal map reconstruction is 1. The measure tends to 2 as the duplicated portions increase.



Algorithm	Measure
AW	1.98
CK	1.66
DBH	1.17
ES	2.19
KP	1.46
optimal	1.00

Data set: *Garraf*

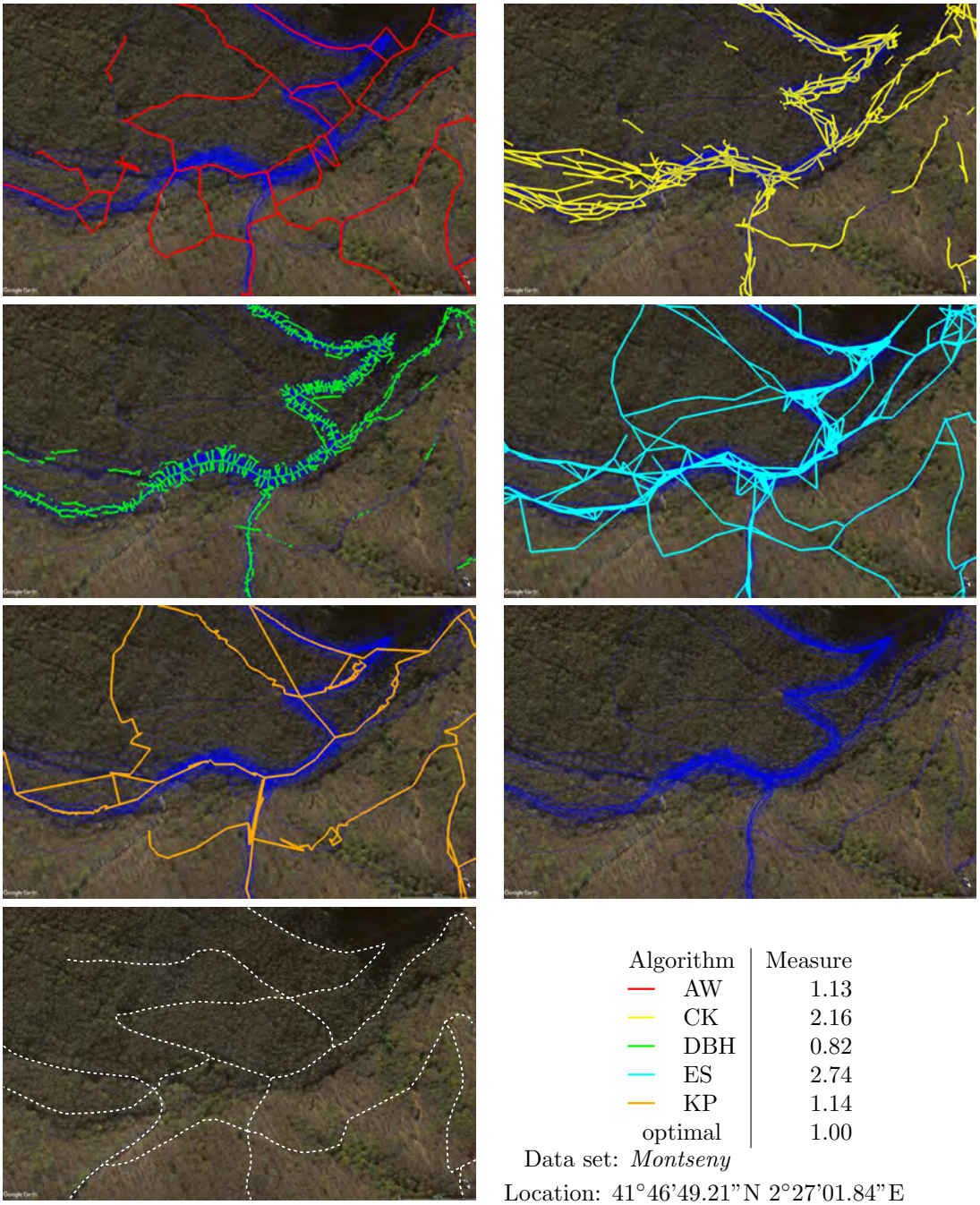
Location: 41°15'39.82"N 1°54'36.15"E

Causes: The worst results for this artifact correspond to ES and AW, followed by CK and KP. The value > 2 for ES indicates that some portions of the path appear more than twice, due to global parameter incompatibility (i.e., distance between intersections smaller than data inaccuracy). The incremental nature of AW makes it choose a whole piece of trajectory that duplicates the path. Most duplication in CK comes from direction issues; the short paths at the end of the main path are due to using hop distance for the detour parameter (h). The use of angular differences to detect turns in KP leads to many intersections that are later connected.

Artifact [$C7$]: Excessive number of connections in area

Description: Excessive number of vertices and connections in an area with multiple crossing paths.

Measure: Length of output map, divided by length of ground truth. The measure value for optimal map reconstruction is 1. The measure increases as the output map includes further connections. Values below 1 indicate incomplete map reconstruction.



Algorithm	Measure
AW	1.13
CK	2.16
DBH	0.82
ES	2.74
KP	1.14
optimal	1.00

Data set: *Montseny*

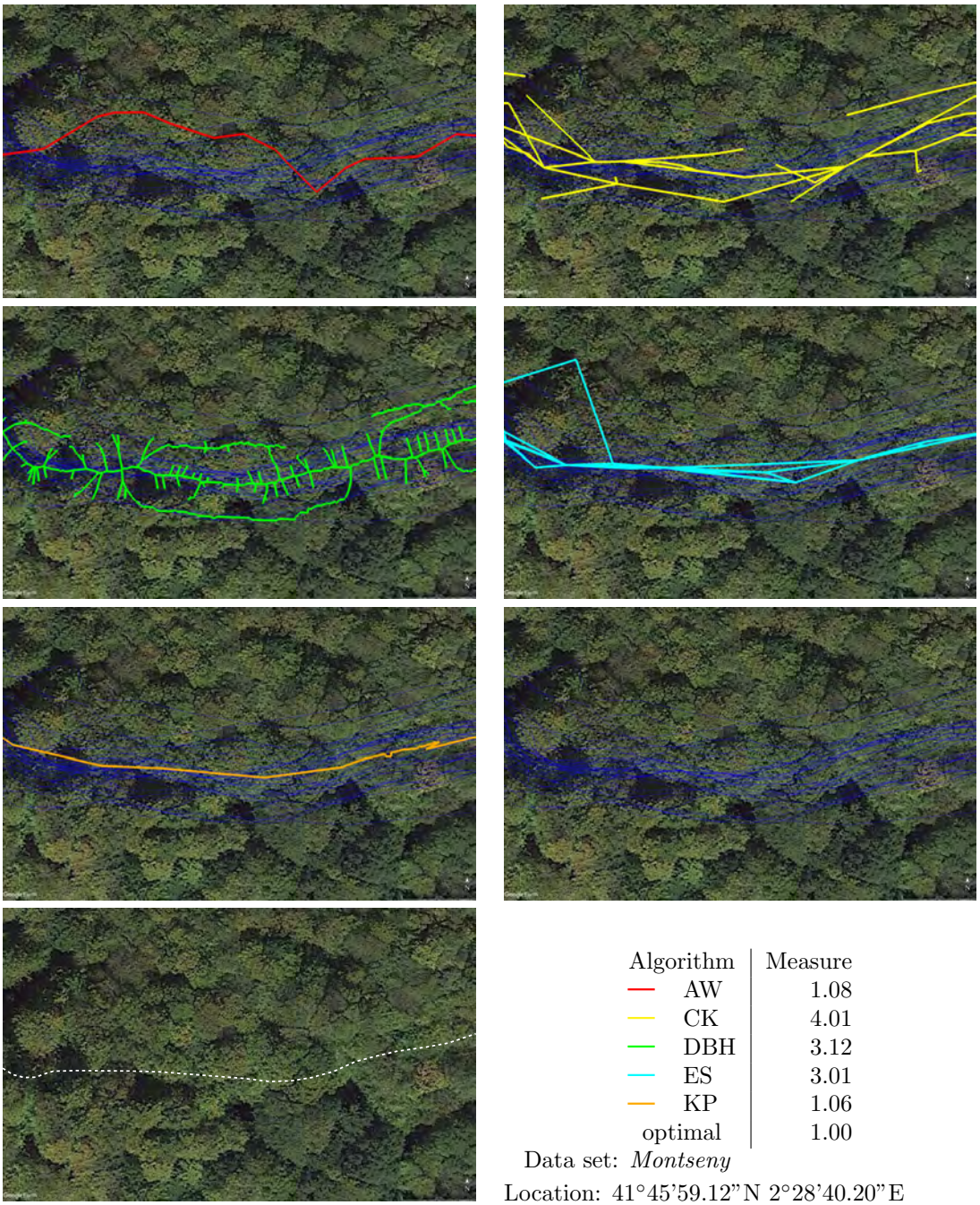
Location: 41°46'49.21"N 2°27'01.84"E

Causes: The worst behaviour is exhibited by ES and CK. For ES this is due to the lack of reliability measures combined with global parameter incompatibilities. The force-based clarification phase of CK has issues coping with many outliers. On the other hand, KP and AW do not perform as poorly as ES and CK, but still leave a lot of room for improvement. For KP, this is the result of the lack of a reliability measure combined with the fact that the algorithm creates too many vertices. For AW, it is a combination of the lack of a reliability measures and the incremental construction approach.

