Journal of Artificial Intelligence Research 79 (2024) 865-893 Submitted 12/2023; published 03/2024

# A Principled Distributional Approach to Trajectory Similarity Measurement and its Application to Anomaly Detection

**Yufan Wang** WANGYF@LAMDA.NJU.EDU.CN
**Zijing Wang** WANGZJ@LAMDA.NJU.EDU.CN
**Kai Ming Ting** TINGKM@NJU.EDU.CN
**Yuanyi Shang** SHANGYY@LAMDA.NJU.EDU.CN
*National Key Laboratory for Novel Software Technology,*
*Nanjing University, China*
*School of Artificial Intelligence,*
*Nanjing University, China*

## Abstract

This paper aims to solve two enduring challenges in existing trajectory similarity measures: computational inefficiency and the absence of the 'uniqueness' property that should be guaranteed in a distance function: $dist(X,Y) = 0$ if and only if $X = Y$, where $X$ and $Y$ are two trajectories. In this work, we present a novel approach utilizing a distributional kernel for trajectory representation and similarity measurement, based on the kernel mean embedding framework. It is the very first time a distributional kernel is used for trajectory representation and similarity measurement. Our method does not rely on point-to-point distances which are used in most existing distances for trajectories. Unlike prevalent learning and deep learning approaches, our method requires no learning. We show the generality of this new approach in anomalous trajectory and sub-trajectory detection. We identify that the distributional kernel has (i) a data-dependent property and the 'uniqueness' property which are the key factors that lead to its superior task-specific performance, and (ii) runtime orders of magnitude faster than existing distance measures.

## 1. Introduction

The advancement of location technology has resulted in the generation of a large amount of trajectory data, causing an increase in the interest in trajectory mining. A trajectory has space-time continuity and is often represented by a sequence of location points representing the route of a moving object. Potential benefits of trajectory mining are in urban planning, transportation management, and public safety.

Like other data mining tasks, similarity measurements for trajectories are a core operation in trajectory data mining. Commonly used measures include Dynamic Time Warping (DTW) (Sakoe & Chiba, 1971), Hausdorff distance (Rockafellar & Wets, 1998), Fréchet distance (Eiter & Mannila, 1994) and edit distances (Chen & Ng, 2004; Chen et al., 2005). However, these existing measures have three fundamental shortcomings:

1. Existing trajectory distance measures are computationally expensive. They have a high time complexity of $O(m^2)$ in making a measurement between two trajectories, each having $m$ data points.

2. None of the existing measures can guarantee the 'uniqueness' property of a distance function: $dist(X, Y) = 0$ if and only if $X = Y$, where $X$ and $Y$ are two trajectories (Li et al., 2018; Ma et al., 2018). As a result, they may produce some 'irregularities' such as two different trajectories having a small distance or two similar trajectories having a large distance. Despite recent efforts that produce learned measure/representation (Li et al., 2018; Ma et al., 2018), they still could not guarantee to have gotten rid of this kind of 'irregularities'. Even deep learning methods (Liu et al., 2020; Malhotra et al., 2016; Xu et al., 2021) do not provide any improvement on this front.

3. Traditional measures are all based on point-to-point distances between two trajectories and ignore the distribution information in a trajectory, which can lead to an unreasonable result in some cases (see Section 5.2 for details).

We are motivated to address these shortcomings using a principled approach, which has three distinctive features in comparison with the above-mentioned approaches. Firstly, a distributional kernel is used for the very first time to represent trajectories and measure the similarity between two trajectories. Secondly, it does not rely on point-to-point distances. Thirdly, it requires no learning, yet it performs as well as, or better than, existing measures and deep learning methods.

Our contributions are:

1. Introducing a distributional kernel for trajectory representation and similarity measurement. Our method has two benefits: (i) linear time complexity $O(m)$ for measuring two trajectories with a maximum length of $m$; (ii) a strong theoretical underpinning based on kernel mean embedding (Muandet et al., 2017; Smola et al., 2007).

2. Analyzing the essential properties of a good measure for trajectories and identifying the importance of the data-dependent property of a measure for applications such as anomalous trajectory detection.

3. Proposing simple and effective algorithms for anomalous trajectory and sub-trajectory detection, based on the powerful distributional kernel. We show for the first time that the distributional measure can be applied successfully to four existing point anomaly detectors to detect anomalous trajectories.

4. Conducting empirical evaluations in two different applications to assess the effectiveness and efficiency of (i) different trajectory distance measures, including the proposed distributional kernel, three commonly used distance measures and one representation learning method, and (ii) the proposed algorithms in comparison with existing algorithms in anomaly detection tasks.

The rest of the paper is organized as follows. Section 2 reviews existing similarity measures for trajectories. Problem formulation is given in Section 3. Section 4 introduces how trajectories could be treated as distributions and how distributional kernel could be used to measure the similarity between trajectories. Section 5 discusses the necessary properties of a similarity measure for trajectories. The proposed distributional kernel-based algorithms for two applications are provided in Section 6. The empirical settings and evaluations are reported in Sections 7 & 8, followed by the discussion and conclusion sections.

## 2. Related Work

In this section, we survey existing trajectory distance measures and trajectory anomaly detection methods.

### 2.1 Trajectory Similarity/Distance Measures

Existing trajectory similarity/distance measures can be divided into three categories: traditional measures, tailored measures, and representation learning.

Examples of traditional measures include Dynamic Time Warping (DTW) distance (Sakoe & Chiba, 1971), Hausdorff distance (Rockafellar & Wets, 1998), Fréchet distance (Eiter & Mannila, 1994), edit distances (Chen & Ng, 2004; Chen et al., 2005), and longest common subsequence distance (Vlachos et al., 2002). Most of these measures compute their final distances between two trajectories based on some best-match point-pairs from the two trajectories. Dynamic programming is often used to find the matched pairs based on some criterion to determine the goodness of a match. High time complexity is the key limitation of these measures. Note that some of these measures are set-based measures, e.g., Hausdorff and Fréchet distances, where time/order information is ignored.

Tailored measures are motivated by the fact that existing distance measures such as DTW, Hausdorff, and Fréchet distances have some irregularities in measuring the distance between trajectories (Li et al., 2018; Ma et al., 2018). Constructing a tailored measure often involves a process to learn the order/time-dependent structure in a dataset of trajectories. An example method enlists RNN Autoencoder to learn a tailored measure (Ma et al., 2018) by minimizing the reconstruction errors between the input sequences and the constructed sequences. Another recent deep learning work ST2Vec (Fang et al., 2022) aims to speed up the computation of an existing distance measure such as Hausdorff and Fréchet distances by learning an approximate measure.

The third category is representation learning which aims to learn a vector representation of trajectories via some transformation. Examples are deep representation learning (Li et al., 2018), tube-droplet method (Lin et al., 2017), time-sensitive Dirichlet process mixture model (Hu et al., 2013) and hidden Markov model (Morris & Trivedi, 2011). We refer the readers to a recent survey of the representation and measures used for trajectories (Sousa et al., 2020) for further details.

It is interesting to note that none of these existing measures and learned methods have been shown to have the uniqueness property, i.e., $dist(X, Y) = 0$ if and only if $X = Y$. The uniqueness property is one of the four axioms of a distance metric. We postulate that the irregularities identified are largely due to the lack of this property.

In summary, traditional distance measures are computationally expensive and can have certain irregularities that affect the accuracy of the measurements. Although some learning techniques try to balance effectiveness with efficiency, they only address the runtime problem. In fact, they exacerbate the effectiveness issue because they only learn an approximation of the intended measure.

### 2.2 Trajectory Anomaly Detection Methods

To deal with a specific data mining task with trajectories, any existing point-based methods, which employ a distance/kernel function, can be used directly with a minimal change. For example, to deal with trajectory anomaly detection, existing anomaly detectors such as LOF (Breunig et al., 2000) and OCSVM (Schölkopf et al., 2001) can be employed directly to detect anomalous trajectory by simply replacing the distance/kernel function used with any of the above-mentioned measures.

Other methods include probabilistic model DB-TOD (Wu et al., 2017), isolation-based methods iBAT (Zhang et al., 2011) and iBOAT (Chen et al., 2013), and the partition-and-detect method TRAOD (Lee et al., 2008). With the exception of TRAOD, no public source codes are made available on these systems.

With the development of deep learning, many task-specific end-to-end approaches have been proposed. Generally, end-to-end deep methods involve the learning of trajectory representations. For example, ATD-RNN (Song et al., 2018) is a supervised method based on RNN. IGMM-GAN (Gray et al., 2018) combines a Gaussian mixture model with GAN. Through the estimation of a generative probability density on the space of trajectories, IGMM-GAN generates realistic synthetic datasets and simultaneously facilitates multimodal anomaly detection. The semi-supervised GM-VSAE (Liu et al., 2020) first converts each trajectory to a series of tokens, and then employs RNN. EncDec-AD (Malhotra et al., 2016) combines LSTM with an autoencoder. It uses the reconstruction error as the anomaly score, assuming that normal data is easier to reconstruct than anomalous data.

## 3. Problem Formulation

In this section, we give the definition of trajectory and sub-trajectory.

**Definition 3.1** ***Trajectory*** $X$ *is an ordered sequence of points, i.e.,* $X = \ulcorner x_1, \ldots, x_i, \ldots, x_\mu \urcorner$, *where* $i \in [1, \mu]$ *indicates the order of traversal in* $X$.

In practice, $x_i \in X$ is usually a GPS point having three attributes: longitude, latitude, and timestamp.

**Definition 3.2** ***Sub-trajectory*** $X_s = \ulcorner x_a, \ldots, x_b \urcorner$ *is a contiguous subsequence of* $X$*, denoted as* $X_s \prec X$*, where* $1 \leq a \leq b \leq \mu$.

Note that $\ulcorner x_i \urcorner, i = 1, \ldots, \mu$ are also sub-trajectories of $X$. We denote this special case of $\ulcorner x_i \urcorner$ as a *point-sub-trajectory*.

**Definition 3.3** ***Maximal sub-trajectory*** $X_s \prec X$ *is a contiguous subsequence in* $X$ *of maximal length if* $x_{a-1}$ *and* $x_{b+1}$ *cannot be valid members of* $X_s = \ulcorner x_a, \ldots, x_b \urcorner$ *based on some criterion.*

A trajectory may contain multiple maximal sub-trajectories separated by invalid members based on some criterion.

