# Towards Effective Trajectory Similarity Measure in Linear Time

Yuanjun Liu[1], An Liu[1]✉, Guanfeng Liu[2], Zhixu Li[3], and Lei Zhao[1]

[1] School of Computer Science and Technology, Soochow University, China
{20214227045@stu., anliu@, zhaol@}suda.edu.cn
[2] School of Computing, Macquarie University, Australia guanfeng.liu@mq.edu.au
[3] School of Computer Science and Technology, Fudan University, China
zhixuli@fudan.edu.cn

**Abstract.** With the utilization of GPS devices and the development of location-based services, a massive amount of trajectory data has been collected and mined for many applications. Trajectory similarity computing, which identifies the similarity of given trajectories, is the fundamental functionality of trajectory data mining. The challenge in trajectory similarity computing comes from the noise in trajectories. Moreover, processing such a myriad of data also demands efficiency. However, existing trajectory similarity measures can hardly keep both accuracy and efficiency. In this paper, we propose a novel trajectory similarity measure termed ITS, which is robust to noise and can be evaluated in linear time. ITS converts trajectories into fixed-length vectors and compares them based on their respective vectors' distance. Furthermore, ITS utilizes interpolation to get fixed-length vectors in linear time. The robustness of ITS owes to the interpolation, which makes trajectories aligned and points in trajectories evenly distributed. Experiments with 12 baselines on four real-world datasets show that ITS has the best overall performance on five representative downstream tasks in trajectory computing.

**Keywords:** Trajectory similarity measure· Trajectory distance measure· Trajectory data mining· GPS data mining.

## 1 Introduction

The last decade has witnessed unprecedented growth in the availability of location-based services. Large amounts of mobile data described as sequences of locations, known as trajectories, are generated and mined for many applications. Typical examples include collecting GPS location histories for management purposes, such as tracking typhoons for better precautions, providing ordinary users better routes, planning for trucks and carpooling, and tracking migration of animals in biological sciences [15, 23]. Trajectory offers unprecedented information to help understand the behaviors of moving objects, resulting in a growing interest in trajectory data mining. An essential problem in mining is designing measures for identifying similar trajectories. Many data analysis tasks can use trajectory

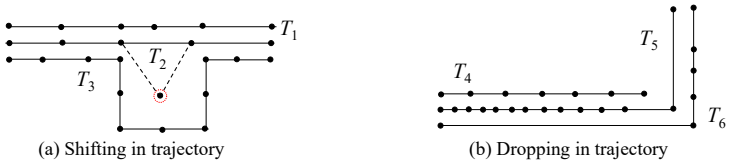


**Fig. 1.** Noise in trajectories. (a): The real trajectory of $T_1$ and $T_2$ are more similar, since they are both straight. Unfortunately, a shifted point brings a fake corner to $T_2$, so $T_2$ may be more similar to $T_3$ than $T_1$. (b): The real trajectory of $T_5$ and $T_6$ are more similar. Nevertheless, most points of $T_5$ are closer to points of $T_4$, then $T_5$ may be more similar to $T_4$ than $T_6$.

similarity measures, including trajectory clustering, classification, and $k$-nearest neighbor ($k$-nn) search, which have a broad range of real applications. For example, in the urban construction and planning bureau, it is helpful to locate popular routes by comparing similarities between vehicle trajectories and deciding on new transportation routes. During COVID-19, governments can track epidemics and warn citizens by searching other trajectories similar to those of attacked person [14]. Navigation systems search similar trajectories close to origin and destination and recommend better routes to users [3].

Several challenges lie in trajectory similarity computing. GPS devices collect trajectories while objects are moving, but noises may be caused during collection. Two kinds of noises, shifting and dropping, exist in trajectories and bring error. Firstly, the positions of points may be shifted since the precision of GPS devices, as reported in [20]: "Typically, the GPS nominal accuracy is about 15m". As shown in Figure 1(a), there is a shifted point in $T_2$. Technically, the actual trajectories of $T_1$ and $T_2$ are more similar since they are both straight. Unfortunately, the shifted point brings a fake corner to $T_2$ so that $T_2$ may be more similar to $T_3$ than $T_1$.

Secondly, objects move at various speeds, but GPS devices sample at consistent intervals. A fast speed leads to a high dropping rate, and a slow speed leads to a low dropping rate. On the one hand, the speeds between objects are typically different, so the dropping rates between trajectories are uncertain. On the other hand, the speed of an object may change over time, so the dropping rate inside a trajectory changes over time, also causing unfairness when counting matched points. Figure 1(b) shows $T_5$ has a low dropping rate at the left half and a high dropping rate at the right half. Meanwhile, the dropping rate of $T_6$ is higher than $T_5$. The actual trajectories of $T_5$ and $T_6$ are similar. However, most points of $T_5$ are closer to points of $T_4$ than points of $T_6$, leading $T_5$ to be more similar to $T_4$ than $T_6$. Shifting and dropping make it challenging to keep accuracy while measuring trajectory similarity.

The third challenge is efficiency. With the ubiquity of positioning techniques, various trajectory data are generated constantly. For example, there are about 1TB GPS data generated each day in JD [11]. Suppose there is a set of trajectories and each trajectory has $n = 1000$ points. While an $O(n)$ method takes 1 second to process such a dataset, an $O(n^2)$ method takes about 2 minutes.

**Table 1.** Summary of trajectory similarity measures. $n$ is the number of points in a trajectory. A good measure should have a short running time and high robustness.

| Measures | Time | Robustness | Parameter |
|---|---|---|---|
| DTW [1] | $O(n^2)$ | High | Free |
| PDTW [9] | $O(n^2)$ | High | Need |
| FastDTW [21] | $O(n)$ | Low | Need |
| BDS [24] | $O(n^2)$ | Medium | Free |
| SSPD [2] | $O(n^2)$ | Medium | Free |
| FD [15] | $O(n^2)$ | High | Free |
| LIP [18] | $O(nlogn)$ | Medium | Free |
| OWD [13] | $O(n^2)$ | High | Need |
| STS [10] | $O(n^2)$ | Medium | Need |
| LCSS [26] | $O(n^2)$ | Low | Need |
| EDR [5] | $O(n^2)$ | High | Need |
| ERP [4] | $O(n^2)$ | Medium | Free |
| MD [7] | $O(n^2)$ | Medium | Free |
| CATS [6] | $O(n^2)$ | Medium | Need |
| STED [17] | $O(n^2)$ | Medium | Free |
| STLC [22] | $O(n^2)$ | Low | Need |
| STLCSS [26] | $O(n^2)$ | Low | Need |
| RID [25] | $O(n)$ | Low | Free |
| EDwP [19] | $O(n^2)$ | Medium | Free |
| ITS | $O(n)$ | High | Free |

Therefore, a fast trajectory similarity measure is a must to mine such a massive amount of trajectory data.

Table 1 summarizes existing trajectory similarity measures. An important observation from the table is that all existing measures can hardly be accurate and efficient. Matching-based measures [1, 2, 21, 24] are generally robust to variable dropping rates but sensitive to shifting. Sequence-based measures [4, 5, 7, 26] are sensitive to dropping rates as the number of points in trajectories changes with the dropping rate. Shape-based measures [15, 13, 18] are robust to the dropping rate, but most need to improve at handling shifting. The running time of time-based measures [6, 17, 22] is $O(n^2)$, and they also generally need parameters that are set manually. EDwP [19] employs projection to align trajectories but takes $O(n^2)$ time to find the best projection positions. RID [25] is $O(n)$ time, but it aims to identify trajectories after the rotation operation. FastDTW [21] is also $O(n)$ time but is not robust. Apart from the above methods, machine learning has recently been adopted to calculate the similarity of trajectories. These machine learning-based methods [12, 28] can calculate similarity in $O(n)$ time but need training and parameter-tuning. As they are not measures, we do not include them in Table 1.

