Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Rapid and Brief Communication

Averaging GPS segments competition 2019

Pasi Fränti, Radu Mariescu-Istodor[*]

University of Eastern Finland, Finland

ARTICLE INFO

Article history:
Received 11 July 2019
Revised 24 August 2020
Accepted 29 October 2020
Available online 30 October 2020

Keywords:
Gps trajectories
Trajectory averaging
Time series
Representative segment

ABSTRACT

Averaging GPS trajectories is needed in applications such as automatic generation of road network and finding representative movement patterns. We organized a challenge where participants submitted proposals to solve the averaging problem. In this paper, we review the proposals and evaluate their performance. We present a synthesis of the submitted methods and develop a new baseline composed of the well-performing components. The new baseline outperforms all existing averaging methods. All datasets, submissions and evaluations can be accessed on the competition webpage: http://cs.uef.fi/sipu/segments.

1. Introduction

Tracking user movement by positioning technologies like GPS have became every day practice. As a result, large number of data has been collected. By analyzing the collected GPS trajectories, it has become possible to analyze traffic patterns [5], traffic anomaly detection [1–4], taxi social dynamics [6], and people movement under natural disaster [7] among many other applications. One important application is to refine existing road networks [40]. Researchers have even proposed methods to extract entire road networks from the collected GPS data [8–11,51,52]. This includes two major challenges:

- Detecting intersections
- Creating road segments

We focus here on the second challenge. Our goal is to find a fast and effective method for averaging a given set of trajectories so that the average segment can be used to represent the road segment. An example of GPS trajectories collected in the city of Joensuu is shown in Fig. 1 where two sample trajectory sets and their average segments are shown. Other applications for segment averaging include clustering of GPS trajectories, calculating average routes for taxi trips, and anomaly detections.

Finding a good representative segment for a set of GPS trajectories is a challenging problem. First, it is not even clear how the average should be defined in case of multivariate data. Second, defining the average as minimization problem leads to a computational infeasible NP complete optimization problem. Third, the data is often noisy and the exact mean might not always be the best solution.

The above-mentioned problems have led us to call for practical solutions to the problem. We organized the *segment averaging* competition where participants were given access to 100 sample trajectory sets to help in designing their methods. Participants were able to test their candidate solutions anytime via a web interface by dragging-and-dropping their candidate solution as a text file. The system then provided visual output with numerical accuracy estimates, see Fig. 2. There were no limitations of how many times a participant could upload a trial solution to the system. In total, we received 8282 trials.

Final solutions were submitted by the deadline, 15th April 2019, via the same web site by uploading a program code and a documentation of the proposed method. The program code was required to read and write the files in a specified format. Accepted programming languages were Matlab, Python, C/C++, Java and PHP. Sample codes were provided in each of these languages. In total, we received 9 submissions from 7 different participants.

In this paper, we summarize the results of the competition. We review the proposed methods; compare their results using both visual quality and several numerical criteria including average accuracy, processing time, number of points and the length of the average segment relative to those in the ground truth. We compare the methods against existing methods from literature; including several Medoid-based solutions, DTW-averaging, and the previous solution in our CellNet road extraction system [11].

Based on the proposals, we form their synthesis as a general framework. We then construct a new baseline method by selecting the best design choices for each component. By best, we mean

[*] Corresponding author.
E-mail address: radum@cs.uef.fi (R. Mariescu-Istodor).

https://doi.org/10.1016/j.patcog.2020.107730

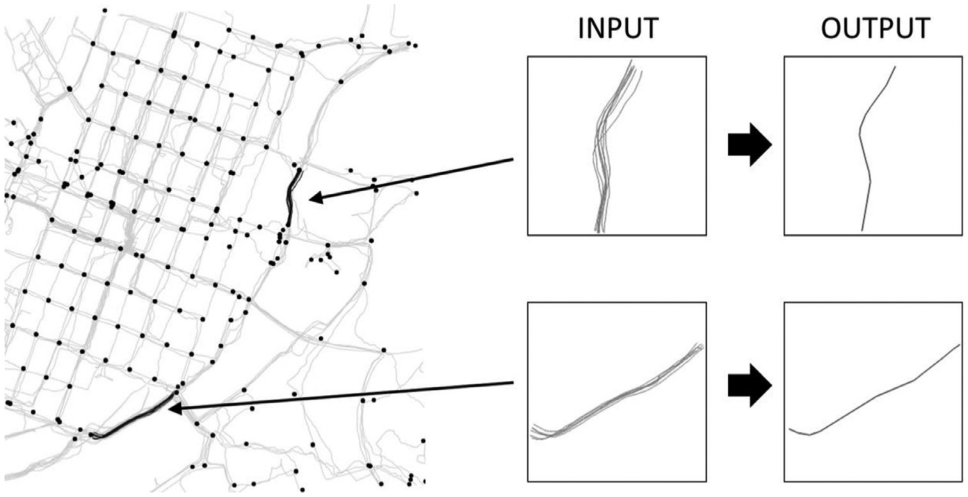


Fig. 1. A set of trajectories and known intersections (left). Two example sets of segments are extracted between two intersection-pairs (input). Expected output is also shown (right).

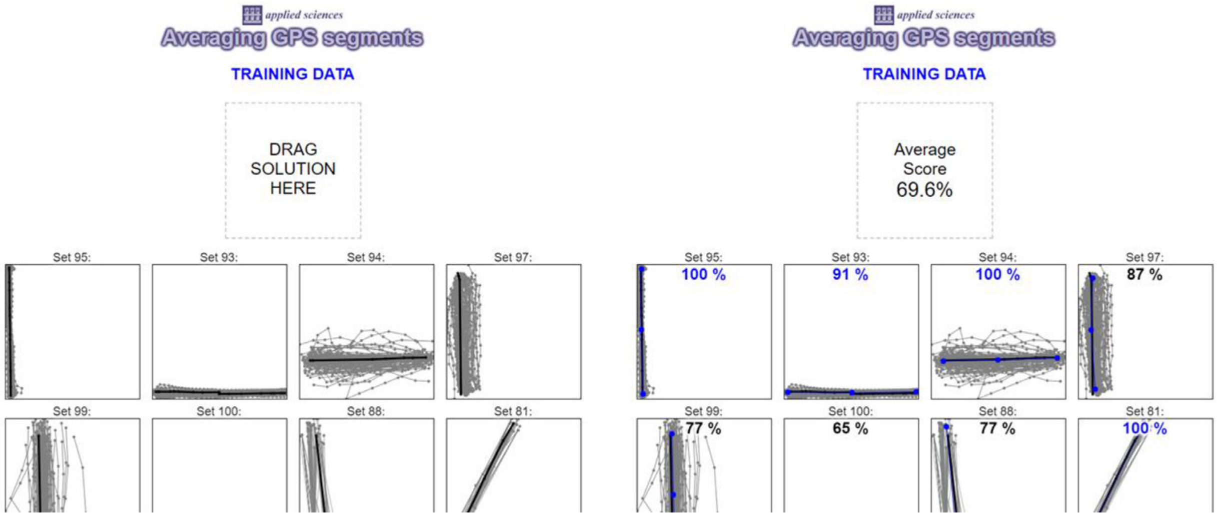


Fig. 2. Training data page (left). A participant could drag and drop a solution in the specified box to receive scores and visualization for every set in the training data (right).

choices that provide best compromise between accuracy, speed and simplicity of the implementation. Sometimes a more complex method can provide slight improvement but we do not consider it worth to add the extra complexity in the overall system.

To evaluate the methods, we introduce a new measure called *hierarchical cell similarity* (HC-SIM), which calculates similarity between the ground truth and the solution obtained by the algorithm. HC-SIM uses a grid with six different cell sizes (0.5%, 1%, 2%, 4%, 8%, and 16%). At each level, we count the number of cells the two segments share (intersection) divided by the total number of cells they occupy (union). This part is taken as such from C-SIM [12]. The final measure is the average C-SIM value at the six levels. The new HC-SIM measure correlates significantly better to human judgment (Pearson's correlation of 0.88) than the existing measures (Pearson's correlations between 0.05 - 0.72) when calculated on a dataset that was subjectively evaluated by two reviewers. For this reason, HC-SIM is used as our primary quality criterion.

The rest of the paper is organized as follows. In Section 2, we first review the existing averaging methods from literature, and then document all the submitted methods. In Section 3, we introduce the training and testing datasets used. We also define the evaluation criteria, including the new HC-SIM similarity measure. The results of the competition are then given in Section 4 in terms of numerical and qualitative evaluation. Conclusions are drawn in Section 5.

2. Averaging time series

The averaging task is seemingly simple as calculating average of numbers is trivial, and it is easy to generalize from scalar to vectors in multivariate data analysis. However, averaging *time series* is significantly more challenging as the sample points do not necessarily align. Given two time series sequences $X=\{x_1, x_2, \ldots, x_k\}$ and $Y=\{y_1, y_2, \ldots, y_k\}$, the sample x_i might align to some other sample than y_i but because of transformations in time. For example, the point x_7 is aligned with point y_5 in Fig. 3.

The problem has been studied using *dynamic time warping* (DTW) as the distance measure. Mean for pairwise average is easy to calculate [29] but it is not necessarily optimal [36]. The problem has been also considered as *multiple sequence alignment* (MSA), which is equivalent to the *Steiner sequence* [32,33]. However, it was shown that even this approach does not guarantee optimality [36]. The averaging problem in the context of DTW spaces has been shown to be NP hard with respect to the number of sequences [28].

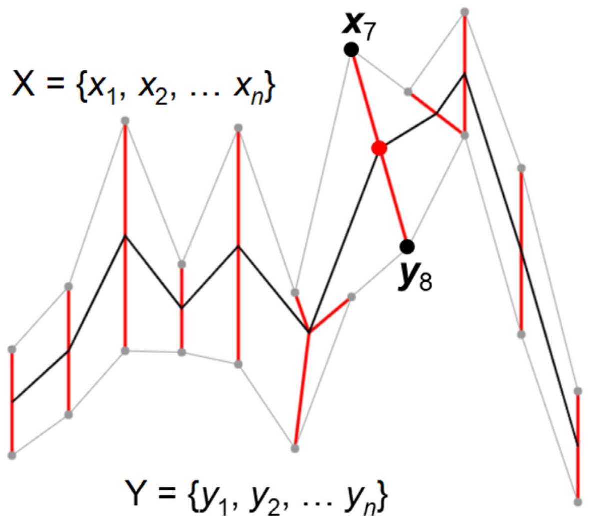

x_7
$X = \{x_1, x_2, \ldots x_n\}$
y_8
$Y = \{y_1, y_2, \ldots y_n\}$

Fig. 3. Averaging two time series X and Y by aligning the two sequences to the average (black line) using dynamic time warping distance.

Segment averaging is a special case of time series where the alignment is performed in geographical space. The spatial location acts as the observation, and the time stamp itself is usually ignored. Compared to time series, averaging a set of GPS trajectories poses two additional challenges:

- The samples are in two-dimensional space (latitude, longitude)
- The data can be very noisy due to GPS errors

We next review existing methods designed for time series, and outline how they have been generalized to averaging GPS trajectories. We recall one existing method [11], and document all the methods submitted to the competition.

2.1. Mean

Mean is defined here as *any* sequence that minimizes the total *squared* distance from all the input sequences to the mean:

$$\arg_C \min \sum_{i=1}^{k} D(x_i, C)^2 \quad (1)$$

Here $X = \{x_1, x_2, \ldots, x_k\}$ is the sample input set and C is their mean, which is also called as *centroid*. However, the search space is huge and exhaustive search by enumeration is not feasible. Some authors consider the problem so challenging that they think it is not self-evident that the mean would even exists [30,35]. However, their reduction theorem showed that mean exists for a set of DTW-spaces [30] and the existing heuristics have therefore theoretical justification. Exponential time algorithm for solving the exact DTW-mean was recently proposed in [36].