The problems of the two applications we considered in this paper are defined as follows:

**Definition 3.4** ***Anomalous trajectory detection.*** *Given a dataset $D = \{X_i, i = 1, \dots, n\}$, anomalous trajectory detection aims to detect trajectories in $D$ which are rare and different from the majority.*

**Definition 3.5** ***Anomalous sub-trajectory detection.*** *Given an anomalous trajectory $X$, anomalous sub-trajectory detection aims to detect all maximal sub-trajectories $X_s \prec X$ that make $X$ anomalous with respect to a given dataset of trajectories.*

Table 1 shows the key notations used in this paper.

| | |
|---|---|
| $x$ | A point in $\mathbf{d}$-dimensional real domain $\mathbb{R}^{\mathbf{d}}$ |
| $\kappa$ | Isolation/Gaussian kernel |
| $\phi$ | Kernel map of $\kappa$ |
| $X$ | A trajectory of $\ulcorner x_1, \dots, x_\mu \urcorner$ with $\lvert X\rvert = \mu$ points |
| $\mathcal{P}_X$ | Probability distribution that generates $x \sim \mathcal{P}_X$ |
| $\mathcal{K}_I$ or $\mathcal{K}_G$ | Isolation/Gaussian Distributional Kernel |
| $\Phi$ | Kernel mean map of $\mathcal{K}_I$ or $\mathcal{K}_G$ |
| $\mathbf{g}$ | Mapped point $\mathbf{g} = \Phi(\mathcal{P}_X)$ in Hilbert space $\mathscr{H}$ |
| $D$ | Set of $n$ trajectories $\{X_i, i = 1, \dots, n\}$ |
| $\Pi$ | Set of mapped points $\{\mathbf{g}_i, i = 1, \dots, n\}$ in $\mathscr{H}$ |
| $\mathrm{F}_I$ or $\mathrm{F}_G$ | Detector F employing $\Phi$ derived from $\mathcal{K}_I$ or $\mathcal{K}_G$ |
| $d_W$, $d_H$, $d_F$ | DTW, Hausdorff, Fréchet distances |

Table 1: Key symbols and notations used

## 4. Distributional Kernel for Trajectory Representation and Similarity Measurement

In this section, we show that trajectories can be treated as independent and identically distributed (i.i.d.) points sampled from an unknown probability distribution. Therefore, it is possible to represent a trajectory via a distributional kernel.

### 4.1 Assumptions and Intuitive Examples

To use a distributional kernel to measure the similarity between two trajectories, we make the following assumptions:

1. **Valid trajectories**: A valid trajectory consists of a series of ordered points that obey the constraints in time and space.

2. **Independent and identically distributed (i.i.d.) assumption**: A valid trajectory $X$ is assumed to be an i.i.d. sample set generated from an unknown probability distribution $\mathcal{P}_X$.

3. **Time is regarded to be one of the dimensions in $\mathbb{R}^{\mathbf{d}}$**: The time information (or order) of points in each trajectory can be included as a dimension, in addition to a spatial $\mathbf{d} - 1$ dimensional space.

With the above assumptions, all points in a valid trajectory $X$ can be seen as i.i.d. points $x \in \mathbb{R}^{\mathbf{d}}$ which are generated from an unknown probability distribution function (pdf) $\mathcal{P}_X$, i.e., $x \sim \mathcal{P}_X$, and a distributional kernel can effectively compute the similarity between two trajectories.

Here, we provide an intuitive example to show that trajectories can be represented by distributions and that the representation results may differ based on whether or not temporal information is taken into account. As shown in Table 2, suppose there is a point moving from the origin in a one-dimensional coordinate system. By recording its distance from the origin at each moment, we can create a one-dimensional trajectory. For simplicity, we assume that the velocity of the point is constant within the trajectory. Additionally, we record all trajectories at a constant sampling rate.

If we do not consider temporal information, the distribution of the trajectory would be one-dimensional, i.e., $x \in \mathbb{R}^1$. However, if we consider temporal information, the sample points from the trajectory would be two-dimensional, i.e., $(t, x_t) \in \mathbb{R}^2$. Assume there are three trajectories, $X, X'$, and $Y$. Both $X$ and $X'$ start from the origin, with different speeds, travel to a location near 200 meters, and then return to the origin. $Y$ moves with a uniform speed from the origin to a distance of 500 meters from the origin. Each column in Table 2 shows the trajectories in different dimensions and their corresponding distributions. We discuss the case of considering the temporal domain and not considering the temporal domain separately below:

- $x \in \mathbb{R}^1$: When the time information is ignored, the pdfs of $X$ and $X'$ are approximately the same, as shown in the first column of Table 2. As a result, a distributional kernel $\mathcal{K}$ that measures the similarity between these two trajectories yields $\mathcal{K}(X, X') \approx 1$ because $\mathcal{P}_X \approx \mathcal{P}_{X'}$. In contrast, the similarity between $X$ and $Y$ yields $\mathcal{K}(X, Y) < \mathcal{K}(X, X')$ because $\mathcal{P}_X \neq \mathcal{P}_Y$.

- $(t, x_t) \in \mathbb{R}^2$, where time is one of the two dimensions shown in the second column of Table 2. Here $X$, $X'$, and $Y$ have different distributions, i.e., $\mathcal{P}_X \neq \mathcal{P}_{X'} \neq \mathcal{P}_Y$, Which means a distributional kernel would not consider $X$ similar to $X'$.

Note that the above two representations can both be legitimate cases in practice. In applications where time is irrelevant in identifying a unique trajectory, it is unnecessary to include the time domain in the representation.

## 4.2 Distributional Kernel

A distributional kernel measures the similarity between two distributions. Let $X$ and $Y$ be two sets of i.i.d. samples generated from two distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively. Based on kernel mean embedding (KME) (Muandet et al., 2017), a distributional kernel is defined as follows:

$$\mathcal{K}_G(\mathcal{P}_X, \mathcal{P}_Y) = \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} \kappa(x, y) \tag{1}$$

While $\kappa$ is typically a Gaussian kernel in the KME framework (Muandet et al., 2017), a recent work (Ting et al., 2020) has shown that using Isolation Kernel (IK) (Ting et al., 2018) is a better option for point anomaly detection.

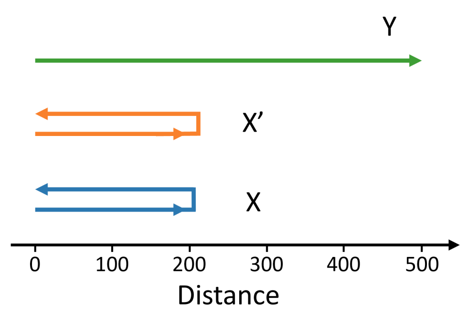
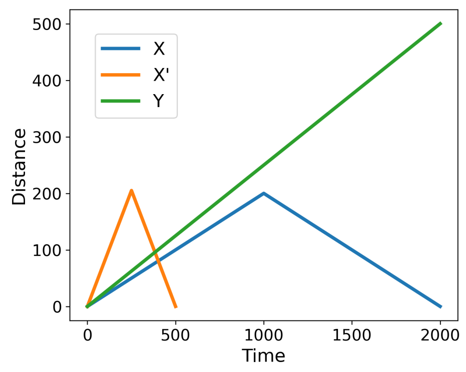
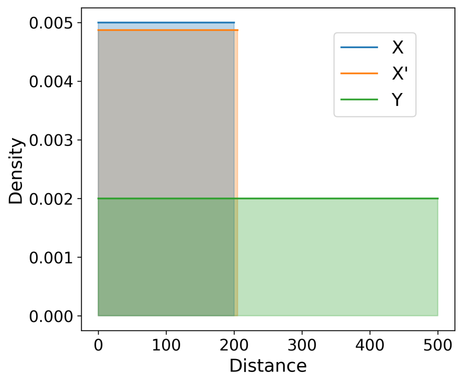
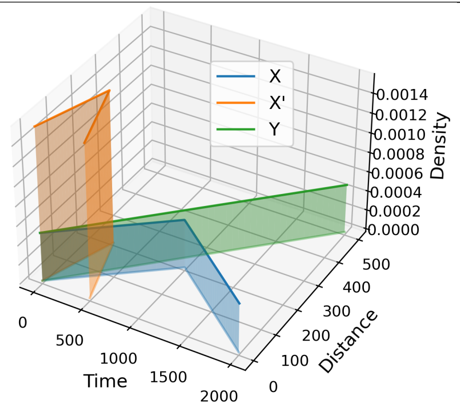
| | $x \in \mathbb{R}^1$ | $(t, x_t) \in \mathbb{R}^2$ |
|---|---|---|
| Trajectories |  Y<br>X'<br>X<br>0<br>100<br>200<br>300<br>400<br>500<br>Distance |  |
| Distributions |  |  |

Table 2: First row: three trajectories in 1-dimensional spatial space $\mathbb{R}^1$ and 2-dimensional spatiotemporal space $\mathbb{R}^2$. Second row: pdfs in $\mathbb{R}^1$ and $\mathbb{R}^2$ spaces.

A new distributional kernel (Ting et al., 2020) constructed by replacing the Gaussian kernel with IK in KME is briefly described as follows.

Given a dataset $\mathsf{D} \subset \mathbb{R}^{\mathbf{d}}$, IK derives a finite-dimensional feature map $\phi$ from $\mathsf{D}$, i.e., $\kappa(x, y|\mathsf{D}) = \langle \phi(x|\mathsf{D}), \phi(y|\mathsf{D}) \rangle$. Then, Eq (1) can be re-expressed as:

$$\begin{aligned} \mathcal{K}_I(\mathcal{P}_X, \mathcal{P}_Y|\mathsf{D}) =& \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} \kappa(x, y|\mathsf{D}) \\ =& \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} \langle \phi(x|\mathsf{D}), \phi(y|\mathsf{D}) \rangle \\ =& \langle \Phi(\mathcal{P}_X|\mathsf{D}), \Phi(\mathcal{P}_Y|\mathsf{D}) \rangle , \end{aligned} \tag{2}$$

The kernel mean map $\Phi$ of $\mathcal{K}_I$, which maps a distribution estimated by a sample set $X$ in input space to a point in Hilbert space, is given as:

$$\Phi(\mathcal{P}_X|\mathsf{D}) = \frac{1}{|X|} \sum_{x \in X} \phi(x|\mathsf{D}). \tag{3}$$

Our approach is distinguished from other existing measures in three key aspects:

1. A distributional kernel is used to represent each trajectory and measure the similarity between two trajectories, without learning. In contrast, existing works focus on some point-to-point distance-based measures or learned representations.
2. The proposed distributional approach inherits the theoretical fundamentals of kernel mean embedding. Specifically, the resultant representation $\Phi : \mathbb{P} \rightarrow \mathscr{H}$ (Hilbert space) is injective, i.e., $\| \Phi(\mathcal{P}_X) - \Phi(\mathcal{P}_Y) \|_{\mathscr{H}} = 0$ iff $\mathcal{P}_X = \mathcal{P}_Y$, where $\mathcal{P}_X, \mathcal{P}_Y \in \mathbb{P}$ and $\mathbb{P}$ is a set of probability distributions on $\mathbb{R}^{\mathbf{d}}$ (Fukumizu et al., 2004). Note that this is equivalent to the uniqueness property of a measure (without mapping $\Phi$) mentioned earlier. None of the existing measures and representation learning methods have been shown to have this property.
3. An implementation of the approach called Isolation Distributional Kernel ($\mathcal{K}_I$) has a unique data-dependent property: **two distributions are more similar to each other when measured by $\mathcal{K}_I$ derived from a sparse region than that from a dense region** (Ting et al., 2020). It enables $\mathcal{K}_I$-based detector to gain higher detection accuracy than existing deep learning methods. Furthermore, this implementation is more efficient than traditional distance-based methods.