To overcome the above limitations of existing measures, we propose a novel trajectory distance measure termed ITS, which is robust to shifting and variable dropping rates and can be evaluated in linear time. The basic idea behind it is to generate a fixed-length vector for each trajectory and then compare two trajectories based on the distance of their respective vectors. Furthermore,

unlike machine learning-based methods [12, 28] in which trajectories are also represented by vectors, ITS employs a fast interpolation strategy to convert trajectories into vectors, thus avoiding expensive training and parameter-tuning. During interpolation, two trajectories are aligned, and points in a trajectory are evenly distributed, so ITS is robust to variable dropping rates. Shifting can only disturb interpolation points around the shifted area, so ITS is also robust to shifting.

In summary, the paper makes the following contributions:

- We propose a novel trajectory measure termed ITS, which has linear time complexity and space complexity.
- ITS is robust to dropping and shifting in trajectory data and satisfies the triangle inequality.
- We compare ITS with 12 baselines on five important downstream tasks in trajectory computing using four real-world datasets. Experimental results show that ITS has the best overall performance.

The rest of the paper is organized as follows. Section 2 discusses the related work. The definition and problem statement are given in Section 3. Section 4 presents the details of our method. The experimental results are presented in Section 5. Finally, we summarize our work in Section 6.

## 2 Related Work

Existing trajectory similarity measures can be classified into several groups according to the technologies they use. We summarize them in Table 1 and discuss their details below.

Matching is one of the most popular technology to measure trajectory similarity. Dynamic Time Warping (DTW) [1] matches points to the nearest point and measures the distances of matched point-pairs. DTW warps trajectories in a non-linear way while allowing two trajectories have different dropping rates. However, DTW matches every point and ignores point distribution, so it cannot handle changing dropping rates inside the trajectory. The running time of DTW is $O(n^2)$ as it compares every point pair. Piecewise DTW (PDTW) [9] takes every several points into a piece, replaces them with their average, and then applies DTW to measure the similarity of averaged trajectories. Averaging makes PDTW robust to shifting, but the number of points to be averaged should be determined artificially. PDTW reduces the number of points to speed up DTW but also consumes $O(n^2)$ time. FastDTW [21] restricts the matching window size to get $O(n)$ computation complexity but at the cost of accuracy. The window size of FastDTW needs to be set manually. Symmetrized Segment-Path Distance (SSPD) [2] matches points to the nearest segment, and Bi-Directional Similarity (BDS) [24] matches points to every segment. They measure the distances of matched point-segment pairs. They are robust to variable dropping rates but not robust to shifting.

In the case of sequence based measures, Longest Common Subsequence (LCSS) [26], Edit Distance on Real sequence (EDR) [5] and Edit distance with Real Penalty (ERP) [4] treat points to be the same as long as the distance between two points is less than a threshold. The threshold of ERP is the distance between a point to the original point. Others should be set manually. They are robust to the shifting that is smaller than the threshold. Merge Distance (MD) [7] finds the shortest trajectory merged by two trajectories. It is robust to variable dropping rates since it considers the length of segments. They all compare the distance of every point pair, which yields $O(n^2)$ time.

Regarding shape-based measures, Fréchet Distance (FD) [15] between two curves is defined as the minimum length of a leash required to connect two separate paths. It regards trajectories as curves, thus being robust to dropping. It compares all point pairs, which yields $O(n^2)$ time. Locality In-between Polylines distance (LIP) [18] counts the area where two trajectories are enclosed. LIP can be transformed into the red-blue intersection problem so that it can be solved in $O(nlogn)$ time. One Way Distance (OWD) [13] grids trajectories and calculates the distance of grids. Its running time is $O(n^2)$, and the grid size needs to be set manually. OWD is robust to shifting that does not exceed the grid size. LIP and OWD fill the gap between adjacent points, which normalizes the distribution of points, thus being robust to variable dropping rates.

Time is also taken into consideration for trajectory similarity computation. Spatiotemporal Euclidean Distance (STED) [17] compares positions for each time instant, but it requires a common temporal domain for trajectories, which is a hard requirement. It excludes subsequence matching and shifting that tries to align trajectories. It has $O(n^2)$ running time. The time instant interpolates the unsampled area; thus, STED is robust to variable dropping rates. Spatiotemporal Linear Combine distance (STLC) [22] measures similarities from both spatio and temporal aspects and needs parameters to control the relative importance of the spatial and temporal similarities. Clue-Aware Trajectory Similarity (CATS) [6] and SpatioTemporal LCSS (STLCSS) [26] match points if the time interval is less than a threshold. Just like LCSS, they are robust to shifting that is smaller than the threshold. They compare every point pair, yielding $O(n^2)$ time.

Spatial-Temporal Similarity (STS) [10] estimates the probabilities of points and measures the similarity of probabilities. Its running time is $O(n^2)$. Probability brings robustness of shifting. Rotation Invariant Distance (RID) [25] records polar radius at each polar angle and utilizes FastDTW to measure the similarity of radius. Its running time is $O(n)$, but the problem it solves is the similarity of trajectories after rotation. Edit Distance with Projections (EDwP) [19] aligns trajectories by projecting points onto another trajectory, thus being robust to dropping. However, finding the best projection positions takes $O(n^2)$ time.

ML-based methods [12, 28] generate representation vectors for trajectories based on neural networks. They are robust to variable dropping rates because they drop points while training. However, training and parameter-tuning are needed in ML-based methods, making them data-dependent and resource-consuming.

Besides, it is hard to use them in applications where indexing and pruning are essential. What is more, they are not measures and are generally unexplainable.

## 3 Problem Formulation

**Definition 1 (Original Trajectory).** *An original trajectory is a sequence of points $T = [p_1, p_2, \cdots, p_n]$ where $p_i = (x_i, y_i)$ is a spatial point in 2D space, $x_i$ and $y_i$ are latitude and longitude, respectively. The length of $T$ is defined as the number of points in $T$, that is, $n$.*

**Definition 2 (Sample Point).** *Given a point $p = (x, y)$, its sample point is given by $\phi(p) = (x + \Delta(x), y + \Delta(y))$ where $\phi$ is a sample function, $\Delta(x)$ and $\Delta(y)$ are the noise introduced during sampling.*

In the above definition, we consider shifting, one noise resulting from physical devices that generate trajectory data. Next, we will consider another kind of noise.

**Definition 3 (Sample trajectory).** *Given an original trajectory $T$, its sample trajectory is also a sequence of points $T' = \varphi(T) = [\phi(p_{s_1}), \phi(p_{s_2}), \cdots, \phi(p_{s_m})]$ where $1 \leq s_1 < s_2 < \cdots < s_m \leq n, \varphi$ samples the trajectory.*

The above definition of sample trajectory indicates that the length of a sample trajectory $T'$ is less than or equal to the original trajectory $T$. In addition, this definition simulates another noise called dropping in practice; that is, not all points in a trajectory can be sampled due to the continuity of movement and the discretization of sampling.