2.2. Medoid

Median is commonly used instead of mean because it is more robust to noise. Median is the observation which lies in the middle of the sorted observations. However, multivariate data lack natural ordering and different definition is therefore needed. *Medoid* is defined as the observation whose total distance to all other observations is minimal, see Fig. 4. By selecting a distance function between the trajectories, we can reduce the problem to linear search.

Medoid is defined as the input sequence x_j that minimizes the following function:

$$\arg_j \min \sum_{i=1}^{k} D\left(x_i, x_j\right) \quad (2)$$

Table 1
Heuristic algorithms to approximate averaging time series with DTW distance.

Acronym	Method	Reference
MM/DBA	Majorize-minimize / DTW Barycenter	[32] + [33]
Soft-DTW	Soft DTW averaging	[34]
SSG	Stochastic subgradient	[35]
iTEKA	Iterative time elastic kernelized averaging	[44]
CellNet	Shortest trajectory + MM (2 iterations)	[Cell Net 2018]

where D is any distance (or similarity) function between two GPS trajectories. For possible choices for the function, we refer to [12].

In other words, Medoid is defined as an optimization problem similarly to mean. The difference is that, instead of allowing any possible trajectory in the space, Medoid is restricted to be one of the input trajectories. This reduces the search space significantly, and the time complexity of finding Medoid is $O(nk)$, where k is the number of trajectories, and n is the number of points in the longest trajectory.

Medoid is a good choice when there are many trajectories in the set to choose from but it has several major problems. First, Medoid is limited to the original trajectories and can provide a poor approximation when there are only few observations to choose from. Second, it carries the properties of the dataset to the representative segment, such as the point frequency and artifacts like the zig zag in the leftmost example in Fig. 5. Third, artifacts of the chosen distance function also influence Medoid.

2.3. Approximating the mean

Several heuristic algorithms can be found in literature to approximate the mean when using dynamic time warping distance. We have listed the most common algorithms in Table 1. Most of them are iterative optimization techniques aimed at minimizing (1).

A two step procedure was proposed in [31] using Medoid as an initial (reference) solution. First, the input sequences are aligned to the reference solution using optimal warping path (*majorize step*). Second, pointwise averages are then calculated for the samples aligned to the same point (*minimization step*).

Based on the idea above, an iterative *Majorize-minimize* (MM) algorithm was introduced in [32] by repeating the two steps in turn. The algorithm is shown to converge to a local minimum after a finite number of iterations [35]. The Majorize-minimize algorithm was later re-formulated and popularized under the name *DTW Barycenter averaging* (DBA) [33], and its locally constrained variant presented in [53].

Soft DTW [34] replaces the hard minimum operator by soft expression so that the optimization function is fully differentiable in all of its arguments. *Stochastic subgradient* (SSG) [35] is very similar to MM but instead of calculating the exact subgradient using all samples, it estimates a subgradient of the Frechet function for a single randomly picked sample at a time, and updates the new average immediately.

We can draw an analog of SSG to MM as follows. MM correspond to the standard k-means (batch variant) [37,39] since all points are first assigned before updating the average. SSG corresponds to the sequential variant of k-means [38], since the average is updated immediately after every point assignment.

3. Averaging GPS trajectories

Averaging GPS trajectories is slightly different than averaging time series in general. Both time series and GPS trajectories are subsequent (ordered set) of observations. However, while time series contains numerical (scalar) observations, the points in GPS tra-

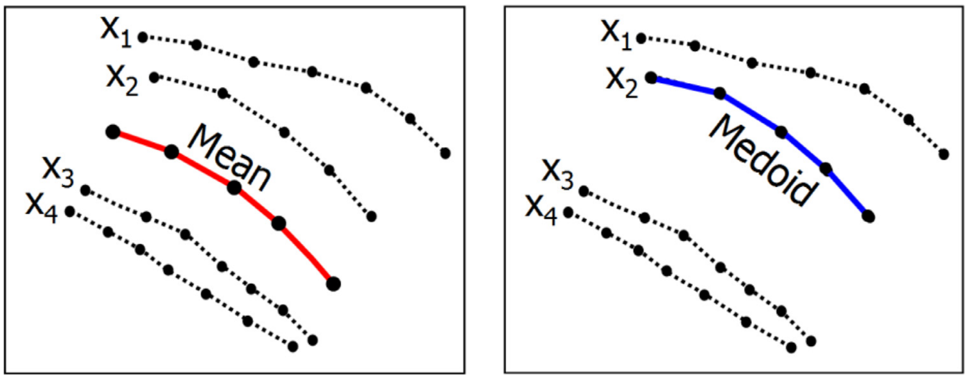


Fig. 4. Examples of using mean and Medoid for segment averaging.

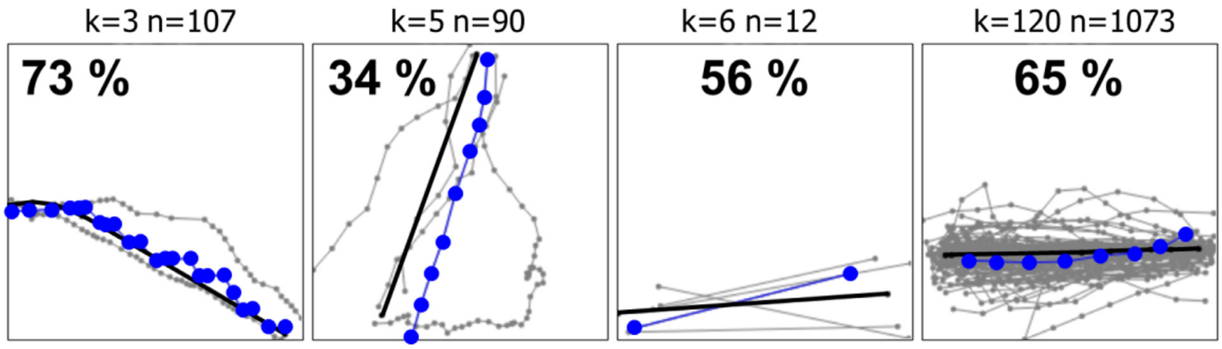


Fig. 5. Examples of using Medoid (DTW) for segment averaging of a set of k trajectories (gray) consisting of n points. The Medoid is emphasized using blue color. Ground truth is the black segment. HC-SIM similarity is shown as a percent.

jectories are two-dimensional vectors representing geo-locations. Most methods developed for time series should generalize to averaging of GPS trajectories with some effort, but there are some additional considerations.

First, the order of the points in the GPS trajectories can be arbitrary and we merely assume that the trajectories represent the same route. In other words, going from A-to-B is usually equivalent to going from B-to-A. Second, the data is likely to have both noisy points, and noisy trajectories. For this reason, and because of the high time complexity, we do not consider the calculation of the optimal average as a practical solution for the problem. Third, dynamic time warping is commonly used as the distance measure between time series but it is not necessarily the best function for measuring distance between GPS trajectories.

K-means clustering has been applied to extract road lanes from a set of GPS trajectories in [41]. The averages of the road segments were referred as *the center of the lane* but without any details how it was defined and calculated. In [40], the centerline is estimated using *spline fitting* with *weight least squared regression*. In other words, the data is treated as an unordered set of points for which piecewise polynomials are fit with continuity conditions at the knot points [42].

The number of knots (control points) was selected heuristically by selecting smallest number of points for which curvature error is below acceptable threshold. An explicit upper limit was also set as 2 N/k, where N is the total number of GPS points in the set, and k is the number of trajectories. This means that the center line can have at most twice the number of average points per trajectory. For example, assume that we have $k = 5$ trajectories each having $n = 20$ points, on average. Then, $N = 100$ and the upper limit will be 40 points.

Besides the method in [40], it is not easy to find alternative solutions from literature. The only other technique we are familiar with is the method used in CellNet [CellNet]. CellNet is a method extracting entire road network from a GPS collection. It first recognizes intersections by clustering the points around the regions where the trajectories split into several directions. The road segments are then created between the detected intersections using segment averaging, which is an important part of the method. The process of CellNet is illustrated in Fig. 6.

The *CellNet segment averaging* is based on the Majorize-minimize algorithm from [32] with few simplifications, see Fig. 7. First, instead of starting with Medoid, the shortest path is chosen as the initial choice as proposed in [Fathi and Krumm2010]. The assumption is that it contains less GPS errors and is more likely a direct connection of the two crossroads. It is also faster to compute than finding Medoid. The method uses the *Fast DTW* [23] instead of the original method with quadratic time complexity. These two modifications reduce the speed down to 1% compared to the original MM algorithm.

Outliers are also detected in CellNet by calculating similarity of the trajectory to the initial average. C-SIM similarity [12] is used with cell size of 25 $m \times 25$ m. Trajectories that are not 100% similar are removed. The CellNet algorithm then iterates the Majorize-minimize algorithm but only for two iterations since it tends to converge fast. Finally, the number of points in the average segment is reduced by polygonal approximation using the algorithm from [43]. As a result, the number of points is reduced to 30%, on average, without significant loss of accuracy. On the contrary, it was reported in [11] that it even improves the accuracy because filtering some noisy points as a side effect.

4. Summary of the new proposals

We received 9 submissions in total from 7 different participants. All methods were either completely new inventions, or

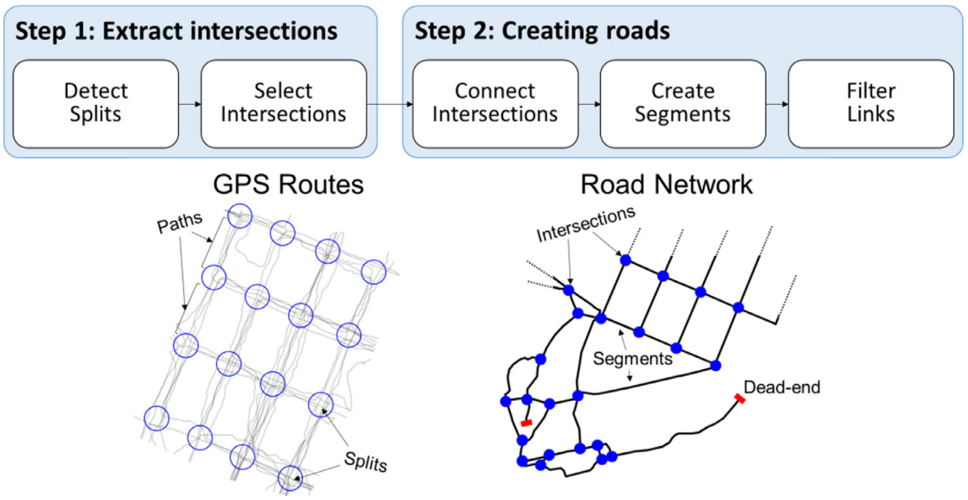


Fig. 6. System diagram of the CellNet (above) and sample result using GPS trajectories collected in the city of Joensuu, Finland (below).