We will discuss these important properties of similarity measures in more detail in the next section.

## 5. Important Properties of Trajectory Similarity Measures

Table 3 presents four important properties of any measures for trajectories: uniqueness, point-to-point distance-based, distribution-based, and data-dependent. We discuss the first three properties in the next subsection and the details of the fourth property in the following subsection.

| | $\mathcal{K}_I$ | $\mathcal{K}_G$ | $d_W$ | $d_H$ | $d_F$ |
|---|---|---|---|---|---|
| Uniqueness property | ✓ | ✓ | × | × | × |
| Point-to-point distance-based | × | ✓ | ✓ | ✓ | ✓ |
| Distribution-based | ✓ | ✓ | × | × | × |
| Data-dependent | ✓ | × | × | × | × |
| Time complexity | $m+n$ | | $mn$ | $mn$ | $mn \log(mn)$ |

Table 3: Compliance with properties listed above of a trajectory similarity measure. Time complexity is for computing the similarity of two trajectories with size $m$ and $n$, respectively. $\mathcal{K}_I$ and $\mathcal{K}_G$ have the same time complexity $m+n$.

### 5.1 Uniqueness, Point-to-point Distance-based and Distribution-based Properties

The uniqueness property is one of the four axioms of a distance metric: $dist(X, Y) = 0$ if and only if $X = Y$, where $X$ and $Y$ are two trajectories. An example with three trajectories is provided in Table 4 to examine whether the five measures comply with this property, where $X'$ is a translated version of $X$, $Y$ is a different trajectory from either $X$ and $X'$.

| | | $(X, X')$ | $(X, Y)$ | $d(X, X') < d(X, Y)$ |
|---|---|---|---|---|
| I | $d_W$ | .50 | .43 | × |
| | $d_H$ | .50 | .50 | × |
| | $d_F$ | .50 | .50 | × |
| II | $\mathcal{K}_I$ | .48 | .23 | ✓ |
| | $\mathcal{K}_G$ | .45 | .27 | ✓ |

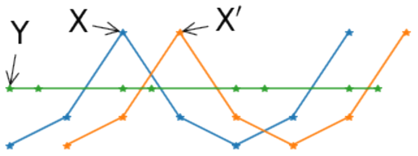


Table 4: An example of unreasonable results produced by the three distance measures $d_W$, $d_H$ & $d_F$, in sharp contrast with distributional kernels $\mathcal{K}_I$ and $\mathcal{K}_G$. The first row presents distances, while the second row presents similarities. $X$, $Y$ and $X'$ are trajectories or sequences of two-dimensional GPS points.

In this example, $d_W$, $d_H$ and $d_F$ produce distances for $X$ and $Y$ which are equal to or smaller than the distances for $X$ and $X'$. The reason for this unreasonable result is that $d_W$, $d_H$ and $d_F$ are based on point-to-point distances that consider point-to-point matching only without considering the distribution of the points.

In contrast, $\mathcal{K}_I$ and $\mathcal{K}_G$ produce measurements that are consistent with the uniqueness property because they are both based on kernel mean embedding which has been shown theoretically to have this property (Muandet et al., 2017; Ting et al., 2020).

## 5.2 Data-dependent Property of $\mathcal{K}_I$

Table 3 shows that only $\mathcal{K}_I$ has the data-dependent property among all the five measures. Given a trajectory dataset $D = \{X_i, i = 1, \dots, n\}$, the data-dependent similarity measure $\mathcal{K}_I$ is derived from $\mathsf{D} = \cup_{i=1}^{n} X_i$ (Ting et al., 2020). $\mathcal{K}_I$ relies on local neighborhoods which are large in the sparse region and small in the dense region. This leads directly to a unique data-dependent property (Ting et al., 2020):

**Definition 5.1** ***Data-dependent property of*** $\mathcal{K}_I$. *Two distributions are more similar to each other when measured by* $\mathcal{K}_I$ *derived from a sparse region than that from a dense region.*

This property is inherited from Isolation Kernel (IK) which has a similar data-dependent property (Qin et al., 2019; Ting et al., 2018), as $\mathcal{K}_I$ (Ting et al., 2020) is built based on IK. The other four measures in Table 3 do not have the data-dependent property because they all use a data-independent measure such as Euclidean distance or Gaussian kernel.

Two scenarios, in which a data-dependent kernel such as IK has a significant impact on detection accuracy, are presented below.

### 5.2.1 Trajectories in Dense and Sparse Clusters

The example dataset, shown in Figure 1, consists of 103 trajectories, with one normal dense cluster (top 50 trajectories), one normal sparse cluster (bottom 50 trajectories), and three anomalous trajectories $X'$, $Y'$, and $Z'$. Anomalous trajectories are all straight-line, whereas the normal trajectories in the two clusters are not straight-line trajectories.

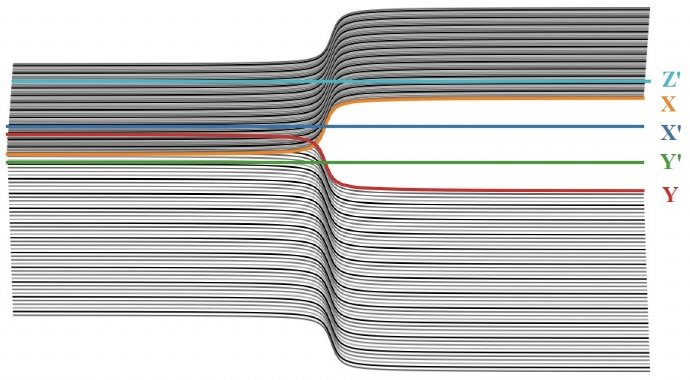


Figure 1: An example dataset. The trajectories are indexed, from #0 to #102 from top to bottom on the right. Three anomalous trajectories $Z'$, $X'$, and $Y'$, which have indices at #40, #51, #52 respectively. On the right half of the trajectories, each pair of $X$ & $X'$ and $Y$ & $Y'$ has the same (vertical) Euclidean distance.

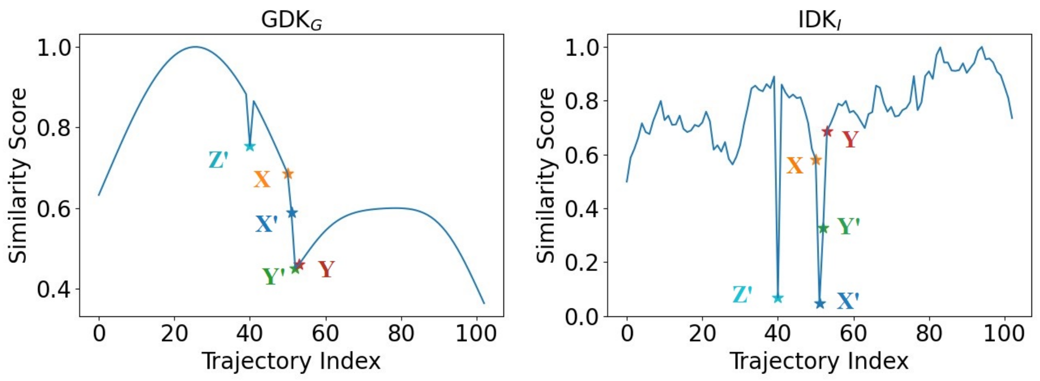


Figure 2: Similarity scores from $\text{GDK}_G$ and $\text{IDK}_I$ on trajectories shown in Figure 1.

As illustrated in Figure 2, $\text{GDK}_G$ exhibits a distribution of similarity scores that is reminiscent of a density distribution, where trajectory indices #0 to #50 are members of the dense cluster, and indices #53 to #102 are members of the sparse cluster. The three anomalous trajectories $Z'$, $X'$ and $Y'$ (indices #40, #51 & #52) have similarity scores close to the edges of the dense cluster. In addition, all trajectories in the sparse cluster have similarity scores less than the dense cluster, and the trajectory with the lowest similarity score is at index #102 (the bottom trajectory in Figure 1). As a result, almost all the normal trajectories in the sparse cluster have lower similarity scores than the two anomalous trajectories $Z'$, and $X'$. Thus, $Z'$ and $X'$ cannot be identified as anomalous trajectories by GDK$_G$.

In contrast, the distribution of the similarity score of IDK$_I$ is more balanced between the dense cluster and sparse cluster in Figure 2, and all three anomalous trajectories have the lowest similarity scores. Thus, $Z'$, $Y'$, and $X'$ are easily identified as anomalous trajectories by IDK$_I$. **This scenario indicates that the data-dependent $\mathcal{K}_I$ is a better measure than the data-independent $\mathcal{K}_G$ for trajectory anomaly detection in datasets containing regions of varied densities**.

### 5.2.2 Sheepdogs Trajectories

Here we use a real-world example, the *Sheepdogs* dataset, which differs from the previous example in many aspects. The dense and sparse regions in *Sheepdogs* have a significant impact on the detection accuracy of a detector that relies on a point-to-point distance measure. Figure 3 shows a total of 538 trajectories, of which 23 are trajectories of sheep and the rest are sheepdogs.

The trajectories of sheep cluster in a small region, whereas each trajectory of sheepdogs is much longer and travels around in a wide area. They are not 'neat' artificial dense and sparse regions, as we have shown in Figure 1. Each trajectory is hap-hazard and there is no clear grouping, especially with sheepdogs. This is a real-world example of a dense region of sheep trajectories lying within a wider region of sparse (and scattered) trajectories of sheepdogs.

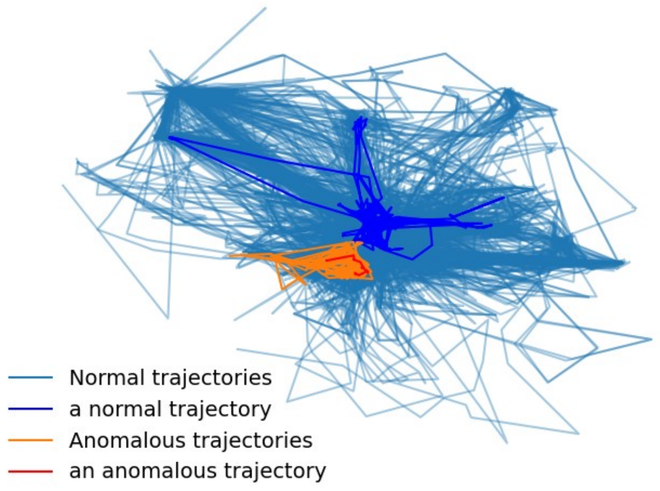


Figure 3: Trajectories in *Sheepdogs* dataset. Trajectories of sheepdogs are shown in blue, while trajectories of sheep are shown in orange.