**Problem Statement.** *The problem to be addressed is to accurately evaluate the similarity of two trajectories in the presence of noise. Specifically, given two sample trajectories $T'_i$ and $T'_j$, we need to design a function $f(\cdot)$ to compute their similarity, with a goal of the similarity of two sample trajectories $f(T'_i, T'_j)$ should be as close as possible to the similarity of their respective original trajectories $f(T_i, T_j)$, that is, $f(T'_i, T'_j) \approx f(T_i, T_j)$.*

## 4 Interpolation based Trajectory Similarity

As mentioned earlier, lots of works try to transform trajectories into fixed-length vectors so that the similarity of trajectories can be evaluated efficiently based on the similarity of their respective vectors. Following this idea, we propose in this paper an interpolation-based trajectory similarity measure ITS, which adopts interpolation to generate a fixed-length vector representation for trajectories. Interpolation is a lightweight operation, so ITS is efficient. It can be done without the time-consuming training and parameter-tuning involved in the ML-based methods or the complex computation (e.g., typically quadratic time) to find the best matching between points needed in traditional methods such as DTW and EDR.

Generally speaking, ITS interpolates a predefined number of points, say $h$ points, to a trajectory despite its length. Therefore, given two trajectories $T_1$ and $T_2$ that may have different lengths, ITS generates two interpolated trajectories $\hat{T}_1 = [\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_h]$ and $\hat{T}_2 = [\hat{q}_1, \hat{q}_2, \cdots, \hat{q}_h]$. The numbers of points in the interpolated trajectories are the same, so they can be matched efficiently by taking $(\hat{p}_i, \hat{q}_i)$ as a pair.

The interpolation should be designed delicately to keep the accuracy of similarity computing. Given a trajectory $T = [p_1, p_2, \cdots, p_n]$, let $\lambda(p_i) \in [0, 1]$ be the normalized arc length from $p_1$ to $p_i$, that is, $\lambda(p_i) = \sum_{j=2}^{i} d(p_{j-1}, p_j)/L(T)$, where $L(T) = \sum_{j=2}^{n} d(p_{j-1}, p_j)$ is the arc length of trajectory $T$, $d$ is the Euclidean distance. Let $P(l)$ be an interpolated point with the normalized arc length $l$, that is, $\lambda(P(l)) = l$. ITS interpolates points evenly over the arc length of the whole trajectory. Figure 2 illustrates the interpolation procedure. First, the trajectory $T$ is considered to be a set of consecutive segments, as shown in Figure 2(a). Then ITS straightens these segments so that the trajectory becomes one large segment shown in Figure 2(b). After that, this large segment is divided into $h-1$ parts evenly. Finally, as shown in Figure 2(c), the $h$ endpoints of these parts are reported as the interpolation points. Formally, given a trajectory $T$ and an integer $h$, ITS generates an interpolated trajectory $\hat{T} = [\, \hat{p}_i \,|\, 1 \le i \le h \,]$, where $\lambda(\hat{p}_i) = \frac{i-1}{h-1}$.

Algorithm 1 shows the details of interpolation. Given a trajectory $T$ and an integer $h$, the algorithm generates $h$ points and returns the interpolated trajectory in the form of vector $v$. The algorithm walks on the trajectory with a constant step $\varepsilon$ (Line 2), and yields $(\boldsymbol{v}_j, \boldsymbol{v}_{j+h}) = P(\frac{j-1}{h-1})$ at each interpolation position.

The above interpolation strategy is efficient since the trajectory needs to be traveled once. It is also effective as it tries to keep the semantics of the original trajectory. In particular, the shape of the trajectory mostly stays the same. Moreover, the points in a trajectory are evenly distributed, which is very useful when estimating the similarity of two trajectories.

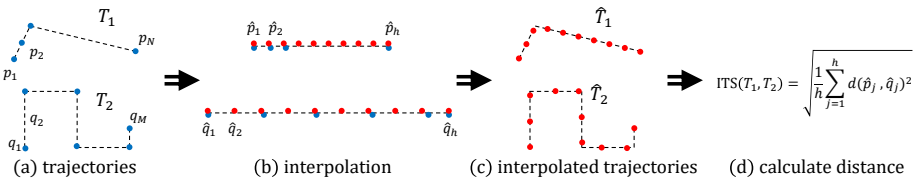


**Fig. 2.** Illustration of ITS. Blue points are points in the original trajectories, and red points are interpolation points generated by ITS.

Algorithm 2 shows how ITS works. To compute the similarity of two trajectories $T_1$ and $T_2$, ITS first generates two interpolated trajectories $\hat{T}_1 = [\hat{p}_j \,|\, 1 \le j \le h]$ where $\hat{p}_j = P_{\hat{T}_1}(\frac{j-1}{h-1})$ and $\hat{T}_2 = [\hat{q}_j \,|\, 1 \le j \le h]$ where $\hat{q}_j = P_{\hat{T}_2}(\frac{j-1}{h-1})$. Then it takes the square root of the average sum of squared Euclidean distances

between every pair $(\hat{p}_j, \hat{q}_j)$ as the distance of $T_1$ and $T_2$:

$$\mathrm{ITS}(T_1, T_2) = \sqrt{\frac{1}{h}\sum_{j=1}^{h} d(\hat{p}_j, \hat{q}_j)^2} \tag{1}$$

Finally, the similarity of trajectories $T_1$ and $T_2$ can be set to their negative distance, that is, $-\mathrm{ITS}(T_1, T_2)$.

**Algorithm 1:** Interpolation

**Input:** trajectory $T$, integer $h$
**Output:** interpolated vector $\boldsymbol{v}$

$\boldsymbol{v}_j \leftarrow 0, 1 \le j \le 2h$
$\varepsilon \leftarrow \frac{L(T)}{h-1}$
$a, b \leftarrow 0, 0$
$j \leftarrow 0$
**for** $i \leftarrow 2$ **to** $n$ **do**
  $w \leftarrow d(p_{i-1}, p_i)$
  $b \leftarrow b + w$
  **while** $b > a$ **do**
    $\boldsymbol{v}_j \leftarrow (x_i - x_{i-1})(1 - \frac{b-a}{w}) + x_{i-1}$
    $\boldsymbol{v}_{j+h} \leftarrow (y_i - y_{i-1})(1 - \frac{b-a}{w}) + y_{i-1}$
    $j \leftarrow j + 1$
    $a \leftarrow a + \varepsilon$

**Algorithm 2:** ITS

**Input:** trajectory $T_1$, trajectory $T_2$, integer $h$
**Output:** distance $its$

$\boldsymbol{v_1}, \boldsymbol{v_2} \leftarrow \mathrm{Interpolation}(T_1, h), \mathrm{Interpolation}(T_2, h)$
$its \leftarrow d(\boldsymbol{v}_1, \boldsymbol{v}_2)/\sqrt{h}$

**Theoretical Analysis.** ITS can be evaluated in $O(n)$ time and $O(h)$ space. During interpolation (Algorithm 1), calculating $L(T)$ costs $n$ arithmetical operations. The outer loop traverses $n$ points, and the inner loop steps $h$ interpolation positions. Thus, interpolation needs $2n+h$ operations and $2h$ space. During ITS (Algorithm 2), interpolating trajectories costs $4n + 2h$ operations, and calculating the vectors' distance costs $h$ operations. Therefore, ITS needs $4n + 3h$ operations and $4h$ space. Moreover, ITS has some desirable properties, given by the following theorem.