Artifact [$C8$]: Excessive number of connections along single path

Description: Excessive number of vertices and connections along one single path.

Measure: Length of output map, divided by length of ground truth. The measure value for optimal map reconstruction is 1. The measure increases as the added connections between path points increase in length. Values below 1 indicate disconnection (see Artifact C_9).



Algorithm	Measure
AW	1.08
CK	4.01
DBH	3.12
ES	3.01
KP	1.06
optimal	1.00

Data set: *Montseny*
Location: 41°45'59.12"N 2°28'40.20"E

Causes: The worst performance is that of CK, caused by a force-based approach that is very sensitive to outliers. Follows DBH, due to the size of the grid it uses, and ES, that makes erroneous connections between the detected intersections, due to global parameter incompatibility issues (i.e., distance between intersections smaller than data inaccuracy).

Artifact [C9]: Fragmented paths

Description: Pieces of single path missing in generated map.

Measure: Length of output map, divided by length of ground truth. The measure value for optimal map reconstruction is 1. The measure tends to 0 as the missing portions of the path increase in length. Values above 1 indicate duplication (see Artifact C_5).

Algorithm	Measure
AW	0.93
CK	0.72
DBH	0.80
ES	2.99
KP	0.98
optimal	1.00

Data set: *Aiguamolls*

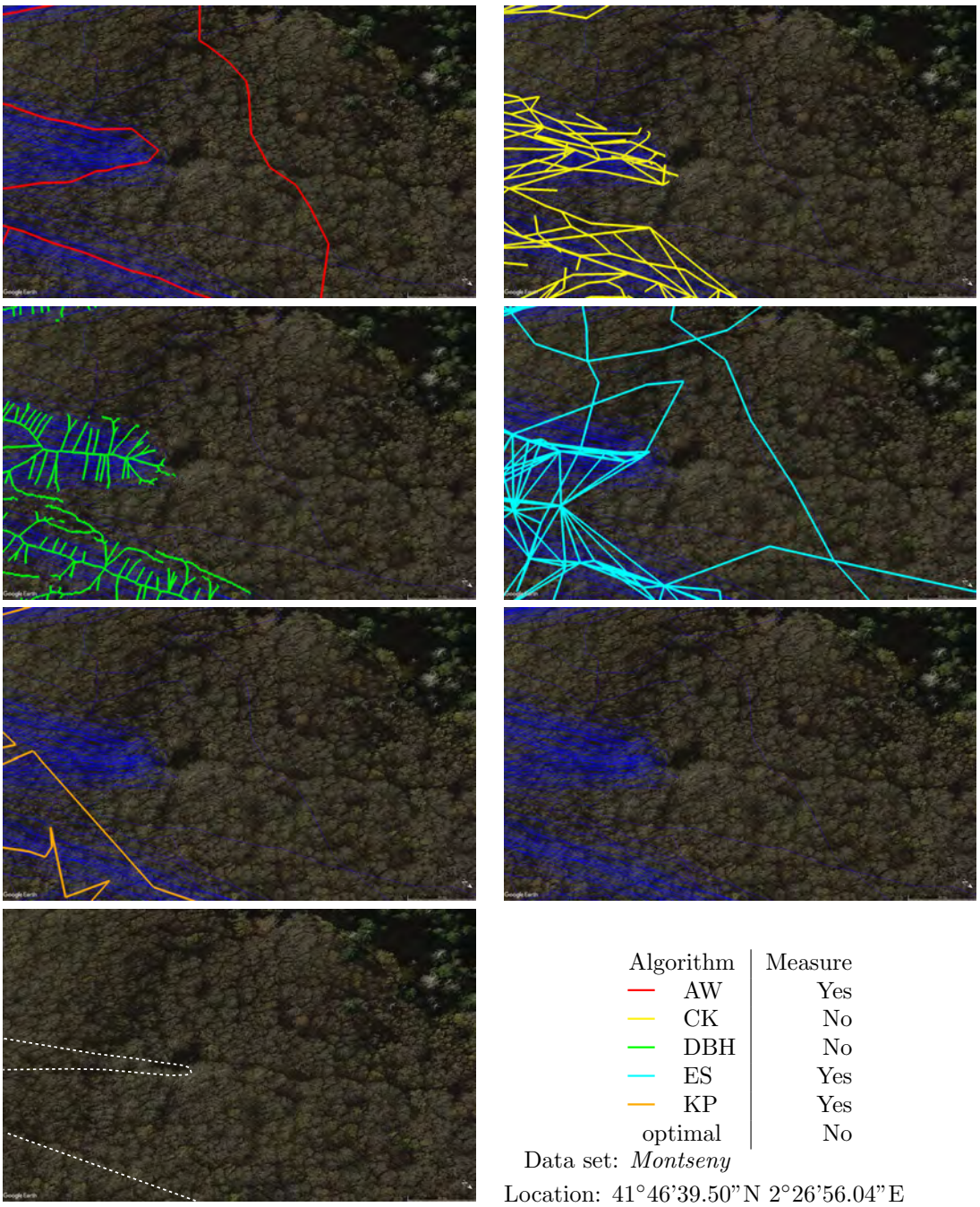
Location: 42°00'45.17"N 3°10'02.70"E

Causes: We start noticing that the measurement has been taken over the S-shaped path, without taking into account the shorter vertical path emanating from the main one, as only one trajectory takes it. The worst performances in this case are those of CK (due to the use of a global reliability threshold that does not work well in this area), DBH (for the same reason) and AW (due to data heterogeneity and inconsistency between time and space parameters of the algorithm).

Artifact [$C10$]: Nonexistent paths created

Description: Relatively long portions of single paths that do not exist in ground truth included in generated map.

Measure: Binary. "Yes" indicates that the algorithm produces the artifact. "No" indicates that it does not.



Algorithm	Measure
AW	Yes
CK	No
DBH	No
ES	Yes
KP	Yes
optimal	No

Data set: *Montseny*

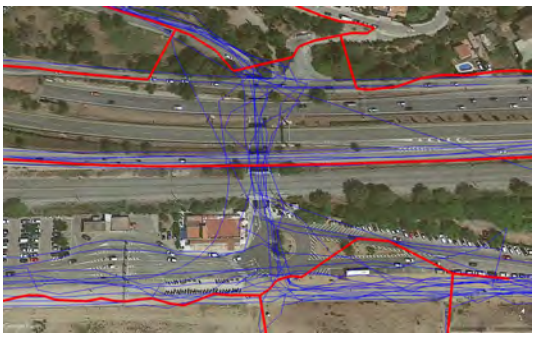
Location: 41°46'39.50"N 2°26'56.04"E

Causes: In this case, the issues show up in the result by AW, due to its incremental construction that selects an outlier trajectory, and ES and KP, due to their lack of a reliability measure.

5.3 Specific artifacts

In this section we present an analysis of the main artifacts that are specific to only one of the output maps of the algorithms. These are the result of some particular algorithm design decision. For each of these artifacts, we give a description, illustrate it for a particular location, and briefly explain its cause.