LOF (with three distance measures and $\mathcal{K}_G$) and GDK$_G$ (with $\mathcal{K}_G$) perform poorly on this dataset. They all have detection accuracy ROC-AUCs ranging between 0.56 and 0.93. In contrast, IDK$_I$ and LOF$_I$ which employ $\mathcal{K}_I$ perform significantly better with ROC-AUC=0.98 and 0.99, respectively (the details are shown in Table 9).

The *Sheepdogs* dataset corresponds to the case of clustered anomalies found in point anomaly detection (Liu et al., 2010), where anomalous trajectories are clustered in a small region, while the majority of the (normal) trajectories are scattered in a wide area.

## 6. Trajectory Anomaly Detection with Distributional Kernel

To show the generality of the proposed distributional kernel for trajectory representation and similarity measurement, we apply the distributional kernel $\mathcal{K}$ to two applications: trajectory anomaly detection and anomalous sub-trajectory detection. We first review how distributional kernel $\mathcal{K}$ is used in point and group anomaly detection and then propose our algorithms of trajectory and sub-trajectory anomaly detection.

### 6.1 Point & Group Anomaly Detectors Based on $\mathcal{K}$

**Point Anomaly detection**: A point anomaly detector based on $\mathcal{K}_I$ called IDK$(x)$ for each point $x \in \mathsf{D}$ (Ting et al., 2020) is given as follows:

$$\text{IDK}(x) = \mathcal{K}_I(\delta(x), \mathcal{P}_{\mathsf{D}}|\mathsf{D}) \tag{4}$$

where $\delta(x)$ is a Dirac measure that converts a point into a distribution.

IDK$(x)$ returns a similarity score of point $x$ with respect to the (unknown) distribution $\mathcal{P}_{\mathsf{D}}$ which generates the dataset $\mathsf{D}$.

**Group Anomaly detection**: Given a dataset of groups of points $\{H_1, \ldots, H_m\}$ and $H_i \subset \mathbb{R}^{\mathbf{d}}$, a group anomaly detector aims to identify the few groups which are different from the majority of the groups in the dataset.

A group anomaly detector applies $\mathcal{K}_I$ in two levels (Ting et al., 2023). The first level maps each group to a point in a Hilbert space, i.e., $\mathbf{g} = \Phi(\mathcal{P}_H)$. Given the set of $m$ points $\Pi = \{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$ in Hilbert space, IDK$(x)$ can be applied to detect the point anomalies, where each point anomaly in Hilbert space corresponds to a group anomaly in input space.

Given that Gaussian kernel[1] can be used in place of the Isolation Kernel at each of the two levels, four variants of group anomaly detectors can be created. They are given in Table 5.

| Level 1 mapping ($\Phi(\mathcal{P}_H)$) | Level 2 detector |
|---|---|
| $\mathbf{g} = \Phi_I(\mathcal{P}_H)$ | $\text{IDK}_I(\mathbf{g}) = \mathcal{K}_I(\delta(\mathbf{g}), \mathcal{P}_{\Pi_\mathbf{g}}\|\Pi_\mathbf{g})$ |
| $\mathbf{g} = \Phi_I(\mathcal{P}_H)$ | $\text{GDK}_I(\mathbf{g}) = \mathcal{K}_G(\delta(\mathbf{g}), \mathcal{P}_{\Pi_\mathbf{g}}\|\Pi_\mathbf{g})$ |
| $\mathbf{h} = \Phi_G(\mathcal{P}_H)$ | $\text{IDK}_G(\mathbf{h}) = \mathcal{K}_I(\delta(\mathbf{h}), \mathcal{P}_{\Pi_\mathbf{h}}\|\Pi_\mathbf{h})$ |
| $\mathbf{h} = \Phi_G(\mathcal{P}_H)$ | $\text{GDK}_G(\mathbf{h}) = \mathcal{K}_G(\delta(\mathbf{h}), \mathcal{P}_{\Pi_\mathbf{h}}\|\Pi_\mathbf{h})$ |

Table 5: Four variants of group anomaly detectors, where the subscripts $_I$ and $_G$ denote the use of distributional kernels based on Isolation kernel and Gaussian kernel, respectively; $\Pi_\mathbf{g}$ and $\Pi_\mathbf{h}$ denote the sets of points $\mathbf{g}$ and $\mathbf{h}$, respectively.

1. Note that though Gaussian kernel has an infinite-dimensional feature map, one can use a kernel functional approximation method such as the Nyström method (Williams & Seeger, 2001) to produce an approximate finite-dimensional feature map. Then, a similar expression as Eq (3) can be produced.

In addition, as $\mathcal{K}_I$ and $\mathcal{K}_G$ are generic kernels, they can also be combined with existing anomaly detectors such as LOF (Breunig et al., 2000) and OCSVM (Schölkopf et al., 2001), by simply replacing the Euclidean distance (used in k-nearest neighbor employed in LOF) and Gaussian kernel (used in OCSVM) with either $\mathcal{K}_I$ or $\mathcal{K}_G$, to enable them to detect group anomalies (Ting et al., 2023).

In the next section, we show for the first time that $\mathcal{K}_I$ and $\mathcal{K}_G$ can be effectively used to represent trajectories (as groups of points) and measure similarity between two trajectories.

Note that k-nearest neighbors used in LOF and effectively 1-nearest neighbor used in both IDK and GDK are ideal in examining the effectiveness of $\mathcal{K}_I$ and $\mathcal{K}_G$ in measuring the similarity of trajectories in anomalous trajectory detection task. Because of the kernel use, they become k-most similar neighbors and 1-most similar neighbor respectively.

## 6.2 Anomalous Trajectory Detection

**Definition 6.1** ***Anomalous trajectory.*** *Given $D = \{X_i | i = 1, \dots, n\}$, an anomalous trajectory $Q \in D$ is rare wrt $D$ and is generated from a pdf different from those generating the normal trajectories $X_i \in D$, that is, for most $i \in [1, n], \mathcal{P}_{X_i} \neq \mathcal{P}_Q$.*

Following Definition 6.1, a distributional kernel $\mathcal{K}$ is used to represent each trajectory and compute the similarity between two trajectories.

In the light of Eq 3, the level-1 kernel mean map $\Phi(\mathcal{P}_X | D)$ from $\mathcal{K}$ that maps a trajectory $X$ to a point $\mathbf{g}$ in Hilbert space via the feature map $\phi$ built from $\mathsf{D} = \cup_{i=1}^{n} X_i$ is given as:

$$\mathbf{g} = \Phi(\mathcal{P}_X | D) = \frac{1}{|X|} \sum_{x \in X} \phi(x | \mathsf{D}) \tag{5}$$

Let $\Pi = \{\mathbf{g}_1, \dots, \mathbf{g}_n\}$ be the set of $\Phi$-mapped points from $D$.

With these representations, an existing point anomaly detector $\mathcal{F}(\cdot | \Pi)$ can be trained from $\Pi$, and then it provides the anomaly score for each $\mathbf{g} \in \Pi$, which corresponds to a trajectory in the given dataset $D$.

When IDK (Ting et al., 2020) is used as the detector $\mathcal{F}$, a score for any mapped point $\mathbf{g}$ with respect to $\Pi$ has the following expression (as used in Equations 2, 3 & 4):

$$\mathcal{F}(\mathbf{g} | \Pi) = \mathcal{K}(\delta(\mathbf{g}), \mathcal{P}_\Pi) = \langle \Phi_2(\delta(\mathbf{g})), \Phi_2(\mathcal{P}_\Pi) \rangle$$

where level-2 kernel mean map $\Phi_2(\mathcal{P}_\Pi) = \frac{1}{|\Pi|} \sum_{\mathbf{g} \in \Pi} \phi_2(\mathbf{g} | \Pi)$.

Note that $\Phi$ and $\Phi_2$ are derived from $D$ and $\Pi$, respectively. We drop '$|D$' and '$|\Pi$' for brevity hereafter.

The anomalous trajectories in $D$ correspond to those points $\mathbf{g} \in \Pi$ which have the highest anomaly scores. The procedure described above is summarized in Algorithm 1. Note that this algorithm is a generalization of the IDK$^2$ (Ting et al., 2023) algorithm which admits any point anomaly detector to be used. Table 6 shows three existing point anomaly detectors that employ mapping function $\Phi$ derived from $\mathcal{K}_I$. Mapping function $\Phi$ derived from $\mathcal{K}_G$ can be similarly applied.

**Algorithm 1** $\mathcal{K}$AT Anomalous Trajectory Detector

**Input**: Dataset of trajectories $D = \{X_i,\ i = 1, \dots, n\}$; kernel $\mathcal{K}(\cdot,\cdot) = \langle \Phi(\cdot), \Phi(\cdot) \rangle$; anomaly detector $\mathcal{F}$.
**Output**: List of $X_i,\ i = 1, \dots, n$ ordered by score $\alpha_i$.

```
1: * Map each trajectory $X_i \in D$ to a point in Hilbert space using the kernel mean map $\Phi(\mathcal{P}_{X_i})$
   For each $i = 1, \dots, n,\ \mathbf{g}_i = \Phi(P_{X_i})$
2: $\Pi = \{\mathbf{g}_i, i = 1, \dots, n\}$
3: * Build a point anomaly detector $\mathcal{F}$ from $\Pi$ and get score $\alpha_i$ for each $\mathbf{g}_i$
   For each $i = 1, \dots, n,\ \alpha_i = \mathcal{F}(\mathbf{g}_i|\Pi)$
4: Sort $X_i \in D$ in decreasing order by $\alpha_i$ if $\alpha_i$ is an anomaly score (in ascending order if $\alpha_i$ is a similarity score)
```

| Detector $\mathcal{F}$ that uses $\mathcal{K}_I$ | Algorithm 1: line 4 ($\alpha_i$) |
|---|---|
| IDK$_I$ | IDK($\mathbf{g}_i|\Pi$) |
| LOF$_I$ | LOF($\mathbf{g}_i|\Pi$) |
| OCSVM$_I$ | OCSVM($\mathbf{g}_i|\Pi$) |

Table 6: Algorithm 1 that incorporates existing point anomaly detectors IDK, LOF, and OCSVM, trained from Π.

### 6.3 Anomalous Sub-trajectory Detection

A trajectory $Q$ is anomalous with respect to a given set $D$ of normal trajectories if it contains anomalous sub-trajectories. We propose a simple yet effective algorithm based on $\mathcal{K}_I$ to detect the anomalous sub-trajectories that exist in an anomalous trajectory. The procedure is presented in Algorithm 2 called $\mathcal{K}$AST.

The idea is to identify the individual point-sub-trajectories (Definition 3.2) in the given anomalous trajectory $Q$ that makes $Q$ anomalous in $D$. Once the anomalous point-sub-trajectories are identified, the maximal sub-trajectories (Definition 3.3) are extracted from them to be the detected anomalous sub-trajectories $Q_s \prec Q$.

$\mathcal{K}_I$ is used to detect the anomalous point-sub-trajectories in $Q$ with respect to the average of kernel mean maps of all trajectories in $D$. In this process, the kernel mean map $\Phi$ of $\mathcal{K}_I$ is used to map each trajectory in $D$ in the input space into a point in Hilbert space, and map each point-sub-trajectory in $Q$ into a point in Hilbert space as well. The same feature map $\Phi$, constructed based on the trajectories in $D$, performs both mappings.