**Theorem 1.** *Given three trajectories $T_1, T_2, T_3$, ITS satisfies:*

1. *Non-negativity:* $\mathrm{ITS}(T_1, T_2) \ge 0$.
2. *Semi-identity:* $T_1 = T_2 \Rightarrow \mathrm{ITS}(T_1, T_2) = 0$.
3. *Symmetry:* $\mathrm{ITS}(T_1, T_2) = \mathrm{ITS}(T_2, T_1)$.

4. *Triangle inequality:* $\mathrm{ITS}(T_1, T_2) + \mathrm{ITS}(T_2, T_3) \geq \mathrm{ITS}(T_1, T_3)$.

*Proof.* Given two trajectories $T_1, T_2$, $\hat{T}_1 = [\hat{p}_j \,|\, 1 \leq j \leq h]$ where $\hat{p}_j = P_{\hat{T}_1}(\frac{j-1}{h-1})$ and $\hat{T}_2 = [\hat{q}_j \,|\, 1 \leq j \leq h]$ where $\hat{q}_j = P_{\hat{T}_2}(\frac{j-1}{h-1})$ are interpolated trajectories,

$$d(\cdot) \geq 0 \Rightarrow \sqrt{\frac{1}{h}\sum_{j=1}^{h} d(\hat{p}_j, \hat{q}_j)^2} \geq 0 \Rightarrow \mathrm{ITS}(T_1, T_2) \geq 0,$$

$$T_1 = T_2 \Rightarrow d(P_{\hat{T}_1}(l), P_{\hat{T}_1}(l)) = 0 \Rightarrow \sqrt{\frac{1}{h}\sum_{j=1}^{h} d(\hat{p}_j, \hat{q}_j)^2} = 0 \Rightarrow \mathrm{ITS}(T_1, T_2) = 0,$$

$$\mathrm{ITS}(T_1, T_2) = \sqrt{\frac{1}{h}\sum_{j=1}^{h} d(\hat{p}_j, \hat{q}_j)^2} = \sqrt{\frac{1}{h}\sum_{j=1}^{h} d(\hat{q}_j, \hat{p}_j)^2} = \mathrm{ITS}(T_2, T_1).$$

Given three trajectories $T_1, T_2, T_3$,

$$\mathrm{ITS}(T_1, T_2) + \mathrm{ITS}(T_2, T_3) = d(v_1, v_2)/\sqrt{h} + d(v_2, v_3)/\sqrt{h} \geq d(v_1, v_3)/\sqrt{h} = \mathrm{ITS}(T_1, T_3)$$

where $v_1 = \mathrm{Interpolation}(T_1, h), v_2 = \mathrm{Interpolation}(T_2, h), v_3 = \mathrm{Interpolation}(T_3, h)$, and $v_1, v_2, v_3$ are viewed as points in $2h$ dimensional space.

As ITS satisfies the triangle inequality, generic indexing structures and pruning strategies can be applied when dealing with a huge amount of trajectories.

The performance of ITS depends on the value of $h$. Generally, a larger $h$ means an interpolated trajectory has more points, which improves the accuracy of ITS but increases its running time. On the contrary, the evaluation of ITS with a smaller $h$ is more efficient. However, its accuracy will decrease as it may need to capture more semantic information about trajectories. Therefore, it is necessary to strike a balance between accuracy and efficiency. As discussed in Section 5.3, a suitable $h$ depends on the length of trajectories. When evaluating the similarity of two trajectories, $h$ can be set to their average length. Therefore, we consider ITS to be parameter-free. As $h = O(n)$, ITS has linear time and space complexity.

## 5 Experiment

We evaluate ITS by comparing it with 12 baselines on four real-world datasets. Section 5.1 gives the details of the selected baselines and datasets. In Section 5.2, we evaluate different measures using five downstream tasks, including rank, precision, cross-similarity, $k$-nn queries, and clustering. Finally, we analyze the effect of the parameter of ITS in Section 5.3.

**Table 2.** Dataset statistics

| Dataset | Split gap | Mean length | # Trajectory |
|---|---|---|---|
| Porto [16] | - | 59 | 1,243,663 |
| T-drive [27] | 12 mins | 111 | 146,761 |
| GeoLife [29] | 10 secs | 131 | 1,392 |
| ASL [8] | - | 57 | 2,565 |

### 5.1 Experimental setup

**Datasets.** Table 2 shows four real-world datasets used in our experiments. For ASL, we use it in clustering tasks only and take the $x$ and $y$ dimensions only. Followed by [12, 19], for T-drive and GeoLife, we split the trajectories if the time gap between consecutive points is too large. We remove the trajectories with points less than 30 since they can hardly contain enough information after dropping. We divide the trajectories with points more than 300 since they have sufficient information, and more points cost too much time for $O(n^2)$ methods. Meanwhile, the mean lengths of datasets are far away from 300. We sample 100 trajectories randomly and yield $100^2$ pairs in each dataset. Due to the limited space, we show the average results on all datasets in Section. 5.2.

**Baselines.** We select 12 representative baselines for experiments: FD [15], DTW [1], FastDTW [21], LCSS [26], EDR [5], ERP [4], OWD [13], CATS [6], BDS [24], SSPD [2], EDwP [19] and STS [10]. All of them are introduced in Section 2. We implement them using Python. Following [10, 23], we do not include ML-based methods since they are not measures. First, we set the spatial threshold of LCSS, EDR, and CATS to 15 meters [24] on other tasks, and to 0.01 meters on the clustering task duo to the nature of ASL dataset. Next, we set the temporal threshold of CATS to infinity, the radius of FastDTW to 10, and the granularity of OWD to 20. Finally, we convert the global grid size of STS into the co-location grid number, which is set to 16.

**Downstream Tasks.** Inspired by [2, 10, 12], we evaluate different measures using rank and precision, cross-similarity, $k$-nn queries, and clustering. The noises on trajectories are shifting and dropping. Let $\mathbb{T} = (T_1, T_2, \ldots, T_{|\mathbb{T}|})$ be original trajectories, $\mathbb{T}' = \{T' | T' = \varphi(T), T \in \mathbb{T}\}$ be sampled trajectories with noises. We take results on $\mathbb{T}$ as ground truth to evaluate the performances on $\mathbb{T}'$. To study the robustness of measures with noises, we perform four kinds of transformations. First, we drop the points at the whole range of $T$ with a rate $r_1$, representing various dropping rates between trajectories. Next, we drop the points at the first half or the second half of $T$ with a rate $r_2$, representing changing dropping rates inside trajectories. Furthermore, we introduce shifting by adding Gaussian noise with radius $\beta$ meters. First, we set $\beta = 30m$ for points sampled by a rate $r_3$, representing shifting for some points. Next, we set $\beta = r_4 \cdot 100m$ for all points, representing shifting for all points. Finally, we apply one kind of transform each