Artifact [$S1$]: AW



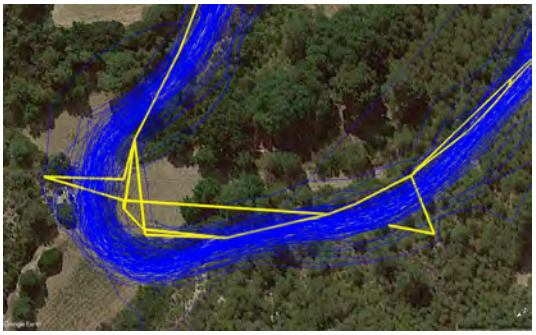
Description: Missed set of trajectories.

Cause: Lack of direction information associates the vertical trajectories to the horizontal ones. That is: t he algorithm considers the vertical trajectories close enough to existing (horizontal) paths in the map, thus they are ignored.

Data set: *Garraf*

Location: 41°15'46.25"N 1°55'48.80"E

Artifact [$S2$]: CK



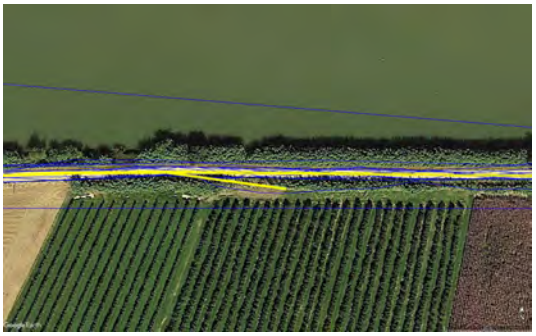
Description: Reduced curvature.

Cause: The clarification step of the algorithm averages sample positions of the neighborhood, reducing the resulting curvature.

Data set: *Garraf*

Location: 41°16'23.63" N 1°52'40.33" E

Artifact [$S3$]: CK



Description: Hair in off-track trajectories.

Cause: Combination of direction issues, use of hop distances, and attraction forces with noise.

Data set: *Aiguamolls*

Location: 42°01'22.41" N 3°10'38.17" E

Artifact [$S4$]: DBH

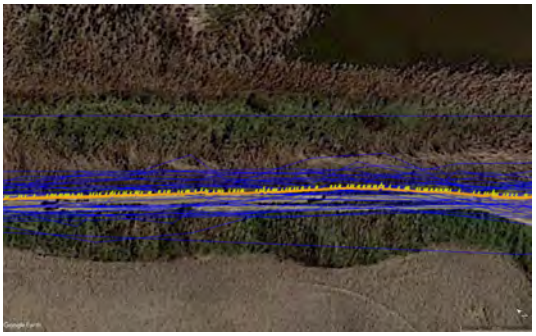
Description: Aliased generated map.

Cause: Discretization in a regular grid causes this artifact at larger scales.

Data set: *Delta*

Location: 41°17'37.13" N 2°06'23.13" E

Artifact [$S5$]: KP



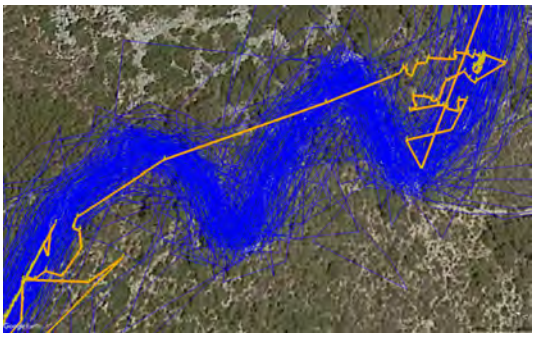
Description: Too many aligned vertices.

Cause: Heterogeneous data combined with an algorithm that is based on the detection of vertices and is prone to create too many of them.

Data set: *Delta*

Location: 41°18'16.06" N 2°07'28.89" E

Artifact [$S6$]: KP



Description: Winding path simplified.

Cause: Noise and zig-zagging path, causing an excessive number of intersections detected.

Data set: *Garraf*

Location: 41°15'51.30"N 1°53'29.22"E

Artifact [$S7$]: KP

Description: Missing trajectories ends.

Cause: The algorithm ignores any portion of the map not laying between path intersections.

Data set: *Garraf*

Location: 41°17'01.72" N 1°52'12.91" E

5.4 Analysis of the results

We begin by observing that some of the artifacts shown in Section 5.2 are mostly due to the characteristics of the data.

- Z-shaped and Y-shaped paths are difficult to deal with for all the algorithms. Some end up merging or shortcutting them: see artifacts [$C1$] for KP, ES and DBH, and, to a smaller extent, also CK, [$C2$] for AW and KP, and [$S6$] for KP. These issues are often due to the fact that in sharp turns with high noise, distance thresholds can be misleading, unless they are combined with angle information.
- Parallel paths combined with noise end up causing trouble to most of the algorithms: see artifacts [$C3$] for ES, CK and AW, and [$C4$] for KP, AW and ES. Again, noise and global distance thresholds make it very difficult to determine whether two parallel trajectories correspond to the same or different paths.
- Particularly noisy areas end up producing maps with an excessive number of vertices and connections, especially for ES and CK (see artifact [$C7$]), and too dense graphs for CK, DBH and ES (see artifact [$C8$]).
- The heterogeneity of the data, both in terms of time and distance gaps between consecutive trajectory points, gives rise to undesired results such as multiple map edges representing the same path (artifact [$C5$] for ES and CK), the absence of necessary connections (artifact [$C9$] for CK, DBH and AW) or too high complexity of the resulting map (artifact [$S5$] for KP).
- Finally, when trajectories go back and forth along one single path, which is not infrequent in our data, several algorithms have trouble in detecting the path as the same, particularly ES and AW (see artifact [$C6$]).

Some of the artifacts come as a consequence of design decisions of specific algorithmic strategies. Particularly:

- Direction issues. Giving a lot of weight to the difference in the direction of trajectories can be problematic under high noise. These issues are amplified in hiking trajectories, where direction is not as meaningful as it is for highway lanes or one-way streets. Indeed, pedestrians use the same path in both directions, no matter how narrow it may be. In contrast, we have seen situations in which ignoring direction leads to wrong maps. For the analyzed algorithms that don't make use of the directions at all, this has caused problems such as excessive simplification and merging in Z-shaped winding paths, like in DBH (see artifact [$C1$]), and missing paths, like in AW (see artifact [$S1$]). CK uses bidirectional edges, causing artificial dead-ends (artifact [$S3$]) and, again, excessive simplification and merging in Z-shaped winding paths (artifact [$C1$]).
- Reliability issues. Some algorithms use a density threshold to decide whether or not to include an edge in the final map. Artifact [$C9$] for CK and DBH show that this may lead to multiple connected components for a single path. The remaining algorithms don't use such a threshold, and this ends up producing erroneous paths or portions of paths, as shown in artifacts [$C7$] for AW, [$C10$] for ES and KP.
- Parameter incompatibility issues. In some cases, the parameters used lead to inconsistencies. In particular, the distance resolution between intersections is sometimes smaller than the GPS noise or inaccuracy in ES, hence producing a dense graph (artifact [$C8$]); and the segmentation threshold is sometimes smaller than the proximity threshold, hence producing disconnections in the map paths in AW (see artifact [$C9$]).
- Vertex detection issues. Two of the algorithms apply a strategy consisting in first finding significant vertices (i.e., intersections) of the map (clusters in ES, turns in KP), and then connecting them based on the trajectories that go through them. Either due to a bad identification of the vertices, or to a bad assignment of the trajectories to them, the fact is that this strategy often produces an excessive number of connections as ES does in artifact [$C8$], or an excessive number of vertices as KP does in artifact [$S5$]. In addition, it must be noticed that any strategy only based on detecting intersections necessarily misses endpoints that are not intersections, as it happens to KP in artifact [$S7$].

Figure 8: Examples of artifacts detected for the urban data sets (left-to-right): too many connections (artifact [$C8$], ES) and aliased map (artifact [$S4$], DBH) in *Chicago*; map missed sets of trajectories in *Athens Large* (artifact [$S1$], AW); many aligned vertices (yellow pins) in *Chicago* (artifact [$S5$], KP).

- Force results issues. Any strategy only based on attraction forces smoothens the curvature of the paths, as it happens for CK in artifact [$S2$].
- Discretization issues. Any strategy only based on discretization using a fixed-size grid—at least without some type of correction—generates aliased maps, as it happens for DBH in artifact [$S4$].