**Summary for Section 6**

The common ingredients in the above two algorithms are that (a) each trajectory $X$ or point-sub-trajectory $\ulcorner x \urcorner \prec Q$ in the input space is represented as a point in Hilbert space induced by distributional kernel $\mathcal{K}$ via its feature map $\Phi(\mathcal{P}_X)$ or $\Phi(\delta(x))$; (b) a group of trajectories is represented as an aggregate of their kernel mean maps; (c) the similarity between a trajectory/point-sub-trajectory and a group of trajectories is computed via a dot product of their mapped points in Hilbert space.

**Algorithm 2** $\mathcal{K}$AST Anomalous Sub-Trajectory Detector

**Input**: Dataset of normal trajectories $D = \{X_i,\ i = 1, \dots, n\}$; threshold $\tau$; kernel $\mathcal{K}(\cdot,\cdot) = \langle \Phi(\cdot), \Phi(\cdot) \rangle$; anomalous trajectory $Q = \ulcorner x_1, \dots, x_m \urcorner$.
**Output**: All maximal anomalous sub-trajectories $Q_s \prec Q$.

```
1: * Map each trajectory $X \in D$ to a point in Hilbert space using the kernel mean map $\Phi(\mathcal{P}_X)$ derived from $D$
   $\Pi = \{\mathbf{g}_i, i = 1, \dots, n\}$, where $\mathbf{g}_i = \Phi(\mathcal{P}_{X_i})$
2: * Score each point-sub-trajectory $\ulcorner x \urcorner \prec Q$ wrt the average of kernel mean maps of all $X \in D$.
   For each $\ulcorner x \urcorner \prec Q, \beta_x = \langle \Phi(\delta(x)), \bar{\mathbf{g}} \rangle$, where $\bar{\mathbf{g}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{g}_i$
3: $\mathsf{G} = \{\ulcorner x \urcorner \prec Q \mid \beta_x \le \tau\}$
4: Extract every maximal sub-trajectory $Q_s \prec Q$ in $\mathsf{G}$.
5: Return all maximal anomalous sub-trajectories $\forall Q_s \prec Q$
```

The key difference between $\mathcal{K}$AT (Algorithm 1) and $\mathcal{K}$AST (Algorithm 2) is that the former deals with trajectories only and the latter intends to find maximal sub-trajectories of an anomalous trajectory only. In $\mathcal{K}$AT, level-2 kernel mean map $\Phi_2$ is required to represent the group of all (normal and anomalous) trajectories in $D$ to detect the anomalous trajectories. In $\mathcal{K}$AST, only an average of the level-1 kernel mean maps of a group of trajectories is required, instead of the level-2 kernel mean map. This is because the input to $\mathcal{K}$AST are normal trajectories only, which can be identified by using $\mathcal{K}$AT.

## 7. Experimental Design and Settings

The experiments are designed to answer the following questions:

1. Does $\mathcal{K}_I$ perform better than $\mathcal{K}_G$ in $\mathcal{K}$AT?
2. Is there any advantage of distributional kernel $\mathcal{K}$ over existing similarity measures and representation methods?
3. Which is the best detector for anomalous trajectory detection and anomalous sub-trajectory detection?

To answer the first two questions, in addition to comparing the proposed kernels $\mathcal{K}_I$ with $\mathcal{K}_G$ in $\mathcal{K}$AT, they are also compared with

1. Three commonly used distance measures: fastDTW (Salvador & Chan, 2007), Hausdorff distance, and Fréchet distance.
2. Deep representation learning t2vec (Li et al., 2018).

These measures and representations are examined mainly in the context of anomaly detection. We examine the effectiveness of the above measures and representations by using them in four existing point anomaly detectors: IDK, GDK, LOF, and OCSVM, as already described in Table 5 and Section 6.1. Each of them assumes the role of point anomaly

| Methods | Parameter search ranges |
|---|---|
| $d_W$,$d_H$,$d_F$ | No parameter tuning is required |
| $\mathcal{K}_I$ | $\psi \in \{2^q \mid q = 1, 2, \ldots, 10\}$; $t = 100$ |
| $\mathcal{K}_G$ | Nyström setting: n_components = 100; |
| | $\sigma \in \{2^q \mid q = -10, -9, \ldots, 5\}$ |
| t2vec | cellsize $\in \{25, 50, 100\}$; minfreq $\in \{10, 50, 100\}$; |
| | hiddensize $\in \{2^q \mid q = 6, 7, 8, 9, 10\}$ |
| | hiddensize is the number of features used |
| LOF | $k \in \{1, \lfloor 0.1n \rfloor, \lfloor 0.2n \rfloor, \ldots, \lfloor 0.9n \rfloor\}$; |
| | $n$ is the number of trajectories |
| GDK,SVM | $\sigma \in \{2^q \mid q = -10, -9, \ldots, 5\}$ |
| | other default settings are used in SVM |
| IDK | $\psi \in \{2^q \mid q = 1, 2, \ldots, 10\}$; $t = 100$ |
| GM-VSAE | $C \in \{1, 5, 10, 20, 50, 80\}$; |
| | $C$ is the number of Gaussian components |
| EncDec-AD | $c \in \{4, 40, 64, 128\}$ |
| | LSTM layers $\in \{1, 2, 4\}$ |
| Anomaly Transformer | $d_{model} \in \{128, 256, 512\}$ |
| | $d_{model}$ is the channel number of hidden states |
| $\mathcal{K}$AST | $\psi = 4096$; $t = 100$; $\tau = 0$ on *Flyingfox* |
| | $\psi = 2048$; $t = 100$; $\tau = 0$ on *Curlews* |
| TRAOD | $\varepsilon = 1$; $\theta = 0.1$ on *Flyingfox* |
| | $\varepsilon = 1$; $\theta = 0.05$ on *Curlews* |
| | $\varepsilon$ is the threshold; $\theta$ is the penalty coefficient |
| RL4OASD | $\Delta = 0.4, D = 8$ |
| | $\Delta$ is threshold; $D$ is the delay factor |

Table 7: Parameter search ranges. The implementation of Isolation Kernel from Ting et al. (2020) is used to produce $\mathcal{K}_I$.

detector $\mathcal{F}$ in $\mathcal{K}$AT (Algorithm 1). In addition, three deep learning anomaly detectors: Deep anomaly detectors GM-VSAE (Liu et al., 2020), EncDec-AD (Malhotra et al., 2016), and Anomaly Transformer[2] (Xu et al., 2021) are also included in the comparison.

In the task of anomalous sub-trajectory detection, we compare $\mathcal{K}$AST (Algorithm 2) with a recent deep learning method called RL4OASD (Zhang et al., 2023) and a well-cited work TRAOD (Lee et al., 2008).

**Evaluation metrics**. ROC-AUC score is used to evaluate the detection accuracy of the anomaly detectors.

**Parameter settings**. The search ranges for all parameters in the experiments are given in Table 7. The machine used in the experiments has two AMD7742 64-core CPUs &

2. Since trajectory is similar to time series, we apply this time series anomaly detector on trajectories to see whether it can work well.

| Dataset | #Points | min – max $\|X\|$ | #Traj | #AT | %AT |
|---|---|---|---|---|---|
| Geolife | 5,399,510 | 22 – 100 | 80,450 | 3,202 | 4% |
| Baboons | 1,020,107 | 30 – 603 | 2,310 | 110 | 5% |
| Curlews | 801,489 | 488 – 71,821 | 42 | 9 | 21% |
| Character | 446,643 | 109 – 205 | 2,643 | 28 | 1% |
| Detrac | 445,052 | 11 – 2,120 | 5,356 | 71 | 1% |
| Vrut | 407,402 | 55 – 1,257 | 1,168 | 100 | 9% |
| Wildebeest | 279,082 | 138 – 5,632 | 92 | 14 | 15% |
| Vultures | 212,485 | 172 – 7,721 | 67 | 15 | 22% |
| Cross | 153,010 | 4 – 30 | 11600 | 200 | 2% |
| Casia | 143,383 | 16 – 612 | 1,500 | 24 | 2% |
| Flyingfox | 132,252 | 517 – 4,768 | 62 | 11 | 18% |
| Sheepdogs | 65,574 | 11 – 3,501 | 538 | 23 | 4% |
| Traffic | 11,500 | 50 – 50 | 230 | 30 | 13% |

Table 8: Real-world datasets. AT: anomalous trajectories. min – max $|X|$: the minimum and maximum numbers of points of individual trajectories in a dataset.

1024GB memory, and two GPUs RTX3090 24GB. The GPUs are used by t2vec (Li et al., 2018), EncDec-AD (Malhotra et al., 2016), and GM-VSAE (Liu et al., 2020) only.

**Datasets**. We perform the evaluations on twelve datasets, among which four datasets (*Cross* (Morris & Trivedi, 2009), *Traffic*[3] (Lin et al., 2017), *Casia*[4] (Hu et al., 2013), *Detrac*[5] (Wen et al., 2020)) have been used in previous anomaly detection works and two are classification datasets (*Character*[6], *Vrut*[7]). Another dataset *Geolife* is unlabeled, we follow the work of E$^2$DTC (Fang et al., 2021) to first cluster all the trajectories in the dataset, and then choose trajectories in one of the clusters as anomalies. The other six datasets (*Baboons*, *Curlews*, *Wildebeest*, *Vultures*, *Flyingfox*, *Sheepdogs*) are collected from MoveBank[8], where each dataset records trajectories of a kind of animal over a period. In MoveBank datasets, trajectories that deviate from the majority are manually labeled as anomalies (see the Appendix for details). The data characteristics of these datasets are given in Table 8.

## 8. Experimental Results

We present the results of two applications: anomalous trajectory detection and anomalous sub-trajectory detection in the following subsections.