**Table 3.** Average MR of measures with dropping and shifting

| | $r_1$ | | | | $r_2$ | | | | $r_3$ | | | | $r_4$ | | | | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | Mean | Rank |
| FD | 1.04 | 1.10 | 1.36 | 2.84 | 1.02 | 1.14 | 1.42 | 2.25 | 1.20 | 1.25 | 1.20 | 1.20 | 1.18 | 1.40 | 1.40 | 1.74 | 1.42 | 5 |
| DTW | **1.00** | 1.08 | 1.70 | 5.66 | **1.00** | 1.01 | 1.08 | 1.22 | 1.05 | 1.21 | 1.26 | 1.37 | 1.31 | 1.71 | 2.12 | 2.38 | 1.64 | 6 |
| FastDTW | 20.38 | 24.36 | 18.98 | 28.56 | 20.51 | 23.21 | 16.66 | 22.89 | 42.99 | 42.02 | 41.61 | 41.02 | 42.26 | 41.93 | 41.27 | 41.27 | 31.87 | 11 |
| LCSS | 9.54 | 11.64 | 8.42 | 20.84 | 17.31 | 8.31 | 16.60 | 21.93 | 21.10 | 13.51 | 14.03 | 11.37 | 11.48 | 10.50 | 12.37 | 6.09 | 13.44 | 8 |
| EDR | **1.00** | **1.01** | 1.03 | 1.05 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 1.03 | 1.26 | 2.05 | 3.00 | **1.21** | **1** |
| ERP | 21.29 | 36.11 | 44.31 | 42.02 | 17.46 | 21.51 | 31.19 | 36.04 | 12.39 | 12.39 | 12.39 | 12.39 | 12.39 | 12.39 | 12.40 | 12.39 | 21.82 | 9 |
| OWD | 1.47 | 1.56 | 1.48 | 1.56 | 1.47 | 1.55 | 1.59 | 1.47 | **1.00** | 1.01 | 1.03 | 1.04 | **1.01** | 1.07 | **1.10** | **1.15** | 1.28 | 3 |
| CATS | 1.02 | 1.02 | **1.02** | **1.04** | 1.02 | 1.03 | 1.04 | 1.07 | 1.02 | 1.02 | 1.06 | 1.10 | 1.22 | 1.75 | 2.47 | 2.91 | 1.30 | 4 |
| BDS | 73.71 | 72.53 | 73.10 | 70.97 | 73.75 | 73.85 | 73.36 | 73.46 | 75.68 | 76.02 | 76.71 | 77.36 | 78.86 | 77.84 | 76.18 | 74.79 | 74.89 | 13 |
| SSPD | 5.41 | 5.66 | 6.01 | 9.32 | 4.64 | 5.51 | 5.14 | 5.99 | 5.19 | 6.14 | 7.20 | 7.05 | 9.48 | 8.17 | 7.43 | 7.48 | 6.61 | 7 |
| EDwP | 31.43 | 32.95 | 33.64 | 37.29 | 25.54 | 28.35 | 30.43 | 31.46 | 17.85 | 22.34 | 21.29 | 18.99 | 22.24 | 23.07 | 22.69 | 23.64 | 26.45 | 10 |
| STS | 32.99 | 32.71 | 32.51 | 31.44 | 33.25 | 32.97 | 32.64 | 32.21 | 32.90 | 32.53 | 32.47 | 32.19 | 32.34 | 32.06 | 31.71 | 31.41 | 32.40 | 12 |
| ITS | **1.00** | 1.30 | 1.96 | 2.19 | **1.00** | **1.00** | 1.42 | 1.21 | 1.04 | 1.04 | 1.04 | 1.04 | 1.03 | **1.06** | 1.13 | 1.16 | 1.23 | 2 |

time and set other rates to 0.

$$x'_i = x_i + \text{Gaussian}(0,1) \cdot \beta$$
$$y'_i = y_i + \text{Gaussian}(0,1) \cdot \beta$$

**Rank and Precision.** We denote $s_{ij'} = f(T_i, T'_j)$ as the similarity of $T_i \in \mathbb{T}$ and $T'_j \in \mathbb{T}'$, $f$ is a trajectory similarity measure function, $\gamma_i$ as the rank of $s_{ij'}$ in $(s_{i1'}, s_{i2'}, \ldots, s_{i|\mathbb{T}|'})$, $\chi_i$ as 1 if $\gamma_i = 1$ else 0, mean rank MR $= \frac{1}{|\mathbb{T}|}\sum_{i=1}^{|\mathbb{T}|}\gamma_i$ and mean precision MP $= \frac{1}{|\mathbb{T}|}\sum_{i=1}^{|\mathbb{T}|}\chi_i$. A smaller MR and a bigger MP indicate that measures can better recognize trajectories with noises.

**Cross-similarity.** We denote $s_{i'j'} = f(T'_i, T'_j)$ as the similarity of $T'_i \in \mathbb{T}'$ and $T'_j \in \mathbb{T}'$, cross similarity CS $= \frac{1}{|\mathbb{T}|^2}\sum_{i=1}^{|\mathbb{T}|}\sum_{j=1}^{|\mathbb{T}|}\frac{\min(|s_{ij}|,|s_{i'j'}|)}{\max(|s_{ij}|,|s_{i'j'}|)}$. The bigger CP is, the more robust measures recognize trajectory variants.

**$k$-nn queries.** We denote $\mathcal{N}_i$ as the $k$-nearest-neighbors ($k$-nn) of $T_i$ in $\mathbb{T}$, $\mathcal{N}'_i$ as the $k$-nn of $T'_i$ in $\mathbb{T}'$, $k$-nn precision KP $= \frac{1}{|\mathbb{T}|}\sum_{i=1}^{|\mathbb{T}|}\frac{|\mathcal{N}_i \cap \mathcal{N}'_i|}{k}$. A bigger KP indicates that measures can find similar neighbors with noises. $k$ is ten here.

**Clustering.** We execute K-Medoids on the ASL dataset and count how many trajectory pairs are classified into correct clusters. Unlike the above tasks evaluate the robustness of measures with dropping and shifting, clustering evaluates measures at a semantic scene. There are 95 clusters in the ASL dataset, and each cluster contains 27 trajectories. We select clusters randomly and evaluate measures with the various numbers of clusters. We denote $\varsigma_i$ as standard label of trajectory $T_i$, $\varsigma'_j$ as predicted label of trajectory $T'_j$, $\zeta_{ij} = 1$ if $\varsigma_i = \varsigma_j$ and $\varsigma'_i = \varsigma'_j$ else 0, clustering precision CP $= \frac{1}{|\mathbb{T}|^2}\sum_{i=1}^{|\mathbb{T}|}\sum_{j=1}^{|\mathbb{T}|}\zeta_{ij}$. A bigger clustering precision indicates that trajectories are classified correctly.

### 5.2 Performance

**Rank.** Table 3 shows the average results of the Rank task on three datasets. MR values increase as noises generally increase. EDR performs the best, yet it is not good at handling shifting on all points. ITS is just next to EDR and only