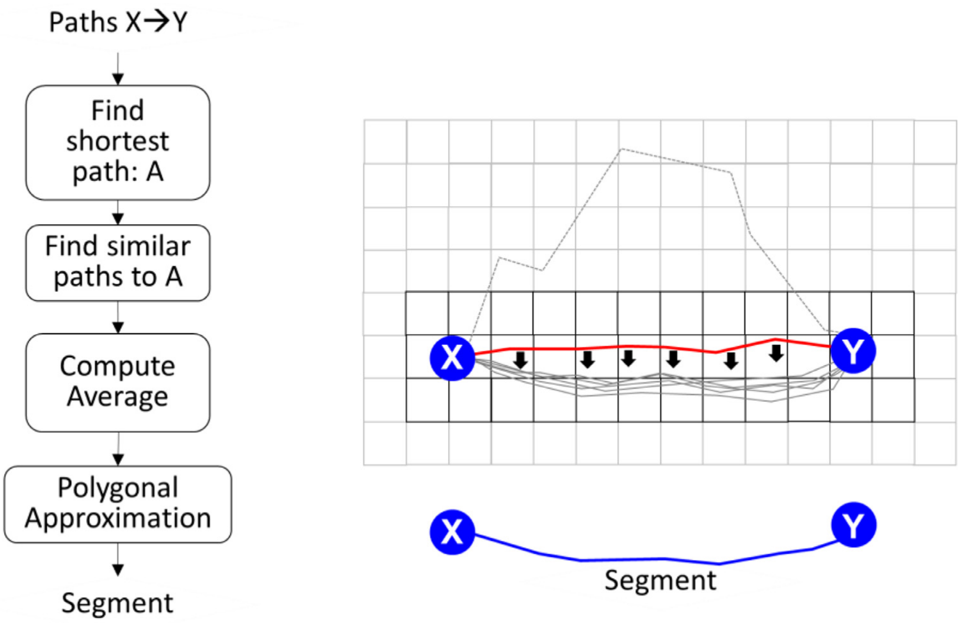


Fig. 7. The CellNet segment averaging (left), with an example (right).

adopted from the context of time series averaging. As far as we know, all participants worked independently and knew nothing about the work of others. Most participants had very little knowledge of the existing theory and literature. It is therefore fair to expect fresh ideas and different viewpoints to appear. However, to our surprise the overall structure and the individual design choices shared lots of similar ideas.

We next present the submitted methods one by one, and after that, we construct their synthesis as a general framework for averaging GPS trajectories. We then analyze how the proposed methods fit into this framework. We study every component and compare the alternative design choices for the components. The submitted methods are summarized in Table 2. Also two reference methods are included: Medoid and CellNet.

Table 2
Submitted methods and their properties.

Method	Author	Ref	Language	Parameters
Marteau	P.-F. Marteau	[Mar19b]	Python/C++	3
Marteau+PA	P.-F. Marteau	[Mar19b]	Python/C++	3
Yang-1	J. Yang	[Yang]	Python	1
Yang-2	J. Yang	[Yang]	Python	6
Leichter	A. Leichter	–	Python	1
Karasek-1	T. Karasek	[Karasek]	Python	3
Karasek-2	T. Karasek	[Karasek]	Python	3
Dupaquis	A. Dupaquis	–	Python	10
Amin	M. Amin	–	R	3
Medoid	–	–	Java	1
CellNet	–	[CellNet]	PHP	1

4.1. Marteau

The method by **Pierre-Francois Marteau** [Mar19b] consists of the following steps:

1. *Re*-ordering of the points
2. Oversampling by interpolation
3. Pointwise averaging
4. Removing outliers
5. Polygonal approximation (**Marteau+PA**)

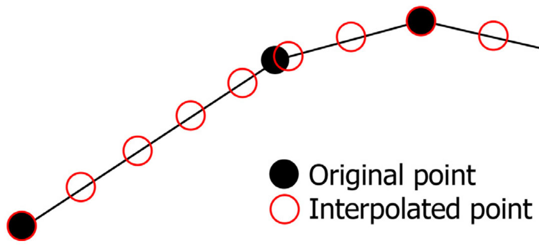


Fig. 8. Example of how the re-sampling procedure adds more points to the line by interpolation. Original trajectory has 4 points while the re-sampled trajectory has 10 points.

First, it solves the ordering problem as follows. It selects a reference trajectory randomly and calculates its similarity to all other trajectories, as such, and in reverse order. *Time elastic kernel* [26] is used as the similarity measure. If the reverse order is more similar, the trajectory is reversed. In the next step, extra samples are added to the trajectories by interpolation so that all trajectories will have the same length (see Fig. 8). The total number of points is fixed to a constant of 300 regardless of the trajectory.

Fig. 9, 10

The averaging itself is an iterative algorithm called *kernelized time elastic averaging* (iTEKA) [Mar19]. It is similar to the MM and DBA [32,33] but uses kernelized version of DTW, which considers all possible alignment paths, and it also integrates a forward and backward alignment principle jointly.

Outliers are detected by calculating the log-*similarity* of all trajectories in the set to the preliminary average. Trajectories are removed if their log-similarity is greater than the standard deviation of all log-similarities. However, this rule is applied only for sets whose standard deviation is greater than 5. After outlier removal, the averaging algorithm is repeated for the cleaned data.

Finally, the result is post-processed by polygonal approximation using the algorithm from [Mar09]. It requires the number of points as input, and then aims at minimizing root-mean squared error as the input trajectory and its simplified version. The average number of points of the input trajectories in the set is used as the parameter. Two variants are considered here: the algorithm without (**Marteau**) and with the post-processing (**Marteau+PA**). Only the first one was submitted to the competition. The latter was developed as an obvious consequence after seeing the results.

4.2. Karasek

The method by **Tomas Karasek** [Karasek] consists of the following steps:

- *Re*-ordering the points by clustering
- Find median number of points (k), and split the trajectories into 10*k intervals.
- Pointwise averaging
- Removing outliers
- Polygonal approximation (Douglas-Peucker)

Since some of the trajectories may be in reverse order, the first step aims at swapping their order so that all trajectories can be aligned. Ten iterations of k-means ($k = 2$) are applied for this. Extra samples are then added to the trajectories similarly as by **Marteau**. The number of points is taken as 10•k where k is the median number of points in the trajectories in the set. This causes some over-sampling effect but it will be taken care later.

Averaging is performed by calculating pointwise averages of each interval individually. This assumes that the trajectories are aligned well enough due to the over-sampling.

Outliers are detected by using two features: *distance* and *length*. The first feature is the distance between the trajectory and the preliminary average. This is done simply by summing up the Euclidean distances of all the aligned points. This is possible because of the over-sampling caused by the interpolation. The second feature is the length of the trajectory. Z-scores are calculated from both features and compared to the thresholds optimized for the training data: $T_{dist} = 2.13$ and $T_{length\text{-}low} = -1.23$, $T_{length\text{-}high} = 1.8129$ (**Karasek-1**), or $T_{dist} = 3.0154$ and $T_{length\text{-}low} = -1.5897$, $T_{length\text{-}high} = 1.9667$ (**Karasek-2**). At minimum, four closest samples are always preserved (if available) regardless of the outlier detection. The averaging step is then repeated after the outlier removal.

The final step is down-sampling by polygonal approximation. Douglas-Peucker splitting algorithm [24] is used with the parameter ε=0.0755 (**Karasek-1**), or ε=0.0058 (**Karasek-2**). Overall, the method is very similar to **Marteau**. The key elements are the over-sampling in the beginning to help the processing, and down-sampling at the end. For the averaging, **Marteau** uses a method adopted from time series context while **Karasek** uses a simple pointwise averaging. Both methods also apply outlier removal. **Karasek** considers the choice of the features for the outlier removal as the crucial part of the method. To avoid over-fitting, the parameters were tuned only using sample sets that visually approximated the true trajectory.

4.3. Leichter

The method by **Artem Leichter** has the following steps:

- *Re*-ordering the points
- Polygonal approximation (Douglas-Peucker)
- Pointwise averaging

First, it solves the ordering problem as follows. It selects the start point of the first trajectory as a reference and then calculates the distance from all the other start and end points. Trajectories whose end point is closer to the reference are reversed.

The method then applies polygonal approximation to reduce the number of points. The same Douglas-Peucker algorithm is used [24] as by **Karasek** with parameter setting ε=0.1. It assumes that the remaining points align because they are the representative points that keep the shape of the trajectory. The method then calculates the Pointwise averages at the same index. If the trajectories have different lengths, shifting is applied to balance the both ends. This can cause some alignment issue.

Overall, the method is similar to those of **Marteau** and **Karasek** in that the key idea is to modify the trajectories so that they will have the same number of points. The difference is that instead of adding extra points (oversampling), **Leichter** removes unnecessary points (down-sampling). The downside of the method is that the aligning can be rather rough, while its positive side is that polygonal approximation will not be needed anymore as a post-processing step. The method lacks outlier removal step.

4.4. Yang

The method by **Jiawei Yang** works as follows:

- *Re*-ordering the points by clustering
- Detecting segment type (linear or curvy)
- Create either linear segment or median segment
- Removing outliers (**Yang-2**)

It first solves the order of trajectories by clustering their start and end points by k-means ($k = 2$). The idea is the same as by **Karasek** except that k-means is iterated here until convergence. In addition to the start and end points, median point is also extracted from every trajectory, and the average of these median points is

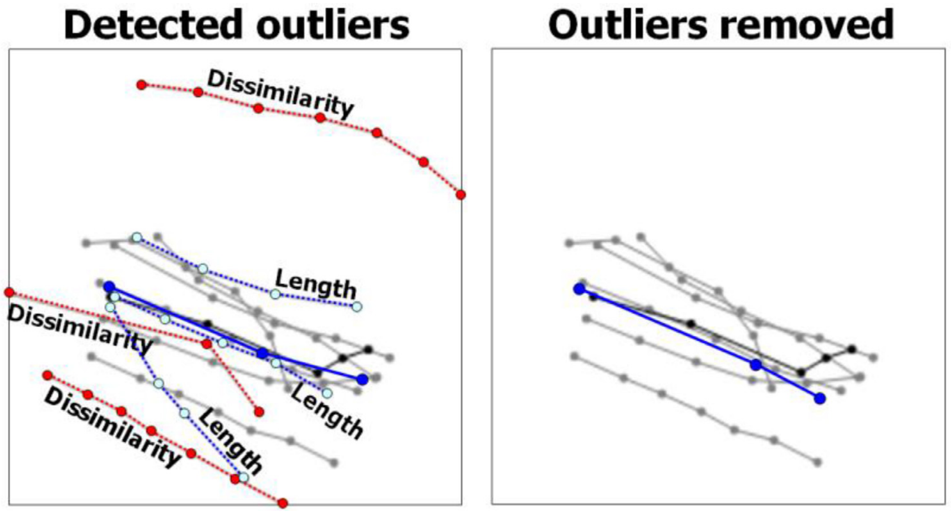


Fig. 9. Two types of outliers are detected: (1) trajectories having low similarity with the average; (2) trajectories whose length deviate too much from the average. The cleaned data and the revised average are shown on right.

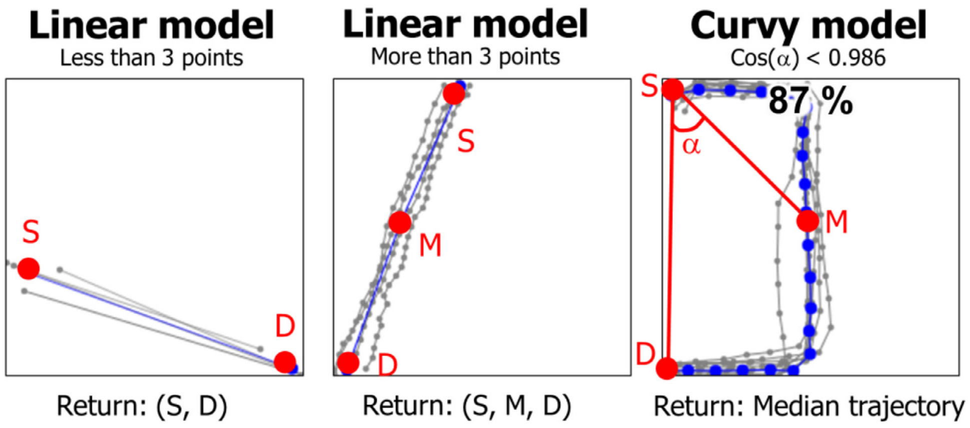


Fig. 10. Two strategies used by the methods **Yang-1** and **Yang-2**. Linear mode outputs only two or three sample point while the curve model finds the median trajectory.

also calculated. As a result, three descriptors are extracted from the set: start (S), median (M) and destination (D).

