Distribution-Based Trajectory Clustering

Zi Jing Wang[1], Ye Zhu[2], Kai Ming Ting[1]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2] Centre for Cyber Resilience and Trust, Deakin University, Australia
Email: wangzj@lamda.nju.edu.cn, ye.zhu@ieee.org, tingkm@nju.edu.cn

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00178

***Abstract*—Trajectory clustering enables the discovery of common patterns in trajectory data. Current methods of trajectory clustering rely on a distance measure between two points in order to measure the dissimilarity between two trajectories, causing problems of both effectiveness and efficiency. In this paper, we propose a new IDK-based clustering algorithm, called TIDKC, which makes full use of the distributional kernel for trajectory similarity measuring and clustering. TIDKC identifies non-linearly separable clusters with irregular shapes and varied densities in linear time. It does not rely on random initialisation and is robust to outliers. An extensive evaluation on 7 large real-world trajectory datasets confirms that IDK is more effective in capturing complex structures in trajectories than traditional and deep learning-based distance measures.**

***Index Terms*—Trajectory Clustering, Trajectory Distance Measure, Isolation Distributional Kernel**

I. Introduction

Clustering trajectories is an important task in trajectory data mining, which aims to compute the similarity among trajectories in a given dataset, and group trajectories in a given dataset into clusters of similar trajectories. However, trajectory clustering is a non-trivial problem, and it has two key challenges: existing trajectory distance measures usually have high time complexity and low fidelity, and existing trajectory clustering algorithms have either effectiveness issues or high time complexity.

No good solutions have been offered in the literature to meet all the above challenges. Although recent deep learning-based representations and clustering methods show potential improvements, we found that they have yet to produce a satisfactory outcome (see the experimental section for details).

Here we propose to use a recent Isolation Distributional Kernel (IDK) [19] as the main tool to meet all these challenges. IDK is used in the first instance to meet the first two challenges as a representation-cum-measure: a trajectory in $\mathbb{R}^d$ is mapped into a new representation as a point (vector) in the feature space of IDK; and the similarity between two trajectories can then be computed as a dot product of two mapped points in the feature space.

We also propose a new IDK-based clustering algorithm, called TIDKC, to handle trajectory clustering tasks in linear time[1]. TIDKC is able to identify non-linearly separable clusters with irregular shapes and varied densities. It also does not rely on random initialisation and is not sensitive to outliers.

The contributions of this paper are:

1) Proposing to use the distributional kernel IDK to represent trajectories and measure the similarity between two trajectories for clustering. IDK has high precision in retrieving the most similar trajectories because they are ranked based on similarity in distributions.
2) Developing a new clustering algorithm TIDKC that makes full use of the distributional kernel for trajectory clustering. It is the first clustering algorithm that employs two levels of the distributional kernel to perform trajectory clustering with linear time complexity.
3) Verifying the effectiveness and efficiency of the proposed IDK and TIDKC on 7 large real-world trajectory datasets. Our empirical results confirm that (a) IDK is more effective in capturing complex structures in trajectories; and (b) TIDKC produces better trajectory clustering outcomes and runs faster than existing methods.

The rest of the paper is organised as follows. Section II provides related works on trajectory distance measures and clustering. Section III and Section IV introduce IDK and TIDKC for trajectory distance measuring and clustering, respectively. Section V presents an empirical evaluation, followed by a conclusion in Section VI.

II. Related work

We survey methods for trajectory distance measures and trajectory clustering in the following two subsections.

A. Trajectory Distance Measures

Trajectory distance measurement computes the distance/dissimilarity between two trajectories. We categorise existing measures into three categories, point-based, distribution-based and deep learning-based methods as follows.

a) Point-based measures: Hausdorff distance, (Discrete) Fréchet distance and Dynamic Time Warping (DTW) distance [12] are point-based measures. Hausdorff distance and Fréchet distance operate on sets, where each trajectory is assumed to be a set of points $x \in \mathbb{R}^d$, while aims at finding the optimal matching of two trajectories by exploiting temporal distortions on trajectories. All the distance measures above have at least quadratic time complexity.

b) Distribution-based measures: Earth Mover's Distance (EMD) assumes that one trajectory is a set of identical and independent distributed (i.i.d.) points, sampled from a probability distribution $\mathcal{P}$. EMD [14], [24] employs a histogram to model a distribution.

[1]Code is at https://github.com/IsolationKernel/TIDKC. An extended version of this paper can be found at https://arxiv.org/abs/2310.05123.

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00178

In this paper, we propose to use the distributional kernel IDK [19] to represent trajectories and measure the similarity between two trajectories. Unlike existing distance measures mentioned earlier which have at least quadratic time complexity, IDK has unique data-dependent and injective properties with linear time complexity.

IDK is based on kernel mean embedding [11] which extends a point-to-point kernel κ to measure the similarity between two distributions. It maps a distribution $\mathcal{P}$ as a point in a reproducing kernel Hilbert space (RKHS) [11] via its feature map $\widehat{\Phi}$, representing each distribution $\mathcal{P}$ on data support $\mathcal{X}$ as a mean function:

$$\widehat{\Phi}(\mathcal{P}) = \int_{\mathcal{X}} \kappa(x, \cdot) \mathrm{d}\mathcal{P}, \tag{1}$$

where κ is a symmetric and positive definite kernel function.