Finally, some issues are intrinsic to specific algorithms. Namely:

- **DBH** In general, maps produced by DBH do not cover as many paths as those of other algorithms. This phenomenon is typical of density-based methods, because the masking threshold must be high enough to omit outliers. This is much more noticeable at zones with high disparity of sampling between different paths, as more outliers are in the area with a high number of trajectories, pushing the threshold high enough to make zones with a low number of trajectories disappear. As mentioned before, the implementation available of DBH did not implement the shaving procedure as described in [12], thus we did not include *hairs* as one of the artifacts of DBH, since we cannot evaluate exactly how they would look with the original algorithm. However, given the nature of the algorithm, they are unavoidable, so this artifact will always appear in the resulting maps.
- **ES** The issues with excessive number of connections (artifact [$C8$]) are ubiquitously present for this algorithm, probably because the algorithm is very sensitive to GPS noise. This makes the quality of the generated maps unacceptable for most purposes.
- **KP** In several occasions we detected areas with an unusually high concentration of vertices and edges, as on the right side of artifact [$S6$]. In addition, we identified a large number of small loops. Since we were not able to explain these two phenomena based on the description of the original algorithm–they may be due to some bug in the available implementation—they have not been listed as artifacts.

It is worth mentioning that many of the aforementioned artifacts were already present in the results for urban data sets. However, there were not observed—at least not explicitly—probably because previous work focused more on the global aspects of the generated maps rather than on the details. In fact, in order to validate this we evaluated the different algorithms with the four urban data set used in [2], and analyzed which of the artifacts appear also in the urban data. The parameter values used for these runs can be found in Appendix B. The artifacts that were not found in any of the urban data sets analyzed, most notably, artifacts [$C9$], [$C1$], [$C4$], and [$S6$], are related to situations such as winding paths and long parallel paths close to each other, which do not occur in those data sets, and are less frequent in urban environments. Nevertheless, these are situations that can perfectly happen is urban settings as well. As expected, artifacts that are clearly intrinsic to the algorithms appear repeatedly in most urban data sets. Examples are artifacts [$C8$], [$S1$], [$S4$], and [$S5$], illustrated in Figure 8.

We conclude the section by observing that our experiments confirm that hiking data are appropriate to carry the desired local analysis of the performance of the algorithms, for they provide a wide variety of situations. Hiking trajectory data seems to have some specific characteristics that are relevant for map construction algorithms. The effects of GPS noise seems particularly important, probably due

to worse reception conditions (such as rough terrain and dense vegetation). The main effects are the bearings of parallel trajectories that are not well aligned, a high number of outliers, and many trajectories with noisy parts. Since our data came from very heterogeneous sources, trajectories varied a lot in sampling rate, something that caused difficulties to many of the algorithms. Finally, trajectories that go back and forth on the same path occur more often in hiking data, and can cause problems.

6 Lessons learned for designing new algorithms

6.1 Main challenges

Based on the previous analysis, we summarize here what we consider the main challenges to fully automatize construction of maps from trajectory data.

- Noise and outliers. How to tell apart the accurate and correct trajectories from those with a large error, even when the latter are only a minority, is probably the main challenge ahead. A large number of the issues identified can be traced back to inaccuracies in the trajectories. In fact, the combination of this with parallel and zig-zagging paths was partially responsible for a large fraction of the artifacts.
- Minimizing assumptions on the width of roads, paths, and their distances. Any algorithm meant to work even in only one specific setting should avoid, as much as possible, to make strong assumptions on these matters. Since this is hard to avoid, a possibility is to try to make only local assumptions, allowing for different parameters in different parts of the map.
- Minimizing assumptions on the sampling and other characteristics of the trajectories. As the amount and the variety of available data increases, algorithms need to be able to cope with rather heterogeneous inputs.
- Minimizing the number of parameters. Algorithm parameters are often related to assumptions on the input, which should be avoided, or at least stated explicitly. Moreover, parameters pose a major issue to the user of the algorithms, which may have to figure out over a dozen values by trial-and-error in order to get an acceptable output.

6.2 Algorithmic strategies to follow

Analyzing globally the different algorithms considered, we observe that certain decisions do not produce good results in our setting.

- Using turns to detect intersections, and detecting turns, in general, due to the presence of many small angles.
- Basing any decision on counting hops or time elapsed, due to the very different sampling used in the trajectories.
- Making decisions based on single vertices of a trajectory, something very sensitive to error (edges are probably more reliable in this respect).
- Incremental approaches, due to the many outliers and noisy trajectories. However, it should be pointed out that incremental approaches do avoid several types of artifacts because they stick to one single trajectory for longer parts of the map. This suggests that a good way to overcome the issues with current incremental methods may be to use some more robust way to select which trajectories to use.
- Giving a lot importance to the direction of trajectories, which have little relevance in pedestrian data, and even in road data when trajectories are very noisy.
- Relying on the detection of significant spots, which are particularly hard to detect in very noisy data, and can lead to issues close to the terrain boundary and around trajectory endpoints.

On the other hand, some algorithmic strategies seem appropriate and even necessary:

- Segmenting the input trajectories to make them more homogeneous, and to split them into meaningful segments.
- Avoiding giving too much weight to any single trajectory.
- Related to the previous, using the number of "similar" trajectories as a parameter to give more robustness to the results.

Thresholds. Our analysis confirms that the use of thresholds is very problematic. The analyzed algorithms use several kinds of thresholds:

1. For segmenting the trajectories. We believe that segmenting by distance is necessary. Segmentation by time is not appropriate in a context of heterogeneous data.
2. To filter for significance/relevance. The problem in this case is that any threshold induces some amount of fragmentation of the paths in the final map. We do not see a solution for this, other than hoping for a significant amount of data.
3. As a bound for similarity (mainly, proximity). Our experiments indicate that there exist no global thresholds that are appropriate for even a map of a few km^2. This is particularly evident when dealing with close parallel paths and narrow winding ones.

Thus it seems that fully automated map construction algorithms should rely only on local thresholds. It is also important to notice that the consistency between the different thresholds used in one single algorithm is also a key aspect in order to achieve good results.

7 Conclusions

In this paper we have analyzed locally five existing map construction algorithm in the context of hiking data. Our goal was not to compare the quality of the obtained maps, but to better understand the consequences of the strategic algorithmic decisions in each method at a local level.

Within this context, using hiking data had a double interest. The first one was to evaluate to what extent the currently existing algorithms, which were designed to be used in a different context, were flexible enough to also apply to a different one. The second, and more important, reason is that hiking data have proven to be a valuable benchmark in order to reveal which characteristics of the map construction algorithms should be improved or overcome.

To the best of our knowledge, this is the first time that local map construction algorithms issues are systematically classified and analyzed, searching for their causes and possible solutions. We have presented a detailed discussion of the artifacts that we believe can be used to understand some of the current challenges for map construction algorithms, and to design new and better methods. It is important to highlight that only a few of them apply exclusively to hiking data or to some specific algorithm. Indeed, many of the issues identified can be traced back to (combinations of) many aspects, like the fact that trajectories were created by pedestrians, often take place in rough terrain and in areas with vegetation and other sources of GPS signal degradation, and on the extremely diverse data sources used. Therefore many of the problems analyzed will show up in other settings in which map construction algorithms are needed.

An interesting byproduct of our work is a systematic discussion of the different parameter values that each method requires. For each algorithm it is unrealistic to find a universal set of parameters independent of the specific characteristics of each data set. Even more, our analysis shows that, even when restricted to one concrete data set, none of the algorithms gives good results for a single set of parameters values. In addition, finding the most appropriate parameter values turned out to be a challenging task, given the many parameters involved, and the lack of an explicit meaning for many of them. An important conclusion is that algorithms with such complex parameters are far from being usable, and that construction algorithms must strive to reduce the number of parameters and to make their meaning clear in terms of the input and output features. Ideally, such parameters should be locally adaptable to the data (an approach along this line has been very recently proposed [9], with promising results).

Finally, we believe that the rapid growth of available trajectory data will unavoidably require map construction algorithms to be able to cope with highly heterogeneous inputs. In order to succeed in this task, the best strategies will be those that assume as little as possible about the input, and are able to adapt to the data at a local level.