The method consists of two alternative strategies: one for linear segments and another one for curvy segments. The choice between these strategies is chosen for each set separately based on the three descriptors. If the cosine of the angle between the segments SM and SD (or MD and SD) is more than 0.99, then simple linear model is applied. If the set have less than three points, the output consists only of the end points (S,D). Otherwise, the output is the triple (S,M,D). The order of S and D is chosen arbitrarily.

The strategy for curvy segments is also rather simple. The number of the points in each trajectory is counted and their median is taken. The trajectory having the median number of points is then chosen as such as the output as such. This is kind of layman's version of Medoid, with the advantage of reducing the time complexity from O(k^2) to O(k).

A second version (**Yang-2**) was also submitted. This version includes also outlier detection step before calculating the descriptors (S, D, M). A general purposed density-based outlier detection method called *local outlier factor* (LOF) [Breunig et al. 2000] is applied. The outlier detection affects only on the calculations of those three descriptors. The revised descriptors are weighted linear average of the results with and without the outlier removal step.

4.5. Dupaquis

The method by **Alexandre Dupaquis** is very similar to **Yang**. It contains the following steps:

- *Re*-ordering the points
- Removing outliers
- Create either linear segment or apply naïve averaging

First, it solves the ordering problem by analyzing the slope between the start and end points. If the magnitude of the slope is greater than one (vertical direction), the topmost of the two points are chosen; otherwise the rightmost is chosen. In this way, the direction of the traversal is assumed to go from top-right to bottom-left.

Second, outlier trajectories are removed by considering the following four features:

- Slope
- Center
- Length
- Speed

If the value is above (or below) a threshold, relative to the average, the trajectory is removed; except it is the only trajectory remaining in the set. Thresholds are set to the following:

- Slope is twice as big (or twice small) as the average slope (and $k \geq 20$).
- Center is further than 3 times standard deviation of all the centers.
- Length is 30% longer (or 30% shorter) than the average length.
- Speed (average path length per point) is 50% faster (or 30% slower) than average.

The remaining trajectories are processed by calculating three descriptors similarly as **Yang**: end-points (S,D) and the middle point (M), which is the average of all points. The length of this segment (S,M,D) is calculated and compared to the average length of the remaining trajectories. If it is within 5%, then (S,M,D) is used as the representative.

In case of curvy segment, naïve pointwise averaging is performed without any alignment. The segment is continued as long as all valid trajectories have remaining points. Two variants are considered: one using normal (L_1) and another using quadratic (L_2) average. The one resulting longer segment is taken as the final result. This approach is somewhat too naïve and can cause unwanted artifacts if the number of points in the trajectories varies a lot.

4.6. Amin

The method by **Mohammad Amin** is probably the most unique among all the submissions. It involves the following steps:

- Selecting either the dimension (x or y) used as the predictor
- Selecting the start and end values for the predictor
- Removing outliers
- Solving piecewise linear regression

It is the only proposal that follows the approach presented in [40], which processes the trajectories merely as a bunch of points, and then solving piecewise linear regression [49,50]. This approach itself is somewhat counterintuitive as it throws out the information about the order of the points, and which trajectory they come from. However, it has the benefit of a smoothing effect as single noise point can no longer create sharp peak or turn into the trajectory because it will be averaged over a larger number of points.

The method **Amin** first selects the coordinate (x or y) that has larger variance, as the predictor in the linear regression. The minimum and maximum values of the predictor are then selected from every trajectory. Their corresponding averages are calculated and moved towards each other by the amount of standard deviation (SD); except if they are already closer than 2•SD (no movement made).

Outliers are detected using DBSCAN clustering algorithm [48] with the parameters eps=0.05 and $MinPts$=N/k, where k is the number of trajectories in the set. If more than 50% of the trajectories are detected as outliers, the DBSCAN is ran again after eps parameter has been increased by +0.01. This step is repeated until the number of detected outliers is less than 50%, after which all detected outliers are removed (see Fig. 11).

The method then calculates the average number of points in the trajectories and produces the average segment having equal number of points. Piecewise linear regression uses the method as explained in [49,50].

Since most original trajectories contain more points than needed, the method is likely to have over-sampling problem because it relies on properties of the dataset to choose the number of points. Polygonal approximation could be added as post-processing to overcome this problem though.

4.7. Synthesis of the methods

The submitted methods have a lot of in common. We next generate a synthesis of the proposals. It serves as a general framework how the averaging problem can be solved in general. Averaging methods for time series focus merely on the sequence averaging, while averaging GPS trajectories involves three additional steps:

- *Re*-ordering the points
- Outlier removal
- Simplifying by polygonal approximation

The synthesis is shown in Fig. 12, and the composite of the submitted methods are summarized in Table 3 accordingly. It is somewhat surprising how similar overall structure the methods have. The methods can be categorized to three overall strategies:

- Sequence averaging: Marteau, Yang, Karasek, Leichter, Dupaquis
- Simple line model: Yang, Dupaquis
- Piecewise regression: Amin

Most methods use the sequence averaging strategy but only **Marteau** adopted an existing optimization technique from time series context as such. **Yang** uses heuristic variant of Medoid while the other three modifies the sequences to have the same number of points and then apply pointwise averaging. This is done either by up-sampling via interpolation (**Karasek, Dupaquis**), or by down-sampling via polygonal approximation (**Leichter**). **Karasek** compensates the up-sampling strategy by a down-sampling post-processing step. **Yang** and **Dupaquis** alter between two different strategies.

Most methods apply pre-processing to guarantee that the points are in the same order before the averaging. Two methods (**Karasek** and **Yang**) cluster the start and end points, two other methods (**Marteau** and **Leichter**) compare the similarity of the original sequence and its reverse form to an arbitrarily chosen reference trajectory. The one with more similar decides the order. Two methods (**Dupaquis** and **Amin**) analyze principal direction either via the piecewise regression strategy (**Amin**) or by comparing the start and end points (**Dupaquis**).

Outlier removal is included in almost all methods. One author submitted two variants (**Yang**): with and without outlier removal. One method did not include any outlier removal at all (**Leichter**). The choice of the method varies a lot. Most methods (**Marteau, Karasek, Dupaquis**) use simple statistics like distance or length. One method (**Yang**) use existing general outlier detection method called LOF, and another (**Amin**) DBSCAN clustering algorithm for the purpose.

Regardless of the feature, selecting the threshold is a challenging step. Some optimize it for the training data, some use statistics like standard deviation, Z-score, or use log-scaling. Even the existing methods (LOF, DBSCAN) depend on the choice of the parameters. Most methods re-apply the sequence averaging step after the outliers have been removed. Two methods (**Karasek, Amin**) apply the outlier removal iteratively to relief the burden of the parameter choices.

Polygonal approximation is applied in three methods (**Marteau, Karasek, Leichter**). The first two need it because of the up-sampling process. The down-sampling approach (**Leichter**) is a bit crude but it avoids the need for this post-processing step. One more method (**Amin**) would have also benefitted from the polygonal approximation.

5. Data and criteria

We generated the benchmark data by extracting segments from four independent trajectory datasets:

- Joensuu 2015 [12]
- Chicago [14]
- Berlin [13]
- Athens [13]

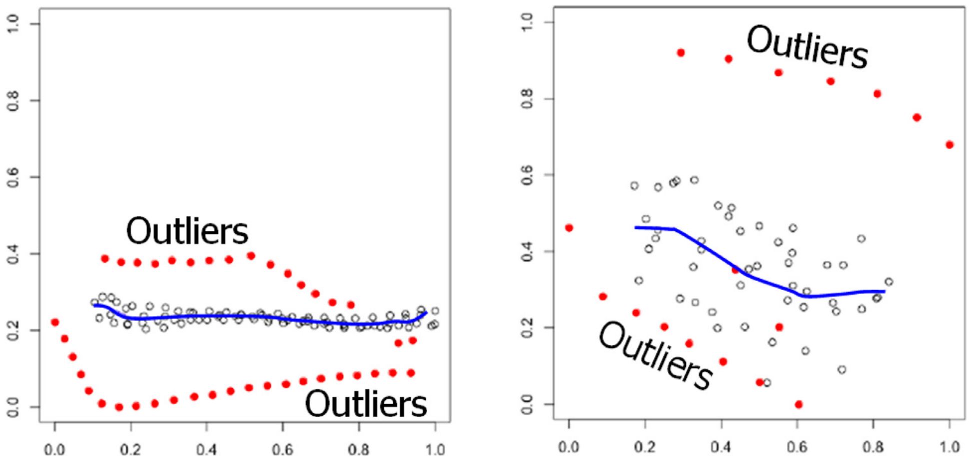


Fig. 11. Examples of the piecewise linear regression and the outlier detection used by **Amin.** On the left, outliers vary significantly from the well-defined path. On the right, bottom-most outliers overlap with considered points.

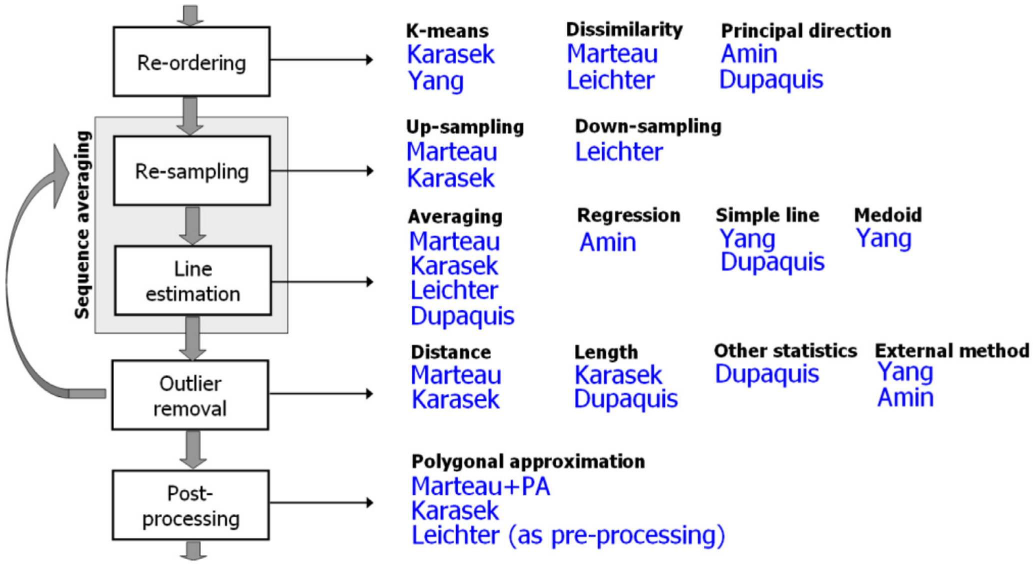


Fig. 12. Flow diagram of the method synthesis (left), and the chosen design components of the submitted methods (right). The core of the averaging step consists of only two steps, while the overall system uses one pre-processing and two post-processing techniques.

Table 3
Summary of the components included in the submitted proposals.

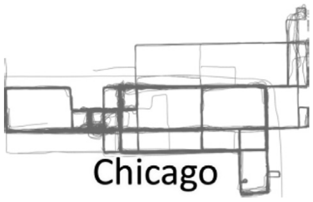
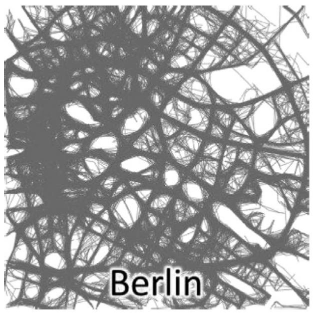
Method	*Re*-order	*Re*-sample	Averaging	Outliers	Polyg. appr.
Marteau	Dissimilar	Up to $n_{\max}$	iTEKA	distance	–
Marteau+PA	Dissimilar	Up to $n_{\max}$	iTEKA	distance	Marteau-Menier [27]
Yang-1	K-means	–	Line/median	–	–
Yang-2	K-means	–	Line/median	LOF	–
Leichter	Dissimilar	Downsample	Average	–	Douglas-Peucker
Karasek-1	K-means	Up to 10^*k	Average	dist+length	Douglas-Peucker
Karasek-2	K-means	Up to 10^*k	Average	dist+length	Douglas-Peucker
Dupaquis	Direction	–	Line/Aver	stats+length	–
Amin	Direction	–	Regression	DBSCAN	–
Medoid	–	–	Medoid	–	–
CellNet	–	–	MM	distance	Chen et. al

We selected 10% of the data for training, and the rest 90% was kept for testing. The data is available via our web site: http://cs.uef.fi/sipu/segments. The datasets and their properties are summarized in Table 4.