**Table 4.** Average MP of measures with dropping and shifting

| | $r_1$ | | | | $r_2$ | | | | $r_3$ | | | | $r_4$ | | | | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | Mean | Rank |
| FD | 0.93 | 0.85 | 0.79 | 0.67 | 0.95 | 0.92 | 0.86 | 0.79 | 0.94 | 0.93 | 0.91 | 0.91 | 0.94 | 0.89 | 0.88 | 0.84 | 0.87 | 6 |
| DTW | **1.00** | 0.95 | 0.83 | 0.48 | **1.00** | 0.99 | 0.95 | 0.89 | 0.97 | 0.92 | 0.90 | 0.89 | 0.89 | 0.83 | 0.79 | 0.77 | 0.88 | 5 |
| FastDTW | 0.71 | 0.58 | 0.70 | 0.52 | 0.71 | 0.64 | 0.73 | 0.62 | 0.50 | 0.52 | 0.53 | 0.54 | 0.53 | 0.53 | 0.53 | 0.54 | 0.59 | 8 |
| LCSS | 0.25 | 0.24 | 0.26 | 0.07 | 0.20 | 0.31 | 0.11 | 0.24 | 0.06 | 0.13 | 0.15 | 0.17 | 0.37 | 0.21 | 0.20 | 0.51 | 0.22 | 11 |
| EDR | **1.00** | **0.99** | 0.97 | 0.96 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 0.98 | 0.87 | 0.71 | 0.62 | 0.94 | 3 |
| ERP | 0.03 | 0.01 | 0.00 | 0.01 | 0.15 | 0.03 | 0.02 | 0.01 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.22 | 10 |
| OWD | 0.99 | 0.98 | **0.98** | 0.94 | 0.99 | 0.98 | 0.99 | 0.99 | **1.00** | 0.99 | 0.98 | 0.96 | **0.99** | 0.95 | **0.94** | 0.91 | **0.97** | **1** |
| CATS | 0.99 | 0.98 | **0.98** | **0.97** | 0.98 | 0.97 | 0.97 | 0.95 | 0.98 | 0.99 | 0.96 | 0.93 | 0.89 | 0.76 | 0.64 | 0.57 | 0.91 | 4 |
| BDS | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 13 |
| SSPD | 0.57 | 0.61 | 0.54 | 0.39 | 0.61 | 0.59 | 0.58 | 0.51 | 0.59 | 0.50 | 0.45 | 0.40 | 0.35 | 0.34 | 0.29 | 0.29 | 0.48 | 9 |
| EDwP | 0.11 | 0.09 | 0.08 | 0.01 | 0.18 | 0.11 | 0.12 | 0.12 | 0.21 | 0.21 | 0.21 | 0.25 | 0.26 | 0.22 | 0.24 | 0.20 | 0.16 | 12 |
| STS | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 7 |
| ITS | **1.00** | **0.99** | **0.98** | 0.90 | **1.00** | **1.00** | 0.98 | 0.94 | 0.98 | 0.97 | 0.97 | 0.97 | 0.98 | **0.97** | 0.93 | **0.92** | **0.97** | **1** |

**Table 5.** Average CS of measures with dropping and shifting

| | $r_1$ | | | | $r_2$ | | | | $r_3$ | | | | $r_4$ | | | | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | Mean | Rank |
| FD | **1.00** | **0.99** | **0.98** | **0.95** | **1.00** | **1.00** | **0.99** | **0.98** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.98 | 0.97 | 0.97 | **0.98** | **1** |
| DTW | 0.98 | 0.97 | 0.95 | 0.91 | 0.97 | 0.94 | 0.91 | 0.86 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.98 | 0.97 | 0.96 | 0.96 | 3 |
| FastDTW | 0.03 | 0.02 | 0.01 | 0.00 | 0.05 | 0.03 | 0.03 | 0.02 | 0.03 | 0.02 | 0.03 | 0.03 | 0.02 | 0.03 | 0.00 | 0.01 | 0.02 | 13 |
| LCSS | 0.69 | 0.62 | 0.60 | 0.28 | 0.67 | 0.54 | 0.61 | 0.62 | 0.89 | 0.90 | 0.75 | 0.66 | 0.68 | 0.62 | 0.64 | 0.68 | 0.65 | 6 |
| EDR | 0.68 | 0.54 | 0.42 | 0.29 | 0.77 | 0.61 | 0.48 | 0.37 | 0.73 | 0.57 | 0.47 | 0.39 | 0.41 | 0.33 | 0.29 | 0.26 | 0.48 | 9 |
| ERP | 0.82 | 0.82 | 0.81 | 0.79 | 0.85 | 0.82 | 0.81 | 0.81 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 0.91 | 5 |
| OWD | 0.98 | 0.96 | 0.93 | 0.89 | 0.99 | 0.98 | 0.96 | 0.94 | 0.95 | 0.94 | 0.93 | 0.93 | 0.94 | 0.91 | 0.89 | 0.87 | 0.94 | 4 |
| CATS | 0.69 | 0.46 | 0.30 | 0.14 | 0.78 | 0.64 | 0.50 | 0.36 | 0.63 | 0.44 | 0.31 | 0.24 | 0.26 | 0.18 | 0.15 | 0.13 | 0.39 | 10 |
| BDS | 0.76 | 0.53 | 0.33 | 0.13 | 0.88 | 0.77 | 0.66 | 0.59 | 0.77 | 0.62 | 0.51 | 0.45 | 0.46 | 0.41 | 0.38 | 0.37 | 0.54 | 8 |
| SSPD | 0.80 | 0.67 | 0.56 | 0.38 | 0.87 | 0.80 | 0.73 | 0.67 | 0.68 | 0.60 | 0.57 | 0.55 | 0.55 | 0.51 | 0.49 | 0.47 | 0.62 | 7 |
| EDwP | 0.32 | 0.21 | 0.14 | 0.10 | 0.40 | 0.31 | 0.24 | 0.21 | 0.29 | 0.24 | 0.20 | 0.18 | 0.19 | 0.15 | 0.15 | 0.12 | 0.21 | 12 |
| STS | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 11 |
| ITS | **1.00** | **0.99** | 0.96 | 0.93 | **1.00** | 0.99 | 0.98 | 0.95 | 0.98 | 0.98 | 0.98 | 0.97 | 0.98 | 0.97 | 0.96 | 0.95 | 0.97 | 2 |

**Table 6.** Average KP of measures with dropping and shifting

| | $r_1$ | | | | $r_2$ | | | | $r_3$ | | | | $r_4$ | | | | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 | Mean | Rank |
| FD | 0.98 | 0.96 | 0.91 | 0.84 | 0.99 | 0.98 | 0.96 | 0.91 | 0.97 | 0.96 | 0.96 | 0.95 | 0.96 | 0.94 | 0.92 | 0.91 | 0.94 | 2 |
| DTW | 0.96 | 0.95 | 0.92 | 0.84 | 0.96 | 0.92 | 0.87 | 0.82 | 0.99 | 0.98 | 0.98 | 0.98 | 0.98 | 0.97 | 0.94 | 0.93 | 0.94 | 2 |
| FastDTW | 0.73 | 0.66 | 0.71 | 0.64 | 0.56 | 0.75 | 0.61 | 0.64 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.82 | 6 |
| LCSS | 0.23 | 0.28 | 0.47 | 0.19 | 0.40 | 0.29 | 0.26 | 0.22 | 0.67 | 0.81 | 0.54 | 0.23 | 0.38 | 0.33 | 0.29 | 0.26 | 0.37 | 13 |
| EDR | 0.91 | 0.86 | 0.79 | 0.70 | 0.94 | 0.88 | 0.83 | 0.76 | 0.92 | 0.87 | 0.82 | 0.79 | 0.80 | 0.75 | 0.71 | 0.68 | 0.81 | 7 |
| ERP | 0.56 | 0.54 | 0.53 | 0.53 | 0.59 | 0.56 | 0.53 | 0.54 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 0.77 | 9 |
| OWD | 0.94 | 0.91 | 0.86 | 0.80 | 0.95 | 0.93 | 0.90 | 0.85 | 0.93 | 0.91 | 0.91 | 0.91 | 0.91 | 0.88 | 0.86 | 0.84 | 0.89 | 5 |
| CATS | 0.92 | 0.84 | 0.75 | 0.63 | 0.93 | 0.90 | 0.84 | 0.77 | 0.90 | 0.82 | 0.78 | 0.72 | 0.74 | 0.67 | 0.63 | 0.61 | 0.78 | 8 |
| BDS | 0.89 | 0.78 | 0.68 | 0.61 | 0.91 | 0.88 | 0.80 | 0.75 | 0.84 | 0.75 | 0.71 | 0.62 | 0.59 | 0.61 | 0.58 | 0.56 | 0.72 | 10 |
| SSPD | 0.69 | 0.57 | 0.49 | 0.40 | 0.78 | 0.69 | 0.60 | 0.57 | 0.56 | 0.44 | 0.41 | 0.34 | 0.35 | 0.32 | 0.34 | 0.32 | 0.49 | 11 |
| EDwP | 0.67 | 0.61 | 0.54 | 0.46 | 0.68 | 0.61 | 0.56 | 0.52 | 0.51 | 0.42 | 0.38 | 0.37 | 0.38 | 0.34 | 0.33 | 0.33 | 0.48 | 12 |
| STS | **1.00** | **1.00** | **1.00** | **0.98** | **1.00** | **1.00** | **1.00** | **0.96** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 0.99 | **1.00** | **0.99** | **1** |
| ITS | 0.98 | 0.96 | 0.92 | 0.87 | 0.99 | 0.97 | 0.94 | 0.91 | 0.96 | 0.95 | 0.95 | 0.94 | 0.95 | 0.93 | 0.92 | 0.90 | 0.94 | 2 |