Though both are distributional measures, IDK has three advantages over EMD in measuring the similarity of two trajectories. First, IDK computes the similarity without explicitly modelling a distribution. Second, the kernel mean map is injective [19]. This means that *similar trajectories are not to be ranked lower than dissimilar trajectories in search of the most similar trajectories*. EMD has yet to be shown to have such a property. Third, IDK has linear time complexity while EMD has at least quadratic time complexity.

The intuition is to treat each trajectory as a sample from an unknown multi-dimensional probability density function (pdf). With this treatment, any distributional measures for unknown multi-dimensional pdf, such as IDK and GDK, can be used to measure the similarity of two trajectories.

c) Deep learning-based methods: A deep neural network method, t2vec, learns a representation for trajectories [9]. It uses a sequence encoder-decoder model to learn from trajectories and generate a vector representation for each trajectory. This method is robust to non-uniform and noisy trajectory data and is able to compute the similarity of two trajectories in linear time. NeuTraj [25] trades off accuracy with efficiency by learning a distance function that approximates a given distance function such as DTW distance or Hausdorff distance. It reduces the time complexity of the original distance function from $O(n^2)$ to $O(n)$ where n is the length of each of the two trajectories under measurement. TrajGAT [26] is a recent improvement over NeuTraj.

B. Clustering Methods

Trajectory clustering algorithms can be generally divided into partition-based methods, density-based methods [21] and deep learning-based methods.

a) Partition-based methods: Partition-based methods divide trajectories into k non-overlapping clusters. Many extensions of K-Means and K-Medoids are proposed to deal with trajectory data clustering. Those clustering methods are usually computationally intractable for large trajectory datasets; thus, they often must reduce the number of trajectories or through some approximation [21], [23], [22]. Different from K-Means based clustering that can only detect globular clusters, Spectral clustering has the ability to handle clusters with arbitrary shapes [6], [10], [27]. The key problem for Spectral-based trajectory clustering is to construct the affinity graph that represents the pairwise relations between trajectories [7]. However, Spectral clustering has fundamental limitations, e.g., high computational complexity and difficulty in identifying clusters with different densities [13].

b) Density-based methods: Density-based algorithms usually identify dense segments and then connect these segments to form representative routes. This kind of algorithm discovers clusters with arbitrary sizes and shapes. For example, TRACLUS [8] aims to discover common sub-trajectories from a set of trajectories. Nevertheless, existing density-based clustering algorithms trade off accuracy and efficiency for clustering massive trajectories. Extracting exact sub-trajectories used to represent each cluster is an NP-hard problem [21].

c) Deep learning-based methods: Deep learning is attracting increasing interest and has made some progress for trajectory clustering in the last few years. There are new trends in trajectory generation, trajectory representation learning and similarity measurement [21], [15], [16]. E^{2}DTC [5] is an end-to-end deep trajectory clustering framework that requires no manual feature extractions. It employs an encoder-decoder scheme to learn the representation of trajectories, and then uses a self-training method to jointly learn the cluster-oriented representation and perform the clustering analysis.

III. Measuring trajectory distance using IDK

In this section, we investigate the key properties of IDK and other trajectory similarity measurements.

A clustering method requires a distance measure to group similar objects together. A good distance measure should produce a short distance to the neighbours of an object from the same cluster. We conduct MDS visualisation [2] and retrieval evaluation on a large Geolife dataset [5] to show the effectiveness of IDK as a trajectory measure in trajectory clustering tasks.

a) MDS Visualisation: Figure 1 shows the results using five distance measures. The IDK measurements congregate mapped points of trajectories of every cluster well. In contrast, each of GDK, Hausdorff distance and DTW distance splits some sparse clusters into two sub-clusters; Hausdorff distance produces quite a few scatter points; and the gaps between clusters are too close with DTW. t2vec has the worst result as there is no clear gap between any two clusters.

b) Retrieval evaluation: Here we assess the goodness of a trajectory distance measure in a trajectory retrieval task. Every trajectory in a dataset is used as a query, and we report the average precision of k nearest trajectories (precision@k) as the final result. The retrieval results in Figure 1 show that IDK has the best retrieval outcomes over all other measures for every k value.

IDK has been shown to be a better measure than GDK in point and group anomaly detection [19] and time series anomaly detection [18]. Our result in trajectory retrieval is consistent with these results.