Joensuu 2015 dataset is collected by Mopsi users [12] mostly by walking, running and using bicycle. The point frequency is high, about 3 s on average but the data is often noisy due to low movement speed and using older phones for data collection.

Chicago data is collected by university shuttle busses of the University of Chicago. The segments are longer and simpler due to the regular city block structure, and the sets contain more samples. Some segments are systematically biased because of tall buildings along the route causing the GPS signal to bounce. Athens

Table 4
Five GPS trajectory datasets and their properties.

Dataset:	Trajectories	Points	Total length	Sampling rate	Speed
Joensuu 2015	109	43 891	252 km	2.99 s	10.58 km/h
	889	118 360	2 869 km	3.62 s	24.60 km/h
	27 189	192 223	40 603 km	41.9 s	23.8 km/h
	120	72 439	13 432 km	61.7 s	11.4 km/h

and Berlin contain much sparser data collected by car driving with recording frequency of about 42 s (Berlin) and 61 s (Athens). The segments extracted from these datasets are therefore simpler and contains only few points. The extraction process is described next.

5.1. Segment extraction

A GPS trajectory is a sequence of points $T = (p_1, p_2, p_N)$where each point has a latitude, longitude and timestamp. To extract the sets of similar segments we do the following steps:

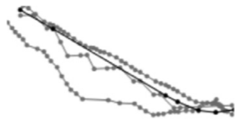
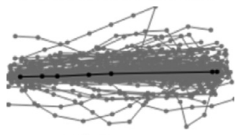
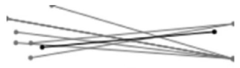
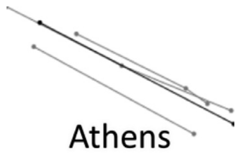
1. Query road network from OpenStreetMap
2. Extract intersections by selecting the nodes with degree more than 2
3. Extract in-between road segments with a minimum length $\varepsilon = 25$ m.
4. Using each road segment:
   a Search trajectories that pass through **both** its end-points ($\leq \varepsilon$)
   b Extract trajectory segments between the two end-points
   c Keep only the trajectory segments similar to the road segment
5. Discard sets that contain less than 2 trajectory segments
6. Convert points to UTM and normalize the values to [0, 1] interval for each set.

Step 4c is needed because there are trajectories that pass through the two end-points but follow a different path than the road segment, see Fig. 7. Using the similarity criterion, we can exclude these trajectories from our data set. For the similarity, we used the C-SIM measure with cell size $\varepsilon = 25$ m.

Step 6 is to allow participants to use the simple Euclidean distance. In this way, they can focus merely on solving the average instead of wasting time on secondary issues like data projection or implementing Haversine distance. It would also make possible to apply solutions from other spatial applications. Resulting trajectory sets are summarized in Table 5, and visualized for the Joensuu 2005 in Fig. 13. In total, we have 901 sets consisting of $k = 10{,}480$ trajectories with $N = 90{,}231$ points, in total.

Table 6

Table 5
Statistics of the segments extracted from each source dataset. A typical set is also shown from each source.

Set	Number of sets	Segments per set (aver.)	Points per segment (aver.)	Segment length (aver.)
 Joensuu 2015	227	4.16	22.84	1.01 km
 Chicago	227	84.63	4.61	0.76 km
 Berlin	625	6.97	2.08	0.98 km
 Athens	614	3.30	2.46	0.88 km

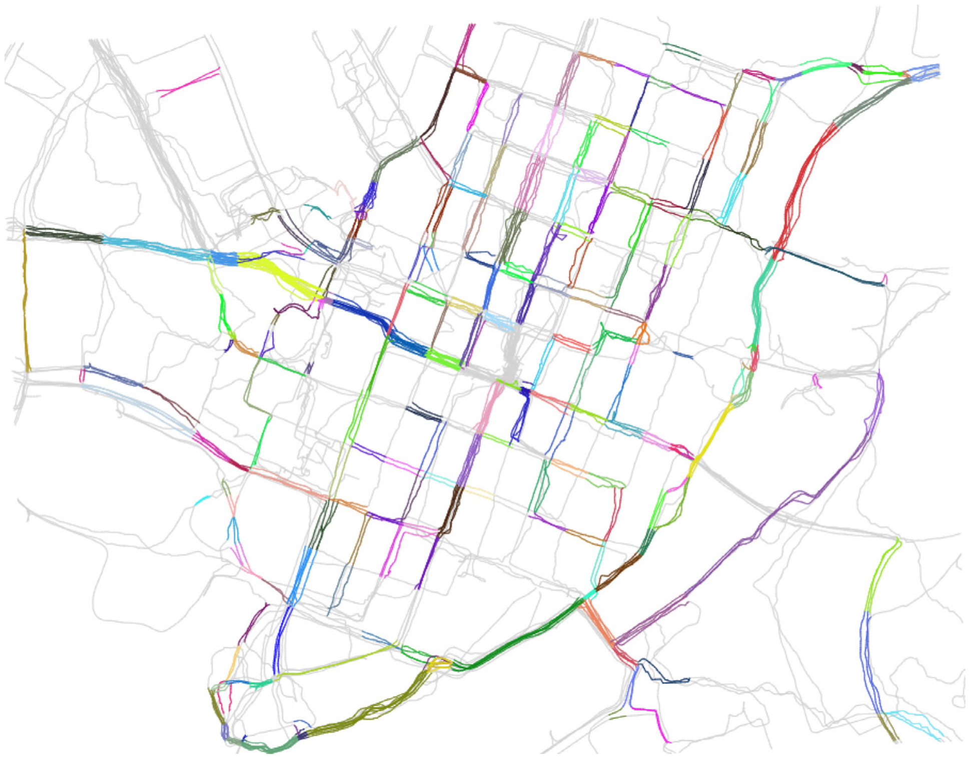


Fig. 13. Sets of similar segments extracted from Joensuu 2015 trajectory data. The sets are emphasized in different color.

5.2. Quality evaluation: hc-sim

Our primary evaluation criterion is a numerical similarity measure called *hierarchical cell-based similarity measure* (HC-SIM). It is based on existing C-SIM measure[1] [12] by extending it to multiple zoom levels. C-SIM divides the area into a grid of 25 × 25 m, and then counts the number of cells the two segments share (intersection) divided by the total number of cells they occupy (union). HC-SIM extends this to six levels with a quad-tree structure: 25 × 25 m, 50 × 50 m, 100 × 100 m, 200 × 200 m, 400 × 400 m 800 × 800 m. In case of our normalized [0, 1] scale, we use the sizes of: 0.5%, 1%, 2%, 4%, 8%, 16%. The final HC-SIM measure takes the average C-SIM value at these six levels:

$$\mathrm{HC-SIM}(A,B)=\frac{1}{L}\sum_{i=1}^{L}\mathrm{C-SIM}\left(A,B,0.005\times 2^{i-1}\right)$$

[1] http://cs.uef.fi/mopsi/routes/grid.

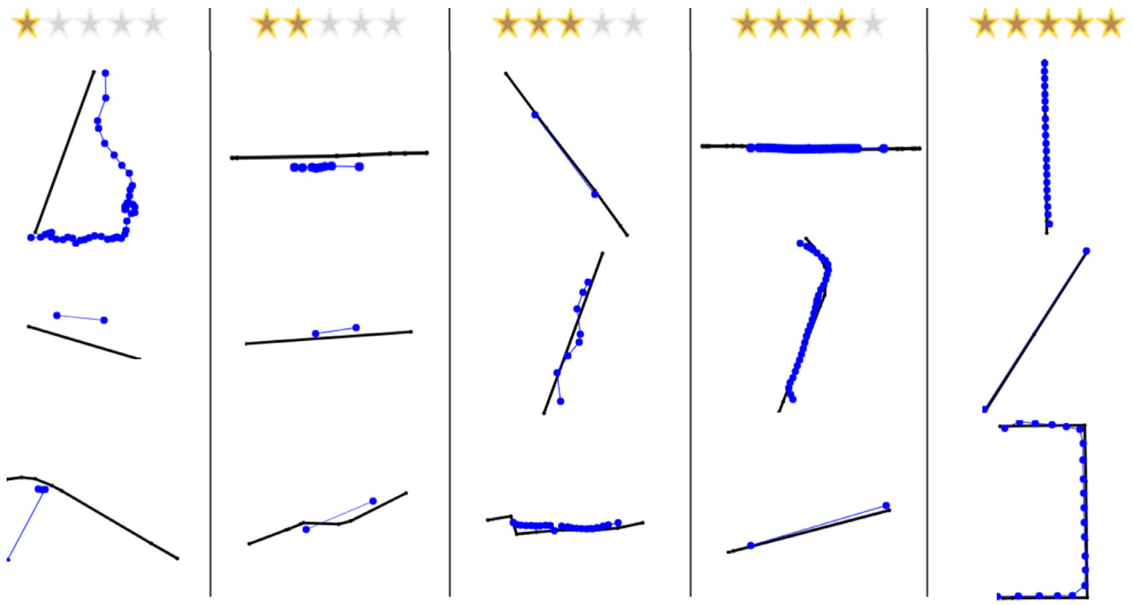


Fig. 14. Examples grades given by the human evaluators with 15 sample segments from the Joensuu 2015 trajectory dataset.

Table 6
Correlation of different trajectory similarity measures to human grades.

Measure	Reference	Correlation
HC-SIM	new	0.84
C-SIM	[12]	0.72
IRD	[15]	0.52
LCSS	[16]	0.45
EDR	[18]	0.37
Hausdorff	[22]	0.32
ERP	[20]	0.21
Dynamic time warping	[17]	0.11
Euclidean	[21]	0.09
Discrete Frechet	[19]	0.05

The new measure was not known by the participants so they could not optimize their methods directly to the similarity measure. One of the main benefits of the proposed HC-SIM is that it does not require any threshold parameter. Instead, we call the C-SIM measure repeatedly starting from smallest cell size of 0.5% and then doubling it at every repeat. We stop the extension at level $L = 6$ because larger values do not yield any further difference in the measure because the cell sizes become so large that all segments will become equal.

To validate the measure, we created a small evaluation set consisting of pairs of segments. Each pair consists of the ground truth segment and the results generated by three distinct methods: Medoid with DTW, CellNet and a random choice. Two humans graded the results from 0 to 5 as demonstrated in Fig. 14. The grades of the two humans correlated to each other with a factor of 0.77 (Pearson's correlation test).

The new HC-SIM measure correlates very well (0.84) to the average human grades, and is therefore used as our numerical quality criterion. We considered also nine other well known similarity measures. Their correlation ratios in Fig. 6 show that most measures perform poorly. IRD performs somewhat better because it uses interpolation. However, most others are sensitive to the number of points, sampling frequency, or the direction of travel. The result of HC-SIM equals to C-SIM if selected $L = 1$. The correlation then steadily increases as L increases from 1 to 6, after which no more changes occur.

Frechet and Hausdorff measures are similar to each other. Frechet utilizes the direction of the movement while Hausdorff does not, and therefore, it works better in our task. However, they both are sensitive to noisy points as they measure maximal differences instead of average. LCSS and EDR perform reasonably well but they are sensitive to the choice of the threshold value. The only other measure that performs even close to HC-SIM, is its single-level variant, C-SIM.