Acknowledgments

We are grateful to the anonymous reviewers for their many suggestions that have improved considerably the presentation of this paper. We also thank Kevin Buchin, Maike Buchin, Frank Staals, and Carola Wenk for useful discussions about the topics of this paper.

References

[1] M. Ahmed, B. T. Fasy, M. Gibson, and C. Wenk. Choosing thresholds for density-based map construction algorithms. In *Proc. 23rd Int. Conf. on Geographic Information Systems*, page 24. ACM, 2015.

[2] M. Ahmed, B. T. Fasy, K. S. Hickmann, and C. Wenk. A path-based distance for street map comparison. *ACM Trans. Spatial Algorithms Syst.*, 1(1):3:1–3:28, July 2015.

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

[4] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. *Map Construction Algorithms*. Springer, 2015.

[5] M. Ahmed and C. Wenk. Constructing street networks from GPS trajectories. In *Proc. ESA*, pages 60–71, 2012.

[6] H. Alt and L. J. Guibas. Chapter 3 - discrete geometric shapes: Matching, interpolation, and approximation*. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121 – 153. North-Holland, Amsterdam, 2000.

[7] J. Biagioni and J. Eriksson. Inferring road maps from GPS traces: Survey and comparative evaluation. In *In Transportation Research Board, 91st Annual*, pages 61–71, 2012.

[8] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proc. 20th Int. Conf. Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 79–88, New York, NY, USA, 2012. ACM.

[9] K. Buchin, M. Buchin, D. Duran, B. T. Fasy, R. Jacobs, V. Sacristán, R. I. Silveira, F. Staals, and C. Wenk. Clustering trajectories for map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017*, pages 14:1–14:10, 2017.

[10] L. Cao and J. Krumm. From GPS traces to a routable road map. In *Proc. 17th ACM SIGSPATIAL*, pages 3–12, 2009.

[11] C. Chen and Y. Cheng. Roads digital map generation with multi-track GPS data. In *Proc. Workshops on Education Technology and Training, and on Geoscience and Remote Sensing*, pages 508–511. IEEE, 2008.

[12] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

[13] T. K. Dey, J. Wang, and Y. Wang. Improved road network reconstruction using discrete morse theory. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, pages 58:1–58:4, New York, NY, USA, 2017. ACM.

[14] D. Duran, V. Sacristán, and R. I. Silveira. Map construction algorithms: an evaluation through hiking data. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2016, Burlingame, CA, USA, October 31, 2016*, pages 74–83, 2016.

[15] S. Edelkamp and S. Schrödl. *Route Planning and Map Inference with Global Positioning Traces*, pages 128–151. Springer, 2003.

[16] A. Fathi and J. Krumm. Detecting road intersections from GPS traces. In *Proc. 6th Int. Conf. on Geographic Information Systems*, pages 56–69, 2010.

[17] X. Ge, I. Safa, M. Belkin, and Y. Wang. Data skeletonization via reeb graphs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 837–845, 2011.

[18] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive GPS traces data. In *Proc. IEEE Int. Geoscience and Remote Sensing Symp.*, pages 667–670, 2007.

[19] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proc. 20th Int. Conf. on Advances in Geographic Information Systems*, pages 89–98, 2012.

[20] 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 International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 669–677, New York, NY, USA, 2012. ACM.

[21] R. Mariescu-Istodor and P. Fränti. Cellnet: Inferring road networks from GPS trajectories. *ACM Trans. Spatial Algorithms and Systems*, 4(3):8:1–8:22, 2018.

[22] B. Niehofer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In *Proc. 1st Int. Conf. Advances in Satellite and Space Communications*, SPACOMM '09, pages 156–161, Washington, DC, USA, 2009. IEEE Computer Society.

[23] D. Pfoser and C. Wenk. Map construction portal. `http://mapconstruction.org`, 2016. [Acc. 17/6/2016].

[24] M. Quddus, W. Ochieng, and R. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, pages 312–328, 2007.

[25] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive GPS vehicle trajectories. In *Proc. 12th Int. IEEE Conf. on Intelligent Transportation Systems*, pages 48–53, 2009.

[26] A. Steiner and A. Leonhardt. Map generation algorithm using low frequency vehicle position data. In *Proc. 90th Ann. Meeting of the Transportation Research Board*, pages 1–17, January 2011.

[27] TopoGraphics. GPX the GPS exchange format. `http://www.topografix.com/gpx.asp`, 2002. [Acc. 1/6/2015.

[28] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proc. 23rd Int. Conf. on Advances in Geographic Information Systems*, page 10 pages, 2015.

[29] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Bellevue, WA, USA, November 3-6, 2015*, pages 25:1–25:10, 2015.

[30] Y. Wang, X. Liu, H. Wei, G. Forman, and Y. Zhu. Crowdatlas: self-updating maps for cloud and personal use. In *The 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'13, Taipei, Taiwan, June 25-28, 2013*, pages 469–470, 2013.

[31] S. Worrall and E. Nebot. Automated process for generating digitised maps through GPS data compression. In *Proc. Australasian Conf. on Robotics and Automation*, 2007.

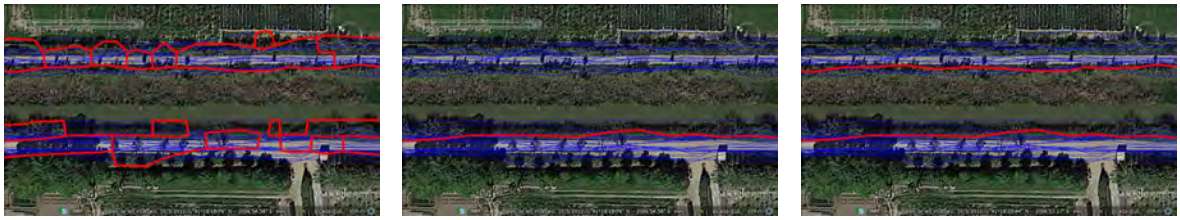
[32] J. Zheng, Y. Wang, and N. L. Nihan. Quantitative evaluation of GPS performance under forest canopies. In *Proc. IEEE Int. Conf. Networking, Sensing and Control*, pages 777–782, 2005.

Appendices

A Deduction of parameter values for hiking data sets

In this section, we carefully discuss how the parameters of the algorithms were set in order to apply them to the hiking data sets in the best possible way.

A.1 AW



(a) ε too small. (b) ε too high. (c) An appropriate value of ε.

Figure 9: Example of the effect of varying the value of ε for the AW algorithm. The image shows an area with parallel paths on both sides of a canal, in *Delta*. Input trajectories are shown in blue, the generated map is shown red. Background image from Google Earth.

Recall that the main parameter for AW is ε, for which four assumptions are made. However, these assumptions are not satisfied by any of our four hiking data sets. It is clear that condition (iv) is often not met in hiking trails (or in car trajectories, for that matter), since it is common to have hikes that repeat certain parts. However, it is relatively simple to preprocess trajectories to break cycles, as suggested in [5].

A more delicate situation arises with the remaining assumptions. Most notably, conditions (i) and (iii) can easily contradict each other. For instance, in *Delta* the presence of parallel paths along both sides of irrigation canals forces a value of $\varepsilon < 3m$ to satisfy (i). At the same time, the presence of some wider roads in conjunction with condition (iii) require $\varepsilon > 16m$, leading to a contradiction. Similar situations occur in the other three data sets. An example illustrating the effect of varying ε is shown in Figure 9.

Given that no single value can satisfy all theoretical conditions on ε, for our experiments we tried several values for each data set, selected based on the road widths and road separation distances observed in the Google Earth aerial images for each region. The algorithm was run on each setting and data set, and the value qualitatively giving the best results with respect to the paths visible from Google Earth was chosen. These values are shown in Table 6.