0.02 worse than EDR. OWD is robust to shifting thanks to the grid but not to changing dropping rates inside the trajectory. ITS and FastDTW are $O(n)$ time complexity, but FastDTW is less robust than ITS.

**Precision.** Table 4 shows the average results of the Precision task on three datasets. MP values descend as noises generally increase. OWD and ITS perform

best. Similar to the results of the Rank task, EDR is not good at handling shifting on all points, and ITS performs better than FastDTW.

**Cross-similarity.** Table 5 shows the average results of the Cross-similarity task on three datasets. CS values descend as noises increase generally. FD is the best because dropping and shifting can hardly change the cross similarity of the shape itself, and shape-based OWD also performs well. ITS is just next to FD and only 0.01 worse than FD. ERP is robust to shifting when shifting is less than the threshold, which is the distance from points to the origin. However, ERP is not doing well with dropping.

**$k$-nn queries.** Table 6 shows the average $k$-nn precision task results on three datasets. KP values descend as noises generally increase. STS performs the best, and ITS is tied for second with FD and DTW. FastDTW restricts the matching window size, so it cannot match points out of the window when the drop rate gets bigger. Just like the results of the Cross-similarity task, ERP is robust to shifting but not good at dropping.

**Clustering.** Table 7 shows the results of the Clustering task on ASL. CP values increases as the number of clusters increase for some measures since the more clusters there are, the more semantic information in trajectories such measures can identify. Conversely, CP values descend for others since they cannot identify semantic information. ITS performs best and performs better as the number of clusters increases. SSPD is the second since it aims at clustering tasks.

**Runtime.** Table 8 shows the running time of measures among increasing $n$. The time complexity of ITS is $O(4n + 3h)$, and FastDTW is $O(n(8r + 14))$ where $r$ is the radius. We set $h$ and $r$ fixed here. ITS is the fastest measure. OWD grows slowly since the number of girds reaches almost max at the very first. ERP calculates the threshold dynamically, so it is slower than EDR. SSPD and BDS calculate distance twice to get symmetry, so they are the slowest.

**Table 7.** CP of measures

| #Clusters | 4 | 5 | 6 | 7 | Rank |
|---|---|---|---|---|---|
| FD | 0.60 | 0.53 | 0.65 | 0.73 | 8 |
| DTW | 0.62 | 0.68 | 0.70 | 0.73 | 5 |
| FastDTW | 0.26 | 0.20 | 0.17 | 0.14 | 13 |
| LCSS | 0.26 | 0.21 | 0.36 | 0.14 | 11 |
| EDR | 0.28 | 0.24 | 0.22 | 0.20 | 12 |
| ERP | **0.72** | 0.73 | 0.77 | 0.77 | 3 |
| OWD | 0.63 | 0.59 | 0.68 | 0.78 | 7 |
| CATS | 0.30 | 0.28 | 0.29 | 0.31 | 10 |
| BDS | 0.58 | 0.66 | 0.69 | 0.73 | 6 |
| SSPD | **0.72** | 0.77 | 0.76 | 0.78 | 2 |
| EDwP | 0.63 | 0.71 | 0.73 | 0.74 | 4 |
| STS | 0.53 | 0.59 | 0.53 | 0.63 | 9 |
| ITS | **0.72** | **0.79** | **0.83** | **0.83** | **1** |

**Table 8.** Runtime(s) of measures

| $n$ | 400 | 600 | 800 | 1000 | 1200 | Rank |
|---|---|---|---|---|---|---|
| FD | 0.779 | 1.906 | 3.055 | 4.617 | 6.576 | 9 |
| DTW | 0.707 | 1.601 | 2.941 | 4.517 | 6.389 | 8 |
| FastDTW | 0.023 | 0.030 | 0.045 | 0.052 | 0.061 | 2 |
| LCSS | 0.558 | 1.238 | 2.216 | 3.467 | 5.903 | 6 |
| EDR | 0.674 | 1.651 | 3.445 | 4.575 | 6.756 | 10 |
| ERP | 1.970 | 4.826 | 10.647 | 16.530 | 22.533 | 11 |
| OWD | 1.255 | 1.391 | 1.601 | 1.558 | 1.350 | 5 |
| CATS | 0.641 | 1.770 | 3.020 | 4.252 | 5.943 | 7 |
| BDS | 3.486 | 8.310 | 15.316 | 22.713 | 31.491 | 12 |
| SSPD | 4.668 | 11.725 | 20.697 | 30.231 | 42.594 | 13 |
| EDwP | 0.034 | 0.050 | 0.066 | 0.081 | 0.102 | 3 |
| STS | 0.509 | 0.702 | 0.958 | 1.203 | 1.508 | 4 |
| ITS | **0.019** | **0.026** | **0.035** | **0.041** | **0.047** | **1** |

**Summary.** To evaluate the overall performance of measures on both effectiveness of five downstream tasks and the efficiency of running time, we sum the rank of mean results on five tasks and the rank of running time. The smaller the sum is, the better the measure can keep both accuracy and efficiency. Table 9 shows that ITS has the best overall performance on the five downstream tasks

**Table 9.** Summary of measures on downstream tasks

| | Rank of Tasks | | | | | Rank of | Summary | |
|---|---|---|---|---|---|---|---|---|
| | MR | MP | CS | KP | CP | Runtime | Sum | Rank |
| FD | 5 | 6 | **1** | 2 | 8 | 9 | 31 | 4 |
| DTW | 6 | 5 | 3 | 2 | 5 | 8 | 29 | 3 |
| FastDTW | 11 | 8 | 13 | 6 | 13 | 2 | 53 | 10 |
| LCSS | 8 | 11 | 6 | 13 | 11 | 6 | 55 | 12 |
| EDR | **1** | 3 | 9 | 7 | 12 | 10 | 42 | 5 |
| ERP | 9 | 10 | 5 | 9 | 3 | 11 | 47 | 8 |
| OWD | 3 | **1** | 4 | 5 | 7 | 5 | 25 | 2 |
| CATS | 4 | 4 | 10 | 8 | 10 | 7 | 43 | 6 |
| BDS | 13 | 13 | 8 | 10 | 6 | 12 | 62 | 13 |
| SSPD | 7 | 9 | 7 | 11 | 2 | 13 | 49 | 9 |
| EDwP | 10 | 12 | 12 | 12 | 4 | 3 | 53 | 10 |
| STS | 12 | 7 | 11 | **1** | 9 | 4 | 44 | 7 |
| ITS | 2 | **1** | 2 | 2 | **1** | **1** | 9 | **1** |

and the running time. ITS is best on the Precision and Clustering tasks. Moreover, it is second on the Rank, Cross-similarity, and $k$-nn precision tasks. ITS is not the best on all tasks but is in the top 2 for all tasks, and other measures cannot achieve it. What is more, ITS is the fastest measure. ITS and FastDTW are both $O(n)$ time complexity, but FastDTW is less robust than ITS, and FastDTW is not as fast as ITS. Thus, ITS can keep both accuracy and efficiency.