5.3. Other evaluation criteria

For evaluation, we use the following additional criteria:

- Visual quality
- Length of the segments relative to the ground truth
- Number of points relative to the ground truth
- Speed of the algorithm
- Simplicity of the implementation

Accuracy and visual quality are the main criteria, and speed the secondary. The other criteria provide additional insight, but they were not decisive in the ranking.

Visual quality was also evaluated by the organizers to detect obvious flaws in the numerical measure. This caused the highest scoring method to be disqualified as it seriously over-sampled the segments. HC-SIM did not penalize over-sampling because it is invariant to the number of points. While having slightly higher number of points than in the ground-truth is not necessarily a bad thing, excessive number of points can increase memory and storage requirements, and also slow down data retrieval and processing times in case of large road networks.

In terms of speed, the submitted methods were divided roughly into two categories: fast and slow. The fast methods took only few seconds for the entire test data while the time taken by the slow methods varied from 10 min to 1 hour, approximately. We could have used the speed as a tie-breaker but since the winning method was already among the fast ones, we did not need to.

The results of the other criteria are also reported but they did not affect the rankings. Some methods had moderate over-sampling issue, or deviation from the expected length, but such issues did not seriously disturb the visual quality.

All submitted methods were run on a test dataset (901 sets in total). The speed was measured by running the methods on the

Table 7
Results of the Medoid with different similarity measures.

Method	Training	Testing	Diff.	Length	Points	Time
DTW	57%	55%	2%	97%	159%	~1 h
ERP	58%	56%	2%	97%	182%	~1 h
Euclidean	58%	55%	4%	97%	175%	~1 h
Frechet	60%	55%	5%	97%	210%	~1 h
Hausdorff	61%	56%	5%	99%	121%	~1 h
IRD	62%	57%	5%	98%	169%	~1 h
LCSS*	59%	54%	5%	99%	202%	~1 h
EDR*	59%	55%	4%	99%	210%	~1 h
C-SIM*	58%	55%	4%	98%	190%	~1 h
HC-SIM	59%	55%	4%	98%	1.9%	~1 h

* Threshold value of 0.05 (eps for LCSS and EDR, cell length for C-SIM).

same Dell R920 machine with 4 x E7–4860 (total 48 cores), 1 TB, 4 TB SAS HD.

6. Results

We next report the results as follows. In Section 5.1, we first study the Medoid with different distance measures, and select the best one to represent the Medoid in future comparisons. In Section 5.2, we provide comparison of the submitted methods with two reference methods: Medoid and CellNet. In the following subsections we then study different compositions of the best individual components.

6.1. Medoid

To get an understanding on the difficulty of the datasets, we calculated the Medoid trajectory in each set and evaluated that as the representative. To calculate the Medoid we experimented using all similarity measures at hand. We found out that while some perform slightly better than the others, in general the results are not very good, see Table 7. Medoid works very well when the set has many trajectories because the probability of one of them to match to the ground truth will increase. However, many of our sets have only a few segments, and thus, the Medoid do not work very well.

On average, Medoid provides quality score of 59% (training data) and 55% (test data). The difference between the training and testing sets is only 4%, on average, and it is consistent, which indicates that the test data is slightly more difficult.

We also tried Medoid using the same measure (HC-SIM) that was used in the evaluation. An important observation is that the result is not any better than that of the other measures. This reveals that the problem is not the choice of the distance measure but the general limitation of the Medoid itself which is restricted to select one of the input trajectories as the average. This can result in poor choices when there are lots of noisy trajectories. This is a clear indication that a better averaging method than Medoid is needed.

6.2. Overall comparison

The results of the competition are presented next. As reference, we also include results for *CellNet* [11], and *Medoid* (with IRD). CellNet is used as an integral part of a more complex system where the points in the trajectories are already provided in the same order. CellNet averaging may therefore perform worse when the data is provided unsorted. For a fair comparison, we therefore test also variant, CellNet^o, in which the k-means re-ordering step is applied. Medoid also does not include any pre-processing step but the chosen distance measure is invariant to the order of the points.

The results are summarized in Table 8. All submitted methods perform significantly better than Medoid and CellNet. This shows the importance of the contributions made to the challenge; we could basically select any of the submitted methods to replace the current segment averaging method in CellNet, and gain improvement in accuracy. Most submitted methods are also fast and suitable for real-time processing.

Marteau has the best accuracy (62.2%) but it over-represents the segment having almost 100 times the number of points what is expected according to the ground truth, thus, requiring orders of magnitude more storage space than the others. The method is also rather slow (30 min). Additional post-processing step was added after the competition by applying polygonal approximation (**Marteau+PA**). This solves the over-sampling problem at the cost of slight decrease in the accuracy from 62.2% to 61.7%. Both variants are described in detail in [Mar19b].

Next three methods (**Karasek-2, Yang-1, Yang-2**) are all fast and close to each other in terms of accuracy. **Karasek-2** was declared as the winner of the competition with 62.0% accuracy. This is only 0.2%-unit worse than by **Marteau**, but it is fast, has almost perfect length (99.1%) and the correct number of points (89.0%) is close compared to what is expected. Fewer points than in the ground truth is not necessarily bad as we have observed some ground-truth instances containing more points than necessary to represent a given shape. Therefore, fewer points can provide high quality average in some cases.

Among the main strategies, the best two methods are both averaging variants (**Marteau, Karasek-2**). The results of **Yang-1** and **Yang-2** show that the alternate strategy using simple line model can also achieve good result even when using Medoid as the backup strategy for curvy segments. This is possible because the data contains mostly many short line segments originating from city block area, and consisting of only a few points. For example, **Yang** uses the curvy model only 3.7% of times and **Dupaquis** only 8.3% of times.

Other proposals are also reasonably good in quality: **Leichter** (61.5%), **Dupaquis** (61.2%), **Amin** (61.2%) and **Karasek-1** (60.9%). The methods **Leichter** and **Karasek-1** select too few points. **Dupaquis** is too slow (10 mins) although still faster than Medoid. **Amin** is fast but provides twice as many points than expected. However, all these methods include some design choices that could be considered (see also Table 3):

- **Leichter** lacks outlier removal step
- **Dupaquis** uses naïve averaging as back-up model
- **Amin** uses possibly inferior strategy (regression)

To sum up: all submitted methods clearly outperform Medoid both in quality and speed. They all exceed 60% limit while the highest score in quality was 62.2%.

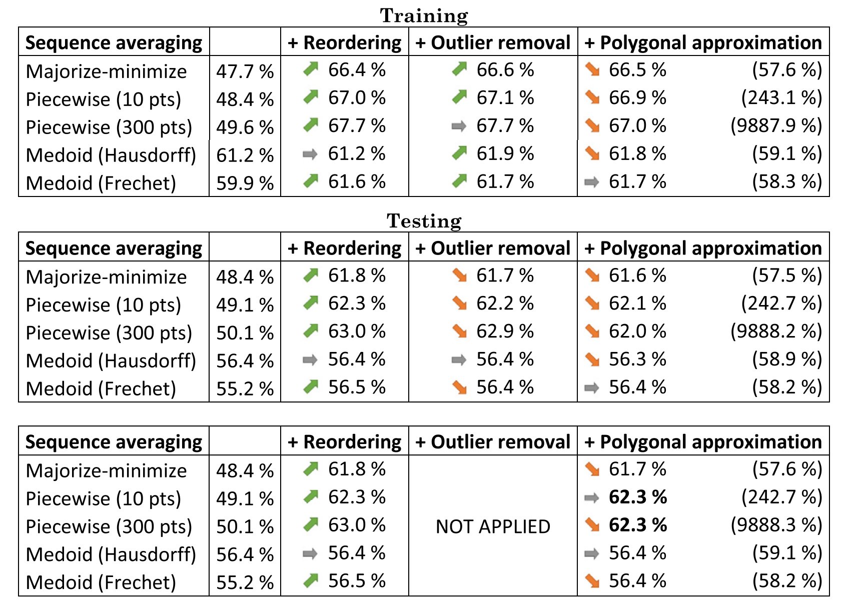
The results overall are far from 100% due to having sets where the exact ground truth is practically impossible to predict without some background information of the area. This created a kind of glass ceiling somewhere around 62%, which makes it somewhat difficult to evaluate which exact%-value is good and which not. However, based on the visual evaluation we conclude that results below 61% suffer occasional quality issues.

Visual examples are shown in Fig. 15. The methods **Dupaquis** and **Amin** have occasional truncation problems and zig zag effect. Smoothing effect also commonly appears at the corners. **Marteau** suffers this most because it uses most points in the re-sampling, which allows cutting the corner more easily. This effect also survives the polygonal approximation post-processing step. **Yang-2** suffers the smoothing effect less because of the Medoid strategy for curvy segments. In general, the methods seem to be unable to detect small curvature. Only noticeable difference is **Karasek-2** which adds one more point near the turning location.

Table 8
Summary of overall results including the submitted methods, Medoid with the best performing similarity measure (IRD), and two variants of CellNet: assuming that the points in all the trajectories follow the same direction; CellNet° for which the points are provided in order. [45–47] The results are also available on web: http://cs.uef.fi/sipu/segments/results.html.

Method	Rank	Training	Testing	Difference	Length	Points	Time
Marteau	–	68.5%	62.2%	6.3%	99%	9882%	30 min
Karasek-2	1.	67.1%	62.0%	5.1%	99%	89%	seconds
Yang-2	2.	70.4%	61.8%	8.6%	101%	83%	seconds
Yang-1	3.	68.0%	61.8%	6.2%	99%	83%	seconds
Marteau+PA	–	68.3%	61.7%	6.6%	99%	145%	30 min
Leichter	4.	66.6%	61.5%	5.1%	100%	70%	seconds
Dupaquis	5.	67.4%	61.2%	6.2%	100%	107%	10 min
Amin	6.	66.6%	61.2%	5.4%	102%	205%	seconds
Karasek-1	7.	68.1%	60.9%	7.2%	99%	67%	seconds
Medoid	–	61.9%	56.7%	5.2%	98%	169%	1 h
CellNet	–	47.7%	48.4%	−0.7%	64.9%	144%	seconds
CellNeto	–	66.4%	61.2%	4.5%	96.9%	144%	seconds

Table 9
Summary of the baseline results. We evaluate each sequence averaging method alone and when adding the reordering, outlier removal and polygonal approximation one by one. The accuracy is shown at each step. Point reduction is shown in the last column in parenthesis.



Training

Sequence averaging		+ Reordering	+ Outlier removal	+ Polygonal approximation	
Majorize-minimize	47.7 %	↗ 66.4 %	↗ 66.6 %	↘ 66.5 %	(57.6 %)
Piecewise (10 pts)	48.4 %	↗ 67.0 %	↗ 67.1 %	↘ 66.9 %	(243.1 %)
Piecewise (300 pts)	49.6 %	↗ 67.7 %	➡ 67.7 %	↘ 67.0 %	(9887.9 %)
Medoid (Hausdorff)	61.2 %	➡ 61.2 %	↗ 61.9 %	↘ 61.8 %	(59.1 %)
Medoid (Frechet)	59.9 %	↗ 61.6 %	↗ 61.7 %	➡ 61.7 %	(58.3 %)

Testing

Sequence averaging		+ Reordering	+ Outlier removal	+ Polygonal approximation	
Majorize-minimize	48.4 %	↗ 61.8 %	↘ 61.7 %	↘ 61.6 %	(57.5 %)
Piecewise (10 pts)	49.1 %	↗ 62.3 %	↘ 62.2 %	↘ 62.1 %	(242.7 %)
Piecewise (300 pts)	50.1 %	↗ 63.0 %	↘ 62.9 %	↘ 62.0 %	(9888.2 %)
Medoid (Hausdorff)	56.4 %	➡ 56.4 %	➡ 56.4 %	↘ 56.3 %	(58.9 %)
Medoid (Frechet)	55.2 %	↗ 56.5 %	↘ 56.4 %	➡ 56.4 %	(58.2 %)

Sequence averaging		+ Reordering	+ Outlier removal	+ Polygonal approximation	
Majorize-minimize	48.4 %	↗ 61.8 %		↘ 61.7 %	(57.6 %)
Piecewise (10 pts)	49.1 %	↗ 62.3 %		➡ **62.3 %**	(242.7 %)
Piecewise (300 pts)	50.1 %	↗ 63.0 %	NOT APPLIED	↘ **62.3 %**	(9888.3 %)
Medoid (Hausdorff)	56.4 %	➡ 56.4 %		➡ 56.4 %	(59.1 %)
Medoid (Frechet)	55.2 %	↗ 56.5 %		↘ 56.4 %	(58.2 %)

6.3. *Effect of the components*

We next study the effect of the design components. For this reason, we construct a new baseline method from the components as presented in Fig. 16. As baseline, we chose components that were verified to work well, we are familiar with, and implementation is readily available for us. Some of the components could be changed either for better accuracy or for simpler implementation.