A.2 CK

Several parameters need to be adjusted to obtain good results for this algorithm. For the preprocessing, the thresholds used for segmenting based on the spatial or temporal discontinuities within trajectories (d_1 and t) are obtained by extrapolating the original values using the ratio between the mean distance (or elapsed time) between trajectory points of the original data sets and each of our data sets. The step for reducing redundancy uses two spatial thresholds (d_2 and d_3) and an angular one (α). The

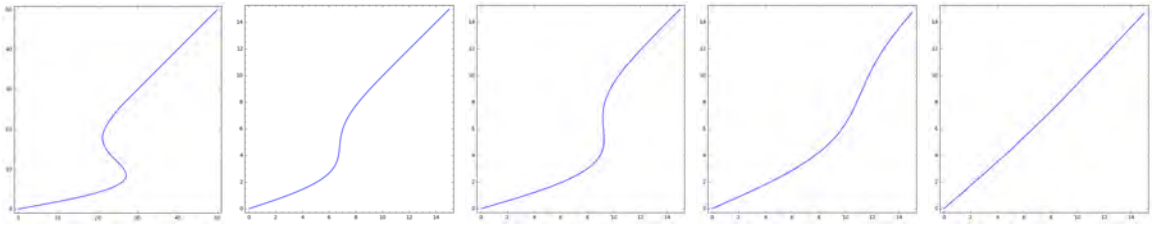
spatial thresholds were set to the original value multiplied by the ratio of the average speed between each of our data sets and the data set used in the original work. The angular threshold was kept unchanged.

The clarification step requires some information from the input data and the terrain. Recall that the attraction forces are parameterized by two values: M and σ_1, while the spring force depends on one parameter k. According to Cao and Krumm [10], the values should be chosen so that the force of one edge attracts all vertices with similar direction within a certain distance, and not those further apart. This implies that there should exist a distance value t_d at which the attraction force drops considerably, in favor of the conservative spring force. Such distance value should be between the maximum width of a one-way road and the minimum distance between two roads. Once the target distance value is found, Cao and Krumm [10] provide an analysis on how trajectories are affected by the forces taking into account the average number of trajectories on a path (N) and their dispersion due to the expected GPS error (σ_2). Using as input the values of N and σ_2, they derive the values of the force parameters so that the forces produce the desired change of behavior roughly after the target distance at which the attraction force must drop. In [10, Figure 8], this target distance seems to be implicitly set to 25m, and considering in average 20 trajectories per path with an standard deviation due to GPS error of 5m, the values of the three force parameters are set to $\sigma_1 = 5$, $k = 0.005$ and $M = 1$, producing the desired effect at 25m.

Therefore once the target distance t_d, together with the N and σ_2 are known, the three parameters M, σ_1 and k can be derived. Table 13 summarizes the values for each parameter as well as the needed information. Figure 10 presents the graphs of how the different attraction forces behave for the corresponding values.

Data set	**Terrain / Input data information**			**Clarification parameters**		
	N	σ_2 (m)	t_d (m)	M	σ_1	k
Original	20	5.0	25.0	1.0	5.0	0.005
Delta	20	2.7	6.5	1.0	0.0	0.2
Aiguamolls	5	1.7	9.5	1.0	3.0	0.02
Garraf	80	5.0	10.0	1.0	0.0	0.2
Montseny	30	7.3	7.5	1.0	0.0	0.2

Table 13: Values for each parameter used in the clarification step of CK using the original parameterization method. The table also includes the needed information about the terrain and input data, namely the average number of trajectories on a sampled path (N), the expected GPS error (σ_2) and the target distance (t_d) set half way between the maximum path width and the minimum separation between two different paths.



(a) Original (25 m) (b) Delta (6.5 m) (c) Aiguamolls (9.5 m) (d) Garraf (10 m) (e) Montseny (7.5 m)

Figure 10: Graphic representation of the resulting attraction forces of algorithm CK, for each data set, using their parameterization method. The axes represent the distance from a trajectory point to the center of the path that it possibly samples. The x-axis represents the original distance, whereas the y-axis represents the final distance after applying the clarification step. Ideally, the function should behave like a piece-wise function with $y = 0$ for $x \leq t_d$ and $y = x$ otherwise. The value between parenthesis is the corresponding t_d for each data set.

Even though the parameters for the clarification phase have been adjusted using the method proposed by the authors, there is still an issue to be addressed. The plots for the *Garraf* and *Montseny* data sets (Figure 10d-e) deviate too much from the ideal shape. Even more, the value of σ_1 in all data sets except *Aiguamolls* is 0 (Table 13), which results into a singularity in the attraction forces. In such cases, the resulting clarified trajectories no longer follow their original shape and the resulting map is indistinguishable from random noise.

Such situations occur when the expected GPS error (σ_2) is too close to the target distance t_d. In the *Garraf* and *Montseny* data sets, σ_2 was higher than 50% of t_d. In the *Delta* data set, it was higher than 41%. Whereas in the original and *Aiguamolls* data sets it was 20% and 18%, respectively. Therefore, whenever the expected error (σ_2) is close to the midpoint between the maximum width of a single path and the minimum distance between two paths, which incidentally is the value of t_d, the method proposed to adjust the parameters cannot be applied. Therefore, for the *Delta*, *Garraf* and *Montseny* data sets, the values have been empirically adjusted. Table 14 summarizes again the final values used.

Data set	M	σ_1	k
Original	1.0	5.0	0.005
Delta	1.0	2.7	0.05
Aiguamolls	1.0	3.0	0.02
Garraf	1.0	5.0	0.025
Montseny	1.0	7.3	0.025

Table 14: Final parameter values for the clarification phase of the CK algorithm.

Finally, the incremental insertion algorithm has four parameters that need to be adjusted. As the trajectories have been clarified, adapting the original values is straightforward. The distance threshold is set to be the maximum path width, as clarified trajectories are much closer to the center of the paths. The angular threshold, the minimum volume of trajectories and the maximum number of hops are all set to 45°, 3 and 5, respectively, as in the original work.

Refer to Table 7 for the summary of all the parameter values.

A.3 DBH

The (i) grid cell size that Davies et al. [12] propose is half the minimum path width. In our data sets, that is 1m for the flat terrains (*Delta*, *Aiguamolls*) and 0.5m for the hilly terrains (*Montseny*, *Garraf*). (ii) The value of σ was taken as the average between the maximum width of a path and half the minimum separation between two different paths. This value ensures that holes within a path will be covered while not interfering in the detection of different paths. Finally, (iii) the mask threshold was empirically adjusted. The value we present is a compromise between the coverage of the generated map and the algorithm's sensitivity to noise. Table 8 summarizes the values used.

A.4 ES

To choose the value of the three parameters for ES we used as a guideline the explanations in the original work [15], which we reproduce here for completeness. The value of $d_{\max}$ "should be in an order of magnitude such that we ensure not to miss any intersection". For δ, "we found that the algorithm is not very sensitive to variations of δ". Finally, "as a conservative lower bound, θ should be at least larger than the maximum lane width, [...], plus a considerable fraction of an estimated standard deviation of the GPS error".

For our data sets, a $d_{\max}$ of 20m is sufficient to detect all intersections. We kept δ at the same value as in the original work (45°). Finally, to set the value of θ we took into account the maximum width and the estimated GPS error of each data set. Table 9 summarizes the values taken for each data set.

A.5 KP

Finding appropriate values for the six parameters of KP, which do not have a clear meaning, was a complex task. Indeed, Karagiorgou and Pfoser mention that the values used in their experiments were obtained "empirically by running a great number of experiments and assessing the quality of the respective results" [19]. We established relationships between the parameters, based on their work, as to minimize the actual number of parameters to be empirically tested. As a result, we concluded that the most critical and independent parameters were (ii)—the angular threshold to determine turns,— and (iv)—the distance threshold to group turning points. We ignored the speed-related parameter (iii), as pedestrians do not perform significant speed reductions in turns. Parameter (vi) was fixed at 45°, the value used in [19]. The other two parameters were set as specified in Table 10 after experimental testing.

It remains to explain how to find appropriate values for (ii) angular difference and (iv) the turn clustering threshold.

The angular difference threshold determines when a change in direction is considered a turn. Ideally, the value should be set so that all (and only) real intersections have at least one turning point associated. As expected, in none of our data sets such an ideal value exists (note that it does not exist in the original *Athens* data set used in [19] either).

We found that the punctual angular differences on GPS data are not reliable enough to avoid false turn detections, both in urban and in hiking data sets. In our hiking data sets, the value of the angular difference threshold seems to be even more critical, since the density of trajectory nodes on the sampled paths is of one order of magnitude higher than in the urban context. Therefore, the probability that multiple falsely identified turn samples are considered by the algorithm to be intersections because they are close enough to each other is also much higher. This problem is specially apparent on *Garraf* and *Montseny*.