### 5.3 Effect of $h$

Table 10 shows the running time in seconds(s) of ITS grows slowly as the $h$ increases while $n$ is fixed. However, the running times of $h = 10$ and $h = 100$ are close since the running time is dominated by $n$ when $h < n$.

Figure 3 shows that ITS performs better as $h$ grows generally, but performances grow slower as $h$ increases. The Y-axis indicate the results of downstream tasks. Let $m = \frac{\sum_{T \in \mathbb{T}} |T|}{|\mathbb{T}|}$ be the mean numbers of points of trajectories. Setting $h = m$ can obtain almost the best results within a very short time.

**Table 10.** Runtime(s) of ITS with various $h$

| $h$ | 10 | $10^2$ | $10^3$ | $10^4$ | $10^5$ |
|---|---|---|---|---|---|
| Runtime(s) | 0.035 | 0.035 | 0.041 | 0.083 | 0.514 |

## 6 Conclusion

In this paper, we revisited the problem of measuring trajectory similarity effectively and efficiently. The problem is challenging due to the ubiquitous shifting in trajectory data and variable dropping rates of different objects. In addition, most existing methods have a quadratic running time, which is good but provides enough room for improvement. Motivated by these observations, we proposed ITS, an effective and efficient trajectory measure. The key idea of the measure is

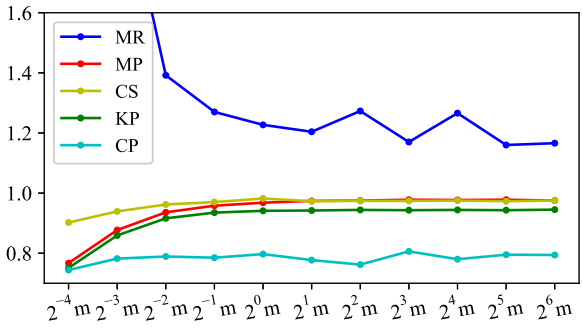


**Fig. 3.** Performances of ITS with various $h$

a fast interpolation that transforms trajectories into vectors without losing the semantics. The effectiveness of our measure results from the well-defined properties of interpolation: trajectories are aligned, the points in a trajectory are distributed evenly, and the shifting can only disturb surrounding interpolated points. Furthermore, the efficiency of our measure is guaranteed by fast interpolation, which is linear time, and fast computation of the distance of vectors, which is linear time too. We conducted extensive experiments with 12 baselines on four real-world datasets. The results showed that ITS is the fastest measure and has the best overall performance on a series of downstream tasks that are important in trajectory computing.

## Acknowledgements

This work is supported by Natural Science Foundation of Jiangsu Province (Grant Nos. BK20211307), and by project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

## References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD. pp. 359–370 (1994)
2. Besse, P.C., Guillouet, B., Loubes, J., Royer, F.: Review and perspective for distance-based clustering of vehicle trajectories. IEEE Trans. Intell. Transp. Syst. **17**(11), 3306–3317 (2016)
3. Ceikute, V., Jensen, C.S.: Vehicle routing with user-generated trajectory data. In: MDM. pp. 14–23 (2015)
4. Chen, L., Ng, R.T.: On the marriage of lp-norms and edit distance. In: VLDB. pp. 792–803 (2004)
5. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD. pp. 491–502 (2005)
6. Hung, C., Peng, W., Lee, W.: Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. VLDB J. **24**(2), 169–192 (2015)

7. Ismail, A., Vigneron, A.: A new trajectory similarity measure for gps data. In: IWGS 2015. pp. 19–22 (2015)
8. Kadous, M.W.: Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. Ph.D. thesis (2002)
9. Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: KDD. pp. 285–289 (2000)
10. Li, G., Hung, C., Liu, M., Pan, L., Peng, W., Chan, S.G.: Spatial-temporal similarity for trajectories with location noise and sporadic sampling. In: ICDE. pp. 1224–1235 (2021)
11. Li, R., He, H., Wang, R., Huang, Y., Liu, J., Ruan, S., He, T., Bao, J., Zheng, Y.: JUST: JD urban spatio-temporal data engine. In: ICDE. pp. 1558–1569 (2020)
12. Li, X., Zhao, K., Cong, G., Jensen, C.S., Wei, W.: Deep representation learning for trajectory similarity computation. In: ICDE 2018. pp. 617–628 (2018)
13. Lin, B., Su, J.: Shapes based trajectory queries for moving objects. In: ACM-GIS. pp. 21–30 (2005)
14. Luo, Y., Li, W., Zhao, T., Yu, X., Zhang, L., Li, G., Tang, N.: Deeptrack: Monitoring and exploring spatio-temporal data: A case of tracking covid-19. Proc. VLDB Endow. **13**(12), 2841–2844 (2020)
15. Magdy, N., Sakr, M.A., Mostafa, T., El-Bahnasy, K.: Review on trajectory similarity measures. In: ICICIS. pp. 613–619 (2015)
16. Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., Damas, L.: Predicting taxi–passenger demand using streaming data. IEEE Trans. Intell. Transp. Syst. **14**(3), 1393–1402 (2013)
17. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. J. Intell. Inf. Syst. **27**(3), 267–289 (2006)
18. Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsi, I., Andrienko, G.L., Theodoridis, Y.: Similarity search in trajectory databases. In: TIME 2007. pp. 129–140 (2007)
19. Ranu, S., P, D., Telang, A.D., Deshpande, P., Raghavan, S.: Indexing and matching trajectories under inconsistent sampling rates. In: ICDE 2015. pp. 999–1010 (2015)
20. Rohani, M., Gingras, D., Gruyer, D.: A novel approach for improved vehicular positioning using cooperative map matching and dynamic base station dgps concept. IEEE Trans. Intell. Transp. Sysy. **17**(1), 230–239 (2016)
21. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007)
22. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Trajectory similarity join in spatial networks. Proc. VLDB Endow. **10**(11), 1178–1189 (2017)
23. Su, H., Liu, S., Zheng, B., Zhou, X., Zheng, K.: A survey of trajectory distance measures and performance evaluation. VLDB J. **29**(1), 3–32 (2020)
24. Ta, N., Li, G., Xie, Y., Li, C., Hao, S., Feng, J.: Signature-based trajectory similarity join. IEEE Trans. Knowl. Data Eng. **29**(4), 870–883 (2017)
25. Vlachos, M., Gunopulos, D., Das, G.: Rotation invariant distance measures for trajectories. In: KDD. pp. 707–712 (2004)
26. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE 2002. pp. 673–684 (2002)
27. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: ACM-GIS 2010. pp. 99–108 (2010)
28. Zhang, H., Zhang, X., Jiang, Q., Zheng, B., Sun, Z., Sun, W., Wang, C.: Trajectory similarity learning with auxiliary supervision and optimal matching. In: IJCAI. pp. 3209–3215 (2020)
29. Zheng, Y., Xie, X., Ma, W.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–39 (2010)