6.3.1. Re-*ordering*

First, the need for the re-ordering is clear. All submitted methods apply some kind of re-ordering and several simple strategies were presented. The results with CellNet showed that lacking this component makes the system perform poorly. Medoid can avoid this component if the chosen distance function is invariant on the travel order: IRD, Hausdorff, C-SIM and HC-SIM have this property. However, other distance functions are sensitive to the point order: DTW, ERD, ERP, LCSS, Frechet, and Euclidean. If these are employed, re-ordering would become necessary also with Medoid, see Fig. 17. For the re-ordering component, we select k-means as it is readily available in most platforms.

The results, with and without re-ordering, are shown in Table 9 for all methods. It is obvious that the re-ordering is critical as almost all tested methods perform poorly without it.

6.3.2. *Outlier removal*

Almost all submitted methods adopted some kind of outlier removal step in the methods. The top three methods use outlier removal and the winner of the competition **Karasek-2** applies two types: by distance and by length.

Here we compare the new baseline and some of the submitted methods with and without outlier removal. For the outlier removal, we use the distance and length criterion as proposed by **Karasek-2**. To our surprise, the outliers had only minor effect on the final result. In fact, outlier removal improved only on the training data but had small negative impact on test data. For example, Majorize-minimize technique improved from 66.4% → 66.6% with training data but worsened from 61.8% → 61.7% with the test data, see Table 9. It seems that the methods are robust to the outliers, or significantly better outlier removal technique should be applied instead.

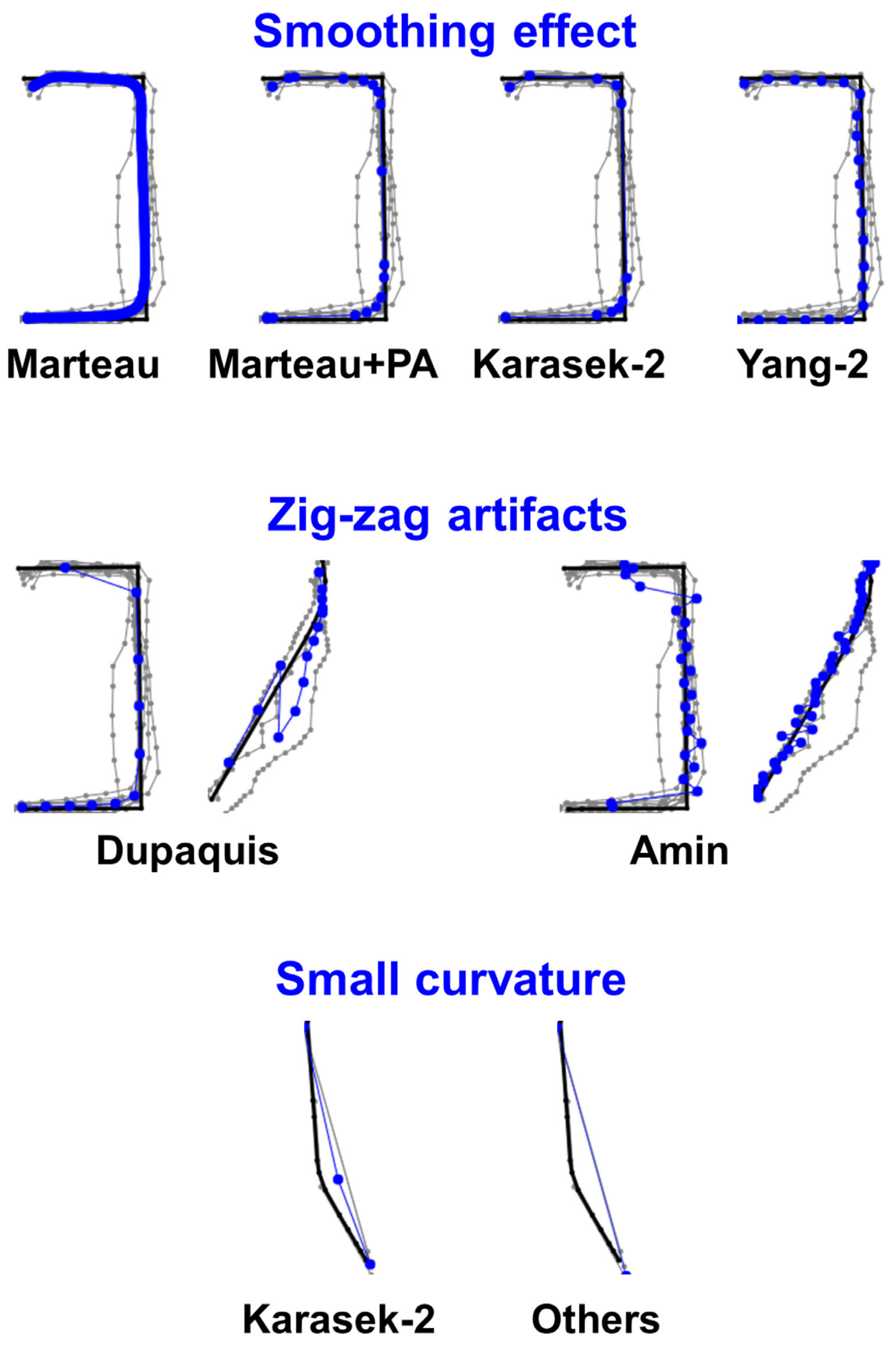


Fig. 15. Selected visual examples from various methods on different input trajectories (gray). The result is colored by blue, and the ground-truth by black.

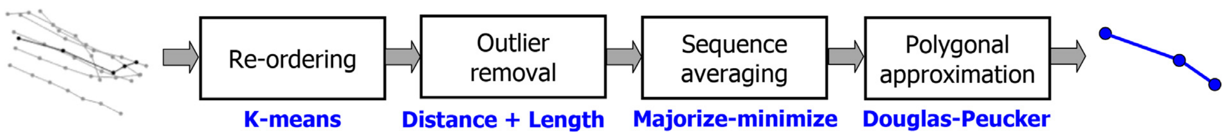


Fig. 16. New baseline based on the Segment averaging competition 2019.

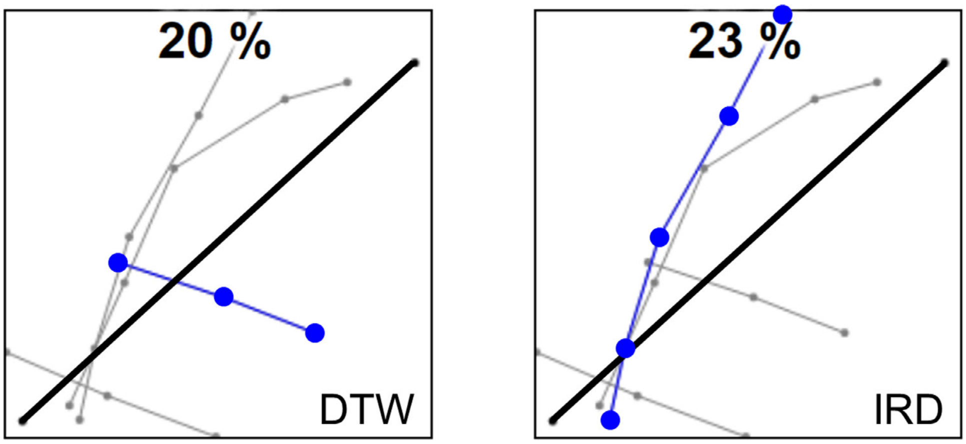


Fig. 17. Medoids (blue) obtained when the distance measure is sensitive to point order (DTW), and when it is not (IRD). The black line is the ground-truth.

6.3.3. *Sequence averaging*

Only two methods used an existing optimization method from the time series context: **Marteau** uses iTEKA [25], and CellNet uses the Majorize-minimize [32]. All other submissions relied on simpler heuristics where the key is to first match the number of points in the sequences and then use simple pointwise averaging. Medoid is the third alternative. We compare how this choice affected the result of the baseline and tested the following choices:

- Majorize-minimize
- Piecewise averaging 10 pt
- Piecewise averaging 300 pt (**Marteau**)
- Medoid with Hausdorff distance
- Medoid with Frechet distance

We tested Majorize-minimize by initializing with the shortest trajectory (as in CellNet) and with the heuristic Medoid, i.e. taking the trajectory of median length as by **Yang**. There was no significant difference between these two so we kept the shortest trajectory strategy for simplicity. For the piecewise averaging, we experimented with re-sampling each trajectory using 10 points as the minimum value, and 300 points as significant over-sampling similarly as by **Marteau**. The results in Table 9 favor the piecewise averaging in all cases. The best result is 63.0% with 300 pt over-sampling, which outperforms the highest accuracy by **Marteau** (62.2%) by 0.8%-unit.

6.3.4. *Polygonal approximation*

As shown by the results of Marteau, and the baseline with 10 points piecewise averaging, we can achieve significantly higher accuracy by over-sampling the segment. The use of polygonal approximation cannot therefore be reasoned by improving accuracy. It is applied merely to represent the segment average by minimum number of points needed to present the shape of the curvature. Nevertheless, we experiment the effect of this step by applying polygonal approximation [24].

The results in Table 9 show that the polygonal approximation degrades the quality by an amount that is relative to the number of points removed. The accuracy of the piecewise averaging decreases only by 0.1%-unit in case of 10-times over-sampling, and by 0.7%-unit in case of 100-times oversampling. However, if outliers are not removed the drop is smaller and new best performance of 62.3% is obtained by the baseline without any outlier removal.

As a consequence of these observations, we attempted to modify some of the submitted methods (**Karasek-2, Marteau-PA**) by removing their respective outlier removal components. The results in Table 10 show that both methods are improved to the same level as our new baseline if no outlier removal is applied. Therefore, applying their respective outlier removal strategies actually worsened the result on the test set.

Table 10
Effect of outlier removal steps on submitted methods.

	Original	No outlier removal
Karasek-2	62.0%	62.2%
Marteau+PA	61.7%	62.3%

A similar behavior is observed also with methods **Yang-1** (without) and **Yang-2** (with outlier removal). The results do not change on the test set even though noticeable difference occurs on the training data (68.0% → 70.4%). We also tried to add the outlier component of **Karasek-2** to **Leichter** since it was originally lacking it. The results improved slightly on the training set 66.6% → 66.8%, but stayed the same on the test set (61.5%). These experiments indicate that outlier removal is difficult to perform without generating over fit.

The medoid-based methods are sensitive to the ordering of the points if the trajectory distance measure is also sensitive (Frechet). Otherwise reordering is not necessary (Hausdorff).