3. https://min.sjtu.edu.cn/lwydemo/Trajectory\%20analysis.htm
4. https://github.com/mcximing/ACCV18_Anomaly
5. https://detrac-db.rit.albany.edu/
6. https://archive.ics.uci.edu/ml/datasets/Character+Trajectories
7. https://www.th-ab.de/ueber-uns/organisation/labor/kooperative-automatisierte-verkehrssysteme/trajectory-dataset
8. https://www.movebank.org/cms/movebank-main

| | ED-based | | | $\mathcal{K}$AT | | | | | | Rep. learning (t2vec) | | | | GM | EncDec | AT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | LOF$_W$ | LOF$_H$ | LOF$_F$ | LOF$_I$ | LOF$_G$ | SVM$_I$ | SVM$_G$ | GDK$_G$ | IDK$_I$ | LOF | SVM | GDK | IDK | -VSAE | -AD | |
| Geolife | OOT | .96 | OOT | .96 | .96 | .61 | .83 | .88 | .96 | .60 | .56 | 56. | .61 | .52 | .52 | .54 |
| Baboons | .91 | .90 | **.99** | .96 | **.99** | .86 | .71 | .95 | **.99** | .72 | .78 | .81 | .80 | .60 | .84 | .84 |
| Curlews | .67 | .79 | **.90** | .82 | .72 | .73 | .78 | .67 | .83 | .62 | .49 | .38 | .36 | .51 | .53 | OOM |
| Character | .85 | .70 | .78 | .76 | .85 | .49 | .47 | .47 | **.88** | .76 | .82 | .83 | .82 | .55 | .81 | .60 |
| Detrac | .82 | .78 | .70 | **.84** | .56 | .83 | .54 | .63 | **.84** | .75 | .68 | .64 | .72 | .50 | .70 | OOM |
| Vrut | .91 | **.93** | .87 | .91 | .91 | .80 | .85 | .84 | .89 | .77 | .72 | .73 | .77 | .50 | .87 | .59 |
| Wildebeest | .78 | .78 | **.89** | .84 | .72 | .61 | .61 | .77 | .82 | .75 | .75 | .73 | .77 | .53 | .59 | OOM |
| Vultures | .79 | .74 | .83 | .84 | .85 | .74 | .81 | .81 | **.87** | .76 | .71 | .75 | .78 | .51 | .69 | OOM |
| Cross | .82 | .89 | .84 | .83 | .82 | .64 | .77 | .92 | **.94** | .62 | .83 | .87 | 83 | .50 | .50 | .55 |
| Casia | .70 | .69 | .73 | .78 | .61 | .84 | .62 | .71 | **.89** | .64 | .65 | .66 | .69 | .50 | .82 | .51 |
| Flyingfox | .85 | .88 | **.96** | .84 | .72 | .80 | .67 | .74 | .89 | .83 | .70 | .71 | .67 | .52 | .76 | OOM |
| Sheepdogs | .81 | .72 | .56 | **.99** | .93 | .95 | .71 | .70 | .98 | .98 | .98 | **.99** | **.99** | .50 | .83 | OOM |
| Traffic | .75 | .62 | .81 | **.98** | .91 | .52 | .69 | .86 | .96 | .77 | .63 | .71 | .76 | .52 | .96 | .91 |
| Rank: | 6.3 | 6.9 | 5.2 | 3.8 | 7.2 | 8.9 | 11.2 | 8.0 | 2.3 | 9.2 | 10.3 | 9.1 | 8.6 | 14.4 | 8.9 | - |

Table 9: ROC-AUC results of different methods. ED is short for Euclidean distance. In $\mathcal{K}$AT, the anomaly detectors that employ kernel mean maps $\Phi$ derived from $\mathcal{K}_I$ & $\mathcal{K}_G$ are denoted with subscripts $_I$ & $_G$, respectively. SVM denotes OCSVM (Schölkopf et al., 2001). AT denotes Anomaly Transformer. Boldface indicates the best ROC-AUC score in each dataset. OOT means that the result could not be obtained in 2 days. OOM denotes that an algorithm has an 'out of memory' error during execution. The last row shows the rankings of all detectors in each dataset, averaged over all datasets.

## 8.1 Anomalous Trajectory Detection

Table 9 presents the ROC-AUC results of anomalous trajectory detection. The following are the main findings of the experiments:

1. In terms of similarity measures: In $\mathcal{K}$AT, all three detectors GDK, LOF & SVM employing $\mathcal{K}_I$ are ranked higher than those using $\mathcal{K}_G$. LOF with $\mathcal{K}_I$ is also ranked better than that employing $d_W$, $d_H$ and $d_F$. Despite having extra learning, t2vec is not competitive compared with all other no-learning measures on most datasets.

2. In terms of anomaly detectors: IDK$_I$ (which employs $\mathcal{K}_I$) performs better than all other detectors. The closest contender is LOF$_I$ which also employs $\mathcal{K}_I$. Deep learning anomaly detectors, GM-VSAE, EncDec-AD and Anomaly Transformer perform worse than LOF$_I$ and IDK$_I$. These deep learning results on anomalous trajectory detection are consistent with those on time series anomaly detection (Paparrizos et al., 2022; Schmidl et al., 2022). Additional analyses are provided in the Appendix.

Figure 4 shows the result of the Friedman-Nemenyi test (Demšar, 2006), comparing IDK$_I$ with GDK$_G$, LOF$_F$ (the top-ranked LOF that uses a distance measure), LOF$_I$ (the top-ranked LOF that uses $\mathcal{K}$), t2vec+IDK (the top-ranked t2vec) and EncDec-AD (the top-ranked end-to-end deep anomaly detector). Although IDK$_I$ is not significantly better than LOF$_I$ and LOF$_F$, it is the only detector that is significantly better than EncDec-AD, GDK$_G$ and t2vec+IDK.

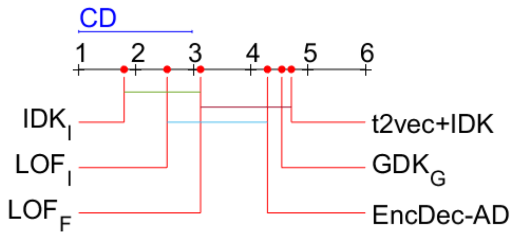


Figure 4: Friedman-Nemenyi test at 0.10 significance level. No significant difference if two detectors are connected by a CD line.

We also examine the time complexity of $\mathcal{K}$AT. For a dataset $D$ with $n$ trajectories and a total of $N$ points, the time complexity for IDK mapping is $\mathcal{O}(N\psi t)$ and for anomaly detection is $\mathcal{O}(n\psi t)$, where both $\psi$ and $t$ are parameters of IDK (Ting et al., 2020, 2023).

| | $10^2$ traj | | $10^4$ traj | | runtime ratio | |
|---|---|---|---|---|---|---|
| | prep | AD | prep | AD | prep | AD |
| LOF$_W$ | 2 | .006 | 1081782 | 19 | 540891 | 3167 |
| LOF$_H$ | 2 | .004 | 549055 | 14 | 274528 | 3500 |
| LOF$_F$ | 3 | .003 | 424061 | 12 | 141354 | 4000 |
| LOF$_I$ | .9 | .004 | 798 | 11 | 887 | 2750 |
| LOF$_G$ | 1.8 | .002 | 1663 | 14 | 924 | 7000 |
| SVM$_I$ | 1.7 | .001 | 1663 | 10 | 978 | 10000 |
| SVM$_G$ | 2.1 | .003 | 1663 | 10 | 792 | 3333 |
| IDK$_I$ | 2.7 | | 203 | | 75 | |
| GDK$_G$ | 1.2 | | 388 | | 323 | |
| t2vec+LOF | 281 | .006 | 452 | 36 | 2 | 6000 |
| t2vec+SVM | 281 | .024 | 452 | 93 | 2 | 3875 |
| t2vec+IDK | 281 | .070 | 452 | 6.4 | 2 | 91 |
| t2vec+GDK | 281 | .064 | 452 | 14.8 | 2 | 231 |
| GM-VSAE | 12 | | 31 | | 3 | |
| EncDec-AD | 702 | | 3908 | | 6 | |
| Anomaly Transformer | 105 | | 27542 | | 262 | |

Table 10: Runtime (all in CPU secs except t2vec, GM-VSAE and EncDec-AD employing GPU). The preprocessing (prep) includes all calculations to produce a similarity matrix. AD is the runtime of the anomaly detector only. The ratio is the runtime for $10^4$ trajectories over that for $10^2$ trajectories.

We perform a scaleup test using $10^2$ trajectories and $10^4$ trajectories randomly selected from the *Cross* dataset, and the result is presented in Table 10. Key observations are:

1. A striking difference between the three distance measures (the first three rows in the prep runtime ratio column) and the distributional kernels (the next two rows) that employ the same LOF: each of the three distance measures runs three orders of magnitude slower than either $\mathcal{K}_I$ or $\mathcal{K}_G$.

2. Out of the six versions of $\mathcal{K}$AT, IDK$_I$ & GDK$_G$ have linear runtime; LOF and SVM (using either $\mathcal{K}_I$ or $\mathcal{K}_G$) have superlinear or quadratic runtime, as shown in the AD runtime ratio column.

3. Because of using GPUs, t2vec, GM-VSAE, and EncDec-AD have the lowest scaleup ratios, while all other methods run on CPUs only.

A neural metric learning method has been proposed to speed up distance measures such as $d_W$ and $d_H$ by using an RNN to approximate a distance measure: achieving 50x to 1000x speedup at the cost of (degraded) 80% accuracy of a distance measure (Yao et al., 2019). This method does not change our conclusion here on anomalous trajectory detection because using it weakens the accuracy of LOF for $d_W$, $d_H$ and $d_F$ we have obtained in Table 9.

## 8.2 Anomalous Sub-trajectory Detection

Anomalous sub-trajectory detection can be conducted in two ways, depending on whether the task is on a network or not. The first subsection describes the detection on a road network, and the second subsection presents the detection without a network.

### 8.2.1 Anomalous Sub-trajectory Detection on a Road Network

A road network is a graph $(V, E)$, where each node $v \in V$ represents a traffic hub, and $e \in E$ is an edge connecting two nodes in the network. A trajectory $Y$ on a road network consists of a series of connecting nodes on the network, i.e., $Y = \ulcorner v_1, \dots, v_\mu \urcorner$, and each edge represents the shortest segment of a trajectory connecting two adjacent nodes.

Because edges of trajectories are readily available in a road network, we propose a new way to detect anomalous sub-trajectories as follows. Firstly, transform all the given trajectories $Y_j, j = 1, \dots, w$ into a set of node-pairs $D = \{X_i = \ulcorner v, \vartheta \urcorner, i = 1, \dots, n\}$ & $v, \vartheta \in V$, where each node pair is a sub-trajectory. Secondly, $D$ is used as input to $\mathcal{K}$AT (Algorithm 1) to detect the anomalous node-pairs (or sub-trajectories) in $D$.

The above treatment of edge as sub-trajectory is the same as that used by a recent method called RL4OASD (Zhang et al., 2023) which is specially designed for road networks. The procedure can be divided into two steps. The first step treats the road network as a graph and performs a node embedding using Toast Embedding (Chen et al., 2021) to obtain a vector for each node in the graph. The second step treats each trajectory in the dataset as consisting of nodes connected in a sequence in the graph, where an edge represents a sub-trajectory. Using the sequences as input, RL4OASD uses LSTM and deep reinforcement learning to train a detector to find the anomalous sub-trajectories (the abnormal edges in the graph).

As the key difference is the second step, we conduct a head-to-head comparison by replacing the LSTM-deep-reinforcement-learning-based detector with $\mathcal{K}$AT, represented by the most effective IDK-based detector IDK$_I$ in Section 8.1. They both use the same features derived from the pre-trained node embedding in the first step.

In the experiment, we use the Chengdu[9] dataset which has a total of $558,098$ trajectories and $3,930$ anomalous trajectories (Zhang et al., 2023). The road network of Chengdu has $4,891$ nodes and $12,469$ edges, where each node has 128 attributes.