The second free parameter, the turn clustering distance threshold, has a less clear effect in the generated map, but it has a high impact in the final result. Essentially, the turn clustering threshold is used to decide when two detected turns represent the same turn in the ground truth. However, its implications extend further, because links between intersections created later on depend on the positions of the intersections, among other properties influenced by this parameter. Given all these implications, the final effect of varying this parameter is very hard to predict. Figure 11 shows an example.

(a) Turn clustering threshold of 5m.

(b) Turn clustering threshold of 100m.

Figure 11: Example showing the effect of varying the turn clustering threshold from 5m to 100m. Input trajectories are shown blue, the generated map in red. Red pinpoints indicate detected turning points, yellow pinpoints show the location of the intersection nodes. Images from Google Earth in *Aiguamolls*.

Based on all these observations, our method to obtain the values for the two parameters consisted in first finding a suitable value for the angular difference, and with this value fixed, looking for a suitable value for the turn clustering threshold, also empirically.

The values that gave the best results can be seen in Table 15. Note that the best values of the angular difference found for the hiking data sets (up to 70°) are much larger than the 15° used in the urban setting.

The need for such a larger angle bound can be explained by the trajectory sample points density in our data sets, when compared to the ones used by Karagiorgou and Pfoser. Assuming that the

Data set	(ii) Angular difference [º]	(iv) Turn clustering threshold [m]
Urban data sets[a]	15	50
Delta	50	10
Aiguamolls	30	25
Garraf	70	30
Montseny	70	30

[a]All city data sets, Athens, Berlin, and Chicago, used the same parameters.

Table 15: Chosen parameter values for KP. The values for the urban data sets are the ones in [3, 19].

identified turns are uniformly distributed, the Athens data set has an identified turn every 382m along a trajectory (45.11% of the input trajectories points are identified as turns). Using the same angular threshold, *Garraf* has an identified turn every 24m (37.76%). Although the percentages of identified turns are similar, the distance between two identified turns in *Garraf* is one order of magnitude smaller. Identifying turns that are too close makes identifying intersections using spatial clusters an even more challenging task.

The chosen angular threshold for our data sets have been selected taking into account the visual apparent density of the identified turns. Figure 12 compares the visual appearance of the identified turns between the original data set, the *Garraf* data set using the same angular threshold (15°) and *Garraf* using our chosen threshold (70°). Assuming that the identified turns are uniformly distributed, *Garraf* with a threshold of 70° has an identified turn every 316m along the trajectories (2.87% of the input trajectories points are identified as turns). Therefore, the density of the identified turns is comparable to the ones in the original data set.

(a) *Athens Large* data set with threshold of 15°.

(b) *Garraf* data set with threshold of 15°.

(c) *Garraf* data set with threshold of 70°.

Figure 12: Example showing how using the same angular threshold as in the original data set (15°) produces a saturated map due to the density of trajectory points. Using our chosen angular threshold (70°), the results for *Garraf* are comparable with the original results. The pinpoints in red are the identified turns, trajectories are in blue. The three images are at a similar scale. Images from Google Earth.

B Parameters used for the urban data sets

To run the algorithms for the urban data sets we tried to stick to the values mentioned in the cross-comparison paper by Ahmed et al. [3]. In most cases this was done, except for some few cases in which the parameters present in the code were different from those in [3], in which case we used those in the code.

AW: ε=180 (*Athens Large*), 90 (*Athens Small*), 170 (*Berlin*), 80 (*Chicago*); t_{gap}=120;

CK: d_1=100; t=10; d_2=10; d_3=30; α=10; min_seg=4; M=1; σ_1=5; k=0.005; d_4=20; β=45; v=3; h=5;

ES: $d_{\max}$=50; δ=45; θ=20; N=80; d_{med}=0.01;

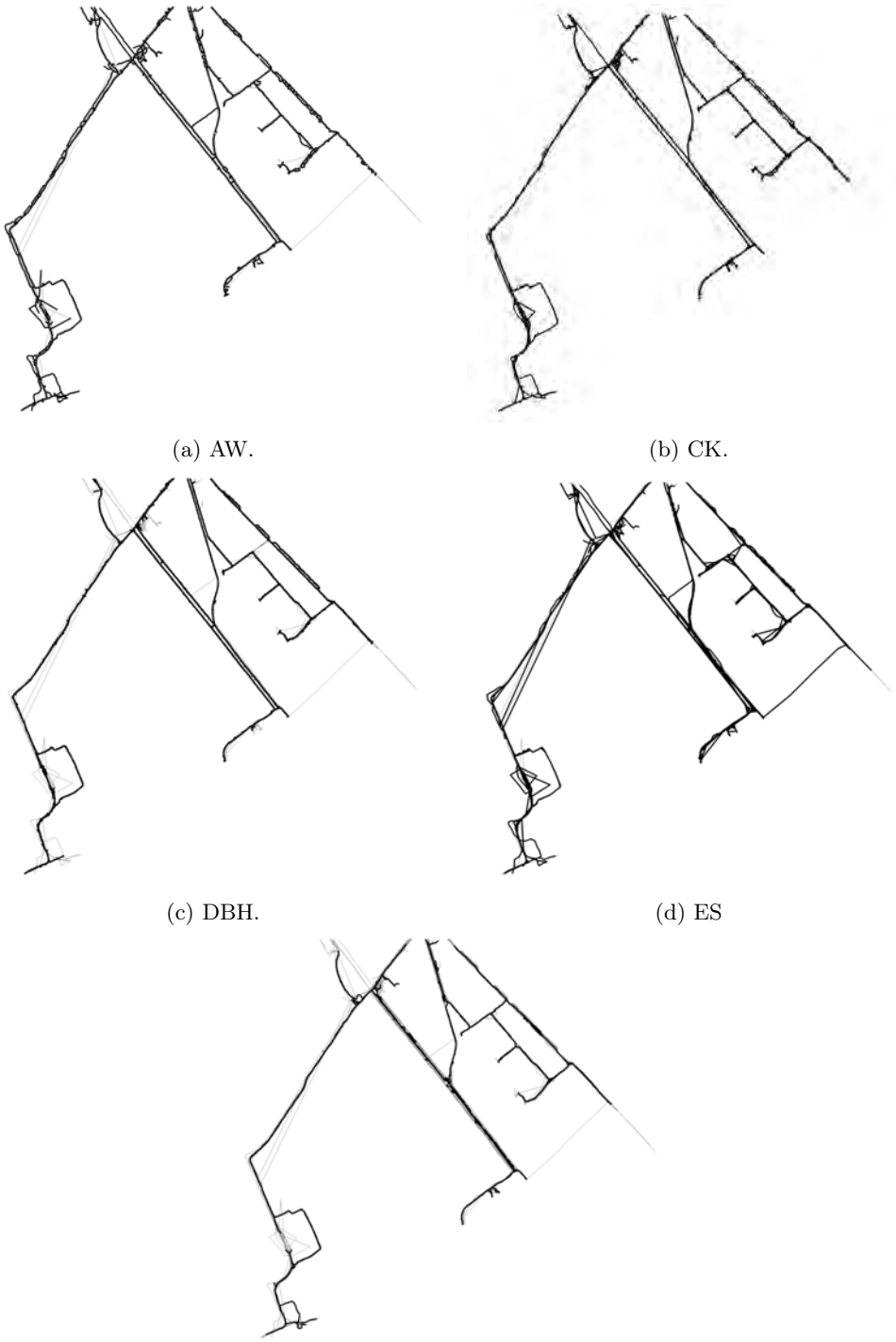
DBH: cell_size=2; mask_threshold=100; σ=17; voronoi_sampling_interval=10;

KP: angular difference=15; dist=25; max_m=1000; mean speed=40;

C Output generated by the different algorithms

In the next pages we present the maps generated for each data set by each of the five algorithms, together with the input trajectories in the background.

C.1 *Delta*



(a) AW.

(b) CK.