7. Conclusions

We have studied how to find representative for a set of GPS trajectories by segment averaging. We have analyzed the methods submitted to the competition, and constructed a new baseline by synthesis of the proposed methods. The baseline works real-time and provides higher accuracy than any of the existing method or the submitted methods in the competition. Our main conclusions are as follows:

- *Re*-ordering the points must be done for order-sensitive methods
- Choice of sequence averaging is important
- Tested outlier removal methods had no effect or were even harmful
- The role of polygonal approximation is merely to simplify the output

7.1. *Applications*

The results can be directly used in several pattern recognition applications. These include lane detection, road segment extraction, and clustering of large-scale data. For example, existing vision-based lane detection systems mostly rely on pointwise processing by time consuming Hough transform [54]. Contrary to this, our new baseline can process 29,207 segments with 425,713 points in just few seconds and would be highly useful also in such applications.

7.2. Limitations

The current dataset was somewhat over-represented by short segments with only few sampling points. There were also many sets that had only few samples, often also very noise ones. This made it very hard to detect the outliers (if any) which made the ground truth quite challenging for the methods. In many cases, the ground truth is almost impossible to predict correctly.

All these factors caused an invisible upper limit for the accuracy. It is unclear how much better accuracy it would be possible to reach for this data without external knowledge. Now data was presented in the scale [0,1] but if we had preserved the original geo-locations, there could be ways to improve further. For example, if we knew the data comes from GPS signals and know the location of the surrounding buildings and their height, we could utilize the expected bias of the GPS coordinates. Also knowing the road network could provide significant help. In our case, the ground truth was taken from OSM so with educated guess one might even had actually concluded the ground truth.

Things to consider in the future are:

– Collect higher quality ground-truth or find a better way to estimate the ground-truth quality per sample and weight the evaluation scores based on the quality.
– Investigate how contextual information such as the surrounding buildings can be used to obtain better results.
– Evaluate methods on data with more complex shapes, such as hiking trails.
– Develop k-means based clustering method for large-scale trajectory data.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We appreciate all the authors submitting to the competition. The process has been very educating and resulting in much better insight to the problem that we could have achieved alone by ourselves. Our sincere thanks!

References

[1] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, A. Tolba, LoTAD: long-term traffic anomaly detection based on crowdsourced bus trajectory data, World Wide Web 21 (2018) 825.
[2] C. Chen, D. Zhang, P. Samuel Castro, N. Li, L. Sun, S. Li, Real-time detection of anomalous taxi trajectories from GPS traces, Mobile and Ubiquitous Systems: Computing, Networking, and Services. MobiQuitous, 104, Springer, Berlin, Heidelberg, 2012 Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.
[3] K. Bhowmick, M. Narvekar, Trajectory outlier detection for traffic events: a survey, Intelligent Computing and Information and Communication. Advances in Intelligent Systems and Computing, 673, Springer, Singapore, 2018.
[4] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network traffic anomaly detection techniques and systems, Network Traffic Anomaly Detection and Prevention. Computer Communications and Networks, Springer, Cham, 2017.
[5] E. Necula, Analyzing traffic patterns on street segments based on GPS data using R, Trans. Res. Proced. 10 (2015) 276–285.
[6] P.S. Castro, D. Zhang, C. Chen, S. Li, G. Pan, From taxi GPS traces to social and community dynamics: a survey, ACM Comput Surv 46 (2013) 2 Article 17.
[7] X. Song, Q. Zhang, Y. Sekimoto, R. Shibasaki, N.J. Yuan, X. Xie, Prediction and simulation of human mobility following natural disasters, ACM Trans. on Intell. Sys. Technol. 8 (2) (2017) 29 1-29:23.
[8] J. Davies, A.R. Beresford, A. Hopper, Scalable, distributed, real-time map generation, IEEE Pervas. Comput. 5 (4) (2006) 47–54.
[9] S. Edelkamp, S. Schrödl, Route planning and map inference with global positioning traces, Comput Sci Perspec. (2003) 128–151.
[10] L. Cao, J. Krumm, From GPS traces to a routable road map, in: ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, Seattle, Washington, USA, 2009, pp. 3–12.
[11] R. Mariescu-Istodor, P. Fränti, Cellnet: inferring road networks from gps trajectories, ACM Trans Spat. Algorithms and Sys. 4 (3) (2018).
[12] R. Mariescu-Istodor, P. Fränti, Grid-based method for GPS route analysis for retrieval, ACM Trans Spat Algorithms and Sys 3 (3) (2017).
[13] M. Ahmed, S. Karagiorgou, D. Pfoser, C. Wenk, A Comparison and Evaluation of Map Construction Algorithms, Geoinformatica 19 (3) (2015) 601–632.
[14] J. Biagioni, J. Eriksson, Inferring road maps from global positioning system traces: survey and comparative evaluation, Trans. Res. Rec.: J Transp Res. Board (2291) (2012) 61–71.
[15] R. Trasarti, R. Guidotti, A. Monreale, F. Giannotti, Myway: location prediction via mobility profiling, Inf Syst 64 (2017) 350–367.
[16] M. Vlachos, G. Kollios, D. Gunopulos, Discovering similar multidimensional trajectories, in: IEEE Int. Conf. on Data Engineering, 2002, pp. 673–684.
[17] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD workshop, 10, Seattle, WA, USA, AAAI Press, 1994, pp. 359–370.
[18] L. Chen, M.T. Ozsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: *ACM SIGMOD Int. Conf. on Management of data and Symposium on Principles Database and Systems* (SIGMOD/PODS), Baltimore, MD, USA, 2005, pp. 491–502.
[19] T. Eiter, H. Mannila, Computing Discrete Fréchet distance, Information Systems Department, Technical University of Vienna, 1994 *Tech. Report* CD-TR 94/64.
[20] L. Chen, R. Ng, On the marriage of lp-norms and edit distance, in: Int. Conf. on Very Large Data Bases, Toronto, Canada, 2004, pp. 792–803.
[21] I.S. Gradshteyn, I.M. Ryzhik, in: Tables of Integrals, Series, and Products, 6th ed., Academic Press, San Diego, CA, 2000, pp. 1114–1125.
[22] T.R. Rockafellar, R.J.-B. Wets, Variational Analysis, 317, Springer Science & Business Media, 2009.
[23] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space, in: ACM Int. Conf. on Knowledge Discovery and Data Mining Workshop on Mining Temporal and Sequential Data, Seattle, Washington, USA, 2007, pp. 70–80.
[24] D.H. Douglas, T.K. Peucker, Algorithm for the reduction of the number of points required to represent a line or its caricature, The Can. Cartogr. 10 (2) (1973) 112–122.
[25] P.-F. Marteau, Times series averaging and denoising from a probabilistic perspective on time-elastic kernels, Int. J. Applied Math Comput. Sci. 29 (2) (June 2019).
[26] P.-F. Marteau, S. Gibet, On recursive edit distance kernels with application to time series classification, IEEE Trans Neural Netw Learn Syst 26 (6) (2015) 1121–1133.
[27] P.-F. Marteau, G. Ménier, Speeding up simplification of polygonal curves using nested approximations, Pattern Analysis and Appl. 12 (2009) 367–375.
[28] L. Bulteau, V. Froese, R. Niedermeier, "Tight hardness results for consensus problems on circular strings", ArXiv:1804.02854-v4, 2019.
[29] L. Gupta, D.L. Molfese, R. Tammana, P.G. Simos, Nonlinear alignment and averaging for estimating the evoked potential, IEEE Trans Biomed Eng. 43 (4) (1996) 348–356.
[30] B. Jain and D. Schultz, "On the existence of a sample mean in dynamic time warping spaces", arXiv preprint: arXiv:1610.04460, 2016.
[31] W.H. Abdulla, D. Chow, G. Sin, Cross-words reference template for DTW-based speech recognition systems, in: Proc. TENCON, 2, Bangalore, 2003, pp. 1576–1579.
[32] V. Hautamäki, P. Nykänen, P. Fränti, Time-series clustering by approximate prototypes, Int. Conf. on Pattern Recognition, 2008.
[33] F. Petitjean, A. Ketterlin, P. Gancarski, A global averaging method for dynamic time warping, with applications to clustering, Pattern Recognit 44 (3) (2011) 678–693.
[34] M. Cuturi, M. Blondel, Soft-DTW: a differentiable loss function for time-series, in: Int. Conf. on Machine Learning, 70, Sydney, Australia, 2017, pp. 894–903.
[35] D. Schultz, B. Jain, Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces, Pattern Recognit 74 (Supplement C) (2018) 340–358.
[36] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, D. Schultz, Exact mean computation in dynamic time warping spaces, Data Min Knowl Discov 33 (2019) 252–291.
[37] E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classification, Biom. 21 (1965) 768–780.
[38] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, Calif., University of California Press, 1967, pp. 281–297.
[39] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. on Inf. Theory 28 (2) (1982) 129–137.
[40] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, C. Wilson, Mining GPS traces for map refinement, Data Mining Knowl. Discov. 9 (2004) 59–87.
[41] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained k-means cluster-

ing with background knowledge, in: Int. Conf, on Machine Learning, 2001, pp. 577–584.
[42] L. Piegl, W. Tiller, The Nurbs Book, Springer Verlag, 1997.
[43] M. Chen, M. Xu, P. Fränti, A fast O(N) multi-resolution polygonal approximation algorithm for GPS trajectory simplification, IEEE Trans. Image Process. 21 (5) (2012) 2770–2785.
[44] P.-.F. Marteau, Estimating road segments using kernelized averaging of GPS trajectories, Appl. Sci. 9 (13) (2019) 2736.
[45] T. Karasek, "SEGPUB.IPYNB", Github, 2019. https://gist.github.com/t0mk/eb640963d7d64e14d69016e5a3e93fd6
[46] M. Amin, "AveragingGPSSegments", Github, 2019. https://github.com/mohamad-amin/AveragingGPSSegments
[47] J.W. Yang, R. Mariescu-Istodor, P. Fränti, Three rapid methods for averaging GPS segments, Appl Sci. 9 (22) (2019) 4899.
[48] M. Ester, H.-.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Int. Conf. on Knowledge Discovery and Data Mining (KDD), 1996, pp. 226–231.
[49] J.E. Ertel, E.B. Fowlkes, Some algorithms for linear spline and piecewise multiple linear regression, J Am Stat Assoc 71 (355) (1976) 640–648.
[50] S. Rosset, J. Zhu, Piecewise linear regularized solution paths, The Annals of Stat. (2007) 1012–1030.
[51] R. Stanojevic, S. Abbar, S. Thirumuruganathan, S. Chawla, F. Filali, A. Aleimat, Robust road map inference through network alignment of trajectories, in: Proceedings of the 2018 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, 2018, pp. 135–143.
[52] T.K. Dey, J. Wang, 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, ACM, 2017, p. 58.
[53] M. Morel, C. Achard, R. Kulpa, S. Dubuisson, Time-series averaging using constrained dynamic time warping with tolerance, Pattern Recognit 74 (2018) 77–89.
[54] S.P. Narote, P.N. Bhujbal, A.S. Narote, D.M. Dhane, A review of recent advances in lane detection and departure warning system, Pattern Recognit 73 (2018) 216–234.

Pasi Fränti received his MSc and PhD degrees in computer science from the University of Turku, Finland, in 1991 and 1994, respectively. From 1996 to 1999 he was a postdoctoral researcher funded by the Academy of Finland. Since 2000, he has been a professor in the University of Eastern Finland (Joensuu) where he is leading the machinbe learning group. Prof. Fränti has published 91 refereed journal and over 170 conference papers. His primary research interests are in clustering and mobile location-based applications but include also other topics in algorithms, artificial intelligence, data mining and machine learning.

Radu Mariescu-Istodor received the MSc and PhD degrees in computer science from the University of Eastern Finland, in 2013 and 2017, respectively. From 2017 he is a postdoctoral researcher working at the University of Eastern Finland. His-research includes GPS trajectory analysis, machine learning and data clustering.