In the experiment, $\mathcal{K}$AT produces ROC-AUC $= 0.89$ in 0.17 hours using CPU; while RL4OASD yields ROC-AUC $= 0.83$ in 0.25 hours using GPU.

In summary, using the same node embedding and treating each edge of the Chengdu network of a trajectory as a sub-trajectory, the distributional kernel based detector $\mathcal{K}$AT achieves higher detection accuracy than deep learning based detector RL4OASD, and completes the task faster with CPU than RL4OASD with GPU.

#### 8.2.2 Anomalous Sub-trajectory Detection on an Anomalous Trajectory

Without a network, $\mathcal{K}$AST (Algorithm 2) with $\mathcal{K}_I$ is used to detect anomalous sub-trajectories of an anomalous trajectory. To evaluate the detection accuracy of $\mathcal{K}$AST, we compare it with a highly cited sub-trajectory detection method TRAOD (Lee et al., 2008). Note that RL4OASD (Zhang et al., 2023) could not be applied here because no information about a network is available.

The ground-truth anomalous sub-trajectories in a dataset are identified using the following method objectively. A point in an anomalous trajectory is labeled anomalous if there is no normal trajectory in the local neighborhood centered at it with a radius of $r$ ($r = 0.1$ in *Curlews* and $r = 0.01$ in *Flyingfox*), and then contiguous anomalous points are linked into anomalous sub-trajectories (sub-trajectories that are too short are discarded). Eleven trajectories on the *Flyingfox* dataset are identified as anomalous in this process.

We employ a commonly used Jaccard index (Jaccard, 1912) to measure the matching between a detected anomalous sub-trajectory and a ground truth. It measures how well the two matched, the larger the index the better.

All results on *Flyingfox* are shown in Table 11. $\mathcal{K}$AST performs better than TRAOD in terms of the Jaccard index, and it runs two orders of magnitude faster. Table 12 shows the example results of anomalous sub-trajectory detection on the *Flyingfox* and *Curlews* datasets.

The example on the *Flyingfox* dataset has two anomalous trajectories $Q_1$ & $Q_2$. $\mathcal{K}$AST identifies that $Q_1$ has two anomalous sub-trajectories (drawn in red, with normal sub-trajectories drawn in green), and $Q_2$ has one anomalous sub-trajectory.

TRAOD included parts of the normal sub-trajectories as anomalous sub-trajectories in both $Q_1$ & $Q_2$, and only one anomalous sub-trajectory was detected in $Q_1$.

On the larger *Curlews* dataset consisting of more than 800,000 points, TRAOD took 2 days to identify the anomalous sub-trajectories of a given anomalous trajectory, whereas $\mathcal{K}$AST using $\mathcal{K}_I$ completed it in less than 15 minutes. Besides, Algorithm 2 produces a more accurate result consistent with the ground truth as shown in the last row of Table 12.

In the absence of a powerful kernel like $\mathcal{K}_I$, TRAOD using a partition-and-detect framework is a sensible method. As we have shown in Table 12, the sub-trajectories, as a result of the partitioning before the detection of anomalous sub-trajectories, are a coarse approximation. It is unable to produce the fine-grained sub-trajectories discovered by the proposed $\mathcal{K}$AST. On the other hand, TRAOD has high time complexity because it employs a com-

9. https://outreach.didichuxing.com/

| Trajectory Index | Jaccard index | | Time (seconds) | |
|---|---|---|---|---|
| | $\mathcal{K}$AST | TRAOD | $\mathcal{K}$AST | TRAOD |
| 1 | 0.94 | 0.57 | 3.4 | 800 |
| 21 | 0.91 | 0.82 | 2.1 | 471 |
| 24 | 0.62 | 0.52 | 4.5 | 1130 |
| 26 | 0.85 | 0.80 | 2.4 | 764 |
| 34 | 0.90 | 0.81 | 1.8 | 581 |
| 38 | 0.71 | 0.60 | 4.4 | 1125 |
| 41 | 0.75 | 0.67 | 2.0 | 647 |
| 45 | 0.91 | 0.81 | 1.5 | 396 |
| 46 | 0.95 | 0.86 | 1.8 | 671 |
| 47 | 0.87 | 0.68 | 0.7 | 208 |
| 48 | 0.98 | 0.91 | 0.6 | 167 |
| avg | 0.85 | 0.73 | 2.3 | 632 |

Table 11: $\mathcal{K}$AST VS TRAOD on *Flyingfox*. Each row shows the detection results and runtimes for each of the eleven anomalous trajectories identified on the *Flyingfox* dataset.

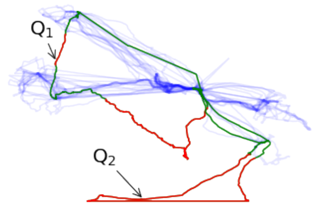
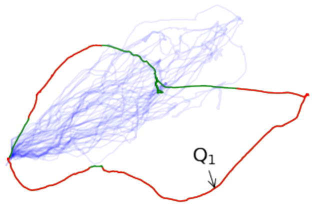
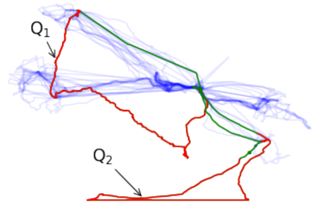
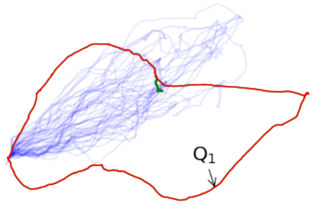
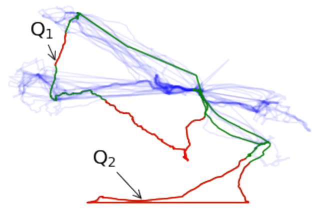
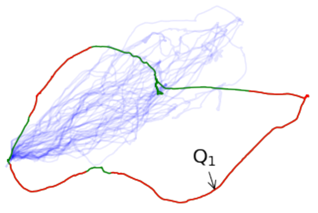
| | Flyingfox | Curlews |
|---|---|---|
| Ground truth |  |  |
| TRAOD |  |  |
| $\mathcal{K}$AST |  |  |

Table 12: Anomalous sub-trajectory detection results of $\mathcal{K}$AST and TRAOD on *Flyingfox* and *Curlews*. $Q_1$ & $Q_2$ are two separate anomalous trajectories. In each anomalous trajectory, the detected anomalous sub-trajectories are colored in red; normal sub-trajectories are colored in green. Every normal trajectory is colored in blue.

bination of three distances (to represent the horizontal, vertical and angular distances) in the partition component to subdivide each trajectory into sub-trajectories. After the partitioning process, it employs LOF (Breunig et al., 2000) with Hausdorff distance to identify anomalous sub-trajectories among all sub-trajectories. Trajectories with identified anomalous sub-trajectories are reported to be anomalous. TRAOD is a computationally expensive process because all computations are point-based, not distribution-based.

## 9. Discussion

Here we provide some further discussion with respect to distribution information and distributional kernel.

### 9.1 Distribution Information VS Shape Information

In practice, we may be interested in different aspects of trajectories. Some applications focus on the shape of the trajectory rather than the distribution of specific sampling points. Other applications are interested in the detailed behavior of the object, such as speed, stopping time in a certain area, etc., and paying more attention to the distribution of data points.

A typical example of a shape-based method (Lee et al., 2007) partitions a trajectory into a set of line segments and measures the similarity between them. Shape-based methods capture the global structure of trajectories but cannot consider more detailed trajectory information. Trajectories with the same shape may be generated by completely different behaviors, and it is not possible to differentiate them by considering shape information only.

In contrast to shape-based methods, our proposed distributional kernel represents and measures the similarity of trajectories based on the distribution information. Distribution-based methods identify local changes within a trajectory, which could indicate the behavior of the moving object.

The flip side of this sensitivity is that distribution-based methods can be more easily influenced by the distribution of sampled points. If the density of sampled points is very high in a certain region (e.g., an object stays at some location for a long period of time), this segment will have a strong influence when measuring the similarity, while information in other regions may carry less weight.

In general, the choice between shape-based and distribution-based methods depends on the data properties and the application requirements. Shape-based methods are more suitable for tasks where the overall shape is more critical than fine-grained details. For tasks involving density, concentration, or local changes, distribution-based methods could be more relevant.

### 9.2 Distributional Kernel VS Deep Learning Methods & Set-based Distances

It is interesting to note that deep learning methods t2vec (Li et al., 2018), EncDec-AD (Malhotra et al., 2016), and Anomaly Transformer (Xu et al., 2021) are not competitive with the proposed distributional kernel $\mathcal{K}_I$ without learning, as shown in Table 9 in Section 8.1. This suggests that a powerful kernel such as $\mathcal{K}_I$ is more effective and efficient than deep learning methods for anomalous trajectory detection. This is mainly due to the use of distributional information and the data-dependent property in $\mathcal{K}_I$.

The above result also raises three questions: (i) Can deep learning produce a representation or measure that is as powerful as $\mathcal{K}_I$? (ii) Can deep learning representations or measures have a data-dependent property like the one provided by $\mathcal{K}_I$? (iii) Can deep learning produce a measure $dist(\cdot,\cdot)$ which guarantees the uniqueness property: $dist(X,Y)=0$ if and only if $X=Y$? These are interesting topics for future research in deep learning.

Many existing measures (e.g., the set-based Hausdorff and Fréchet distances) are based on i.i.d. implicitly. In contrast, our approach brings the i.i.d. assumption to the forefront and uses a distributional measure. The fact that it works well in practice indicates that i.i.d. is a veritable assumption for valid trajectories in the real world.

### 9.3 When Distributional Kernel Fails and Its Fixes

In Section 8, we demonstrated that the use of a distribution kernel for trajectories is effective in real-world datasets. However, there are certain circumstances where the distributional kernel may fail. In this regard, we would like to examine two such situations where minor alterations can fix the problems.

Firstly, when multiple cycles occur on the same path, the distributional kernel fails to differentiate between the trajectory of one cycle and the trajectory of many cycles. If it is necessary to distinguish between these two scenarios, the length of each trajectory can be added as an additional attribute, so that trajectories with the same shape but different lengths can be differentiated.

Secondly, when it is important to differentiate between different traveling agents, the distributional kernel may not suffice. In such cases, an additional agent attribute can be included to distinguish between different agents.

Generally, by adjusting the attributes of the trajectory data, the distributional kernel could be able to process different trajectory information. These tweaks are minor and can accommodate special requirements, and they are not fundamental limitations of the proposed distributional kernel.

## 10. Conclusion

In this paper, we show that distributional kernel is a powerful tool for both trajectory similarity measurement and trajectory anomaly detection. While precious works are mostly point-based, distributional kernel captures the distribution information of trajectory data.

With the important uniqueness property and linear runtime, distributional kernel is able to address the effective and efficient issue of existing measures for trajectories. Moreover, the data-dependent $\mathcal{K}_I$ is always better than the data-independent $\mathcal{K}_G$, even though both have the same uniqueness property, which indicates the significance of the data-dependent property to lift the detection capability in datasets with clusters of varied densities.