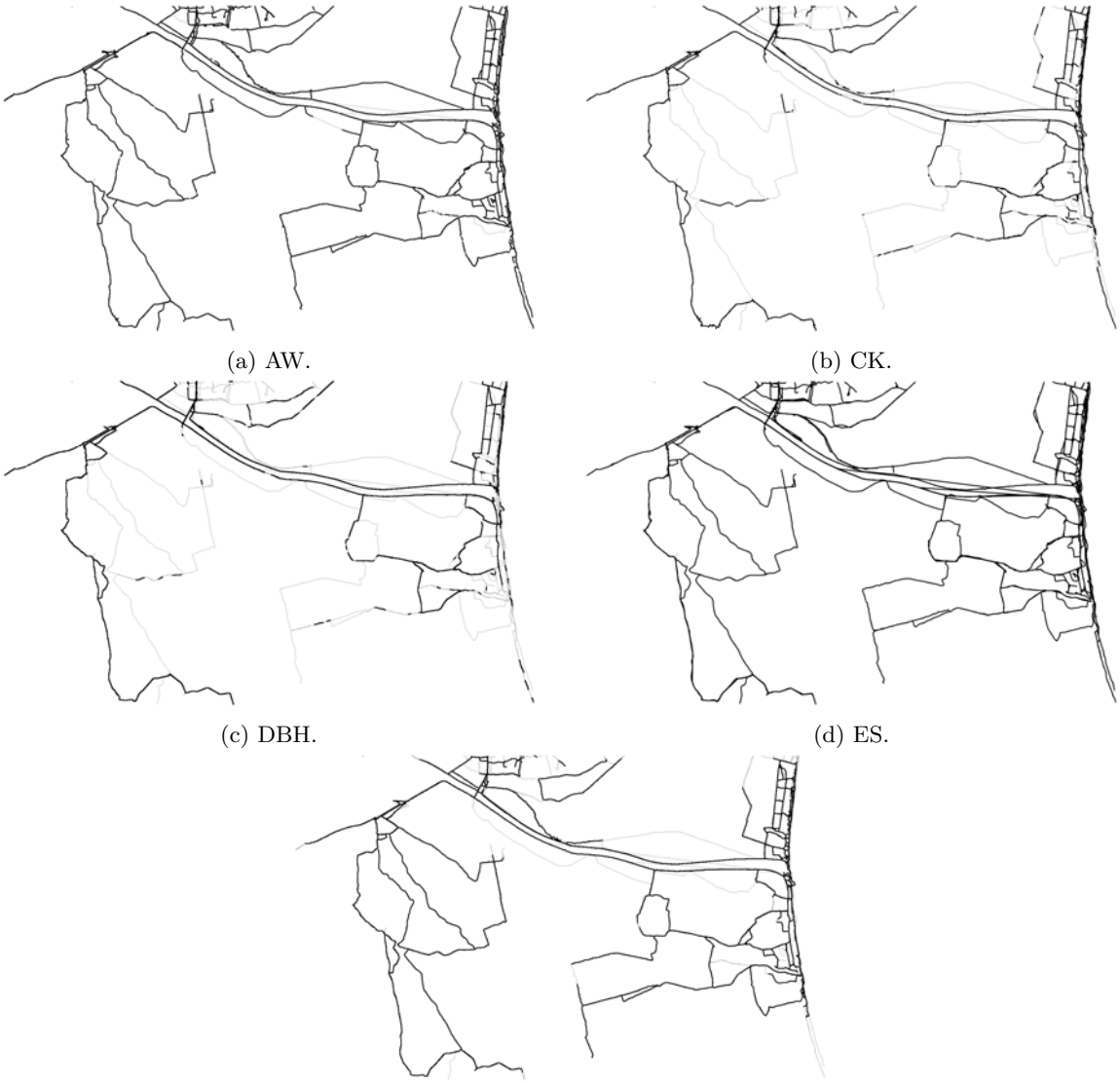
(c) DBH.

(d) ES

(e) KP.

Figure 13: Maps generated (in black) for *Delta*, with the input trajectories (in gray).

C.2 *Aiguamolls*



(a) AW.

(b) CK.

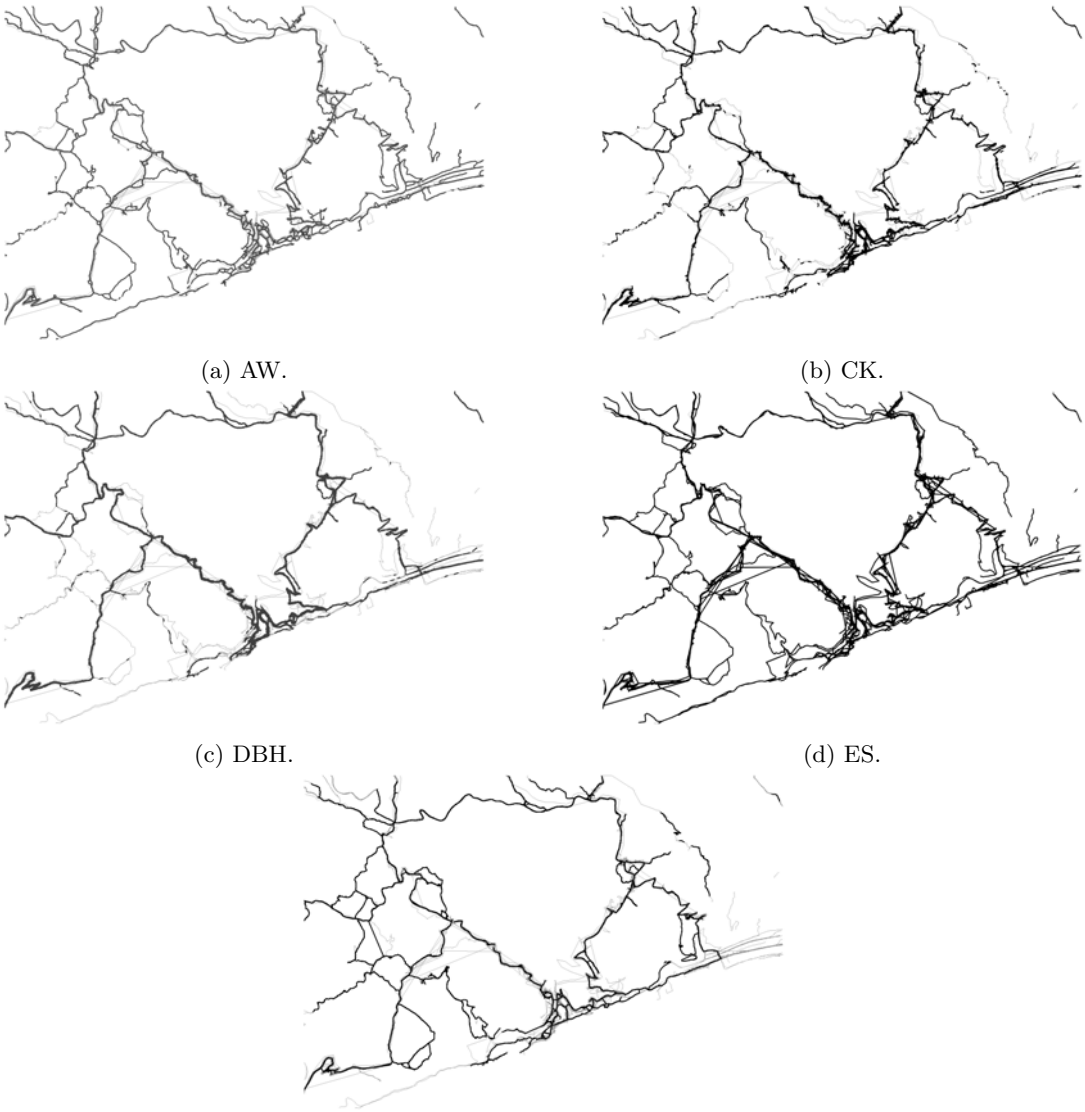
(c) DBH.

(d) ES.

(e) KP.

Figure 14: Maps generated (in black) for *Aiguamolls*, with the input trajectories (in gray).

C.3 *Garraf*



(a) AW.

(b) CK.

(c) DBH.

(d) ES.

(e) KP.

Figure 15: Maps generated (in black) for *Garraf*, with the input trajectories (in gray).

C.4 *Montseny*

(a) AW.

(b) CK.

(c) DBH.

(d) ES.

(e) KP.

Figure 16: Maps generated (in black) for *Montseny*, with the input trajectories (in gray).