We show the power of the distributional kernel and its impacts in two anomaly detection tasks. $\mathcal{K}_I$ produces better detection accuracy than all other measures and representations mentioned in this paper. Coupled with the existing detector IDK, IDK$_I$ performs significantly better than four deep learning anomaly detectors. This is because only $\mathcal{K}_I$ has the uniqueness and the data-dependent properties. None of the deep learning anomaly detectors have been shown to have these properties. In addition, $\mathcal{K}_I$ runs orders of magnitude faster than the three existing distance measures.

We also show that the proposed $\mathcal{K}$AST algorithm for anomalous sub-trajectory detection is simpler, faster, and more effective than the deep learning-based RL4OASD and the widely cited partition-and-detect method TRAOD.

## 11. Acknowledgements

This project is supported by National Natural Science Foundation of China (Grant No. 62076120).

## Appendix A. Datasets

*Baboons*, *Curlews*, *Wildebeest*, *Vultures*, *Flyingfox* and *Sheepdogs* are collected from Movebank[10], which record trajectories of different animals over a time period.

Trajectories in each dataset are extracted as follows:

**Baboons**: Each original trajectory is a GPS-recorded activity of a baboon over half a month in August 2012. Because the original has a very high sampling rate, we reduced the sampling rate by a factor of 1000. A small number of trajectories that deviate from the majority are considered anomalous.

**Curlews**: Each trajectory is an annual migration path of a curlew. A few trajectories which have different starting points or are not back to the same starting point are considered to be anomalous.

**Wildebeest**: Each trajectory is an annual migration path of a wildebeest, and a few trajectories that have different routes are considered to be anomalous.

**Vultures**: Each trajectory is an annual migration path of a vulture. Those having no return trips or are too short are considered anomalous.

**Flyingfox**: Each trajectory is a daily activity path of a flying fox. Trajectories that are significantly different from the majority or do not return to the starting point, are considered to be anomalous.

**Sheepdogs**: Trajectories are extracted between a long pause of a recording device. 515 trajectories belonging to sheepdogs are normal, while 23 trajectories belonging to sheep are anomalies. An example visualization is shown in Figure 3.

## Appendix B. Additional analyses of deep learning

This section provides the details of additional analyses on two deep learning anomaly detectors **GM-VSAE** and **EncDec-AD**, and representation deep learning **t2vec**, focusing on the issue of the kind of datasets used for training.

**GM-VSAE** achieves ROC-AUC score of 0.69 on *Baboons*, which is the worst among all methods listed in Table 9. Its ROC-AUC scores on all other datasets are worse than that on *Baboons*. GM-VSAE has been given the advantage of using a training set of normal trajectories only because it aims to model normality (Liu et al., 2020). All other methods in Table 9 are trained using the given dataset that contains anomalous trajectories.

10. www.movebank.org/cms/movebank-main

GM-VSAE was reported to have high PR-AUC on two datasets only (Liu et al., 2020). However, the result is an outcome of wrongly assigning normal trajectories as positive examples in computing the precision-recall curve (see their code at https://git.io/JelML, retrieved on 26 October 2021). This means that GM-VSAE is good at ranking many normal trajectories at the top. But this says nothing about its ability to detect anomalous trajectories.

Recall that GM-VSAE has one major drawback, i.e., its grid-based representation could not guarantee the uniqueness property such that two different trajectories can potentially be mapped into the same series of tokens. We think that this is the main cause of its poor detection performance.

**EncDec-AD**: We conducted an additional supervised version experiment for EncDec-AD by training the model with normal trajectories only. The experiment result shows that although the ROC-AUC score increases on some datasets, EncDecAD is still not competitive with IDK$_I$.

**t2vec**: We also conducted an experiment to train t2vec on the given dataset versus the set of normal trajectories only. The difference is small, and there is no suggestion that the latter will produce a better result. Their best results are still significantly worse than all results of Euclidean distance-based measures and distributional kernels (except one) shown in Table 9.

An interesting phenomenon is that IDK performs better when trained with the given dataset than with normal trajectories only, unlike other detectors. The robustness of IDK to noise in the training set has been previously revealed (Ting et al., 2020).

Our results on deep learning are consistent with those on time series anomaly detection (Paparrizos et al., 2022; Schmidl et al., 2022), summarized below:

".. deep learning approaches are not (yet) competitive despite their higher processing effort on training data." (Schmidl et al., 2022)

".. CNN and LSTM ... are the third and the second-worst for sequence-based anomalies." (Paparrizos et al., 2022)

## References

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 93–104.

Chen, C., Zhang, D., Castro, P. S., Li, N., Sun, L., Li, S., & Wang, Z. (2013). iBOAT: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 806–818.

Chen, L., & Ng, R. (2004). On the marriage of Lp-Norms and edit distance. *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, 792–803.

Chen, L., Özsu, M. T., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 491–502.

Chen, Y., Li, X., Cong, G., Bao, Z., Long, C., Liu, Y., Chandran, A. K., & Ellison, R. (2021). Robust road network representation learning: When traffic patterns meet traveling semantics. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 211–220.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, *7*, 1–30.

Eiter, T., & Mannila, H. (1994). Computing discrete Fréchet distance.

Fang, Z., Du, Y., Chen, L., Hu, Y., Gao, Y., & Chen, G. (2021). E$^2$DTC: An end to end deep trajectory clustering framework via self-training. *2021 IEEE 37th International Conference on Data Engineering*, 696–707.

Fang, Z., Du, Y., Zhu, X., Hu, D., Chen, L., Gao, Y., & Jensen, C. S. (2022). Spatio-temporal trajectory similarity learning in road networks. *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 347–356.

Fukumizu, K., Bach, F. R., & Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, *5*(Jan), 73–99.

Gray, K., Smolyak, D., Badirli, S., & Mohler, G. (2018). Coupled IGMM-GANs for deep multimodal anomaly detection in human mobility data. *arXiv preprint arXiv:1809.02728*.

Hu, W., Li, X., Tian, G., Maybank, S., & Zhang, Z. (2013). An incremental DPMM-Based method for trajectory clustering, modeling, and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(5), 1051–1065.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, *11*(2), 37–50.

Lee, J.-G., Han, J., & Li, X. (2008). Trajectory outlier detection: A partition-and-detect framework. *Proceedings of the IEEE 24th International Conference on Data Engineering*, 140–149.

Lee, J.-G., Han, J., & Whang, K.-Y. (2007). Trajectory clustering: A partition-and-group framework. *Proceedings of ACM SIGMOD international conference on Management of data*, 593–604.

Li, X., Zhao, K., Cong, G., Jensen, C. S., & Wei, W. (2018). Deep representation learning for trajectory similarity computation. *Proceedings of the IEEE 34th International Conference on Data Engineering*, 617–628.

Lin, W., Zhou, Y., Xu, H., Yan, J., Xu, M., Wu, J., & Liu, Z. (2017). A tube-and-droplet-based approach for representing and analyzing motion trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(8), 1489–1503.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2010). On detecting clustered anomalies using sciforest. *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 274–290.

Liu, Y., Zhao, K., Cong, G., & Bao, Z. (2020). Online anomalous trajectory detection with deep generative sequence modeling. *IEEE 36th International Conference on Data Engineering*, 949–960.

Ma, C., Miao, Z., Li, M., Song, S., & Yang, M. H. (2018). Detecting anomalous trajectories via recurrent neural networks. *Proceedings of the 14th Asian Conference on Computer Vision*, 370–382.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Morris, B., & Trivedi, M. (2009). Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. *IEEE Conference on Computer Vision and Pattern Recognition*, 312–319.

Morris, B. T., & Trivedi, M. M. (2011). Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(11), 2287–2301.

Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al. (2017). Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, *10*(1-2), 1–141.

Paparrizos, J., Kang, Y., Boniol, P., Tsay, R. S., Palpanas, T., & Franklin, M. J. (2022). TSB-UAD: An end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment*, *15*(8), 1697–1711.

Qin, X., Ting, K. M., Zhu, Y., & Lee, V. C. S. (2019). Nearest-neighbour-induced isolation similarity and its impact on density-based clustering. *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence*, 4755–4762.

Rockafellar, R. T., & Wets, R. J.-B. (1998). *Variational analysis*. Springer.

Sakoe, H., & Chiba, S. (1971). A dynamic programming approach to continuous speech recognition. *Proceedings of the 7th International Congress on Acoustics*, 65–69.

Salvador, S., & Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, *11*(5), 561–580.

Schmidl, S., Wenig, P., & Papenbrock, T. (2022). Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, *15*(9), 1779–1797.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computing*, *13*(7), 1443–1471.

Smola, A., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert Space embedding for distributions. *Proceedings of Algorithmic Learning Theory*, 13–31.

Song, L., Wang, R., Xiao, D., Han, X., Cai, Y., & Shi, C. (2018). Anomalous trajectory detection using recurrent neural network. *Proceedings of the 14th International Conference on Advanced Data Mining and Applications*, 263–277.

Sousa, R. S. D., Boukerche, A., & Loureiro, A. A. F. (2020). Vehicle trajectory similarity: Models, methods, and applications. *ACM Computing Survey*, *53*(5).

Ting, K. M., Xu, B.-C., Washio, T., & Zhou, Z.-H. (2020). Isolation distributional kernel: A new tool for kernel based anomaly detection. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 198–206.

Ting, K. M., Xu, B.-C., Washio, T., & Zhou, Z.-H. (2023). Isolation distributional kernel: A new tool for point and group anomaly detections. *IEEE Transactions on Knowledge and Data Engineering*, *35*(03), 2697–2710.

Ting, K. M., Zhu, Y., & Zhou, Z.-H. (2018). Isolation kernel and its effect on SVM. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2329–2337.

Vlachos, M., Kollios, G., & Gunopulos, D. (2002). Discovering similar multidimensional trajectories. *Proceedings 18th International Conference on Data Engineering*, 673–684.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.-C., Qi, H., Lim, J., Yang, M.-H., & Lyu, S. (2020). UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, *193*, 102907.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems 13* (pp. 682–688).

Wu, H., Sun, W., & Zheng, B. (2017). A fast trajectory outlier detection approach via driving behavior modeling. *Proceedings of the ACM on Conference on Information and Knowledge Management*, 837–846.

Xu, J., Wu, H., Wang, J., & Long, M. (2021). Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*.

Yao, D., Cong, G., Zhang, C., & Bi, J. (2019). Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. *Proceedings of the IEEE 35th International Conference on Data Engineering*, 1358–1369.

Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L., & Li, S. (2011). IBAT: Detecting anomalous taxi trajectories from GPS traces. *Proceedings of the 13th International Conference on Ubiquitous Computing*, 99–108.

Zhang, Q., Wang, Z., Long, C., Huang, C., Yiu, S.-M., Liu, Y., Cong, G., & Shi, J. (2023). Online anomalous subtrajectory detection on road networks with deep reinforcement learning. *2023 IEEE 39th International Conference on Data Engineering*, 246–258.