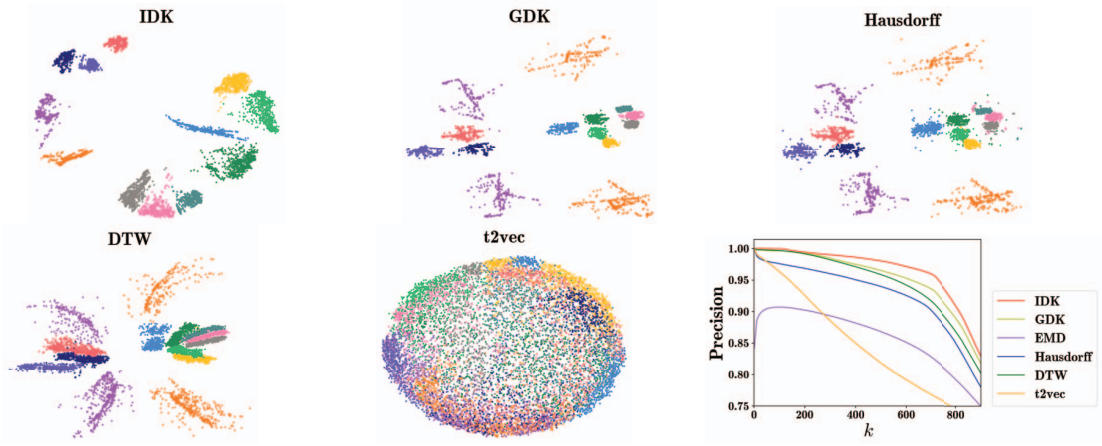


Fig. 1: The MDS visualization results and Precision@k with different distance measures on the Geolife dataset.

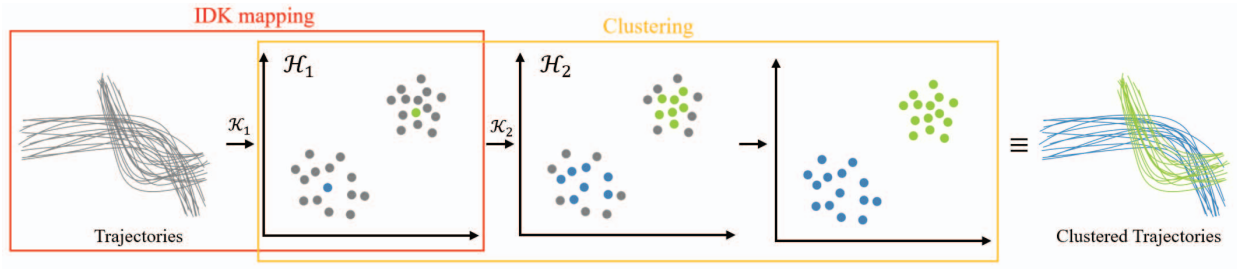


Fig. 2: The top-level view of TIDKC trajectory clustering based on IDK.

IV. Isolation Distributional Kernel Clustering

In this section, we propose a Trajectory clustering using IDK, called TIDKC (Trajectory Isolation Distributional Kernel Clustering). The overview of the clustering procedure is presented in Figure 2. The detailed procedure is provided in Algorithm 1 for a given dataset of trajectories $D = \{T_1, \dots, T_n\}$. TIDKC uses two levels of IDK. The level-1 IDK $\mathcal{K}_1$ is derived from $S = \cup_i T_i$, and its feature map $\widehat{\Phi}_1(\cdot|S)$ is used to map each trajectory T_i to a level-1 point $g_i = \widehat{\Phi}_1(\mathcal{P}_{T_i}|S) = \frac{1}{|T_i|}\sum_{x \in T_i} \phi(x|S)$ in RKHS $\mathcal{H}_1$ to yield $G = \{g_1, \dots, g_n\}$ (line 1 in Algorithm 1), where $\phi(\cdot|S)$ is the feature map of Isolation Kernel [20].

The clustering, in the following step, is performed on set G by growing each cluster (of level-1 points in G) via level-2 IDK $\mathcal{K}_2$ in RKHS $\mathcal{H}_2$. The idea is to identify k seeds to initialize the k clusters C_j, represented as distribution P_{C_j}, which are then grown up to a similarity threshold τ. As the clusters have grown, they are used as the new basis to grow further to a lower threshold. This repeats until all points in G are assigned to one of the k clusters. Lines 5 to 9 in Algorithm 1 show the procedure of the clustering.

The final clustering outcome is clusters E_j of trajectories which correspond to clusters C_j of the level-1 IDK-mapped points in RKHS $\mathcal{H}_1$ for $j = 1, \dots, k$.

Clustering objection function: Given a dataset of level-1 mapped points $G = \{g_1, \dots, g_n\}$, TIDKC aims to find clusters

Algorithm 1: TIDKC: Trajectory clustering using IDK

Input: $D = \{T_1, \dots, T_n\}$: trajectory dataset, k: number of clusters, $\varrho \in (0, 1)$: growth rate of clusters
Output: $C = \{E_1, \dots, E_k\}$: k Clusters.

1: Map each trajectory $T_i \in D$ to a point $g_i = \widehat{\Phi}_1(\mathcal{P}_{T_i})$ in RKHS $\mathcal{H}_1$ using feature map $\widehat{\Phi}_1$ of $\mathcal{K}_1$ to yield $G = \{g_1, \dots, g_n\}$
2: Select k cluster seeds $c_j \in G$ based on local-contrast estimation; and initialize clusters $C_j = \{c_j\}, j = 1, \dots, k$
3: Initialize $N \leftarrow G \setminus \cup_j C_j$
4: Initialize $\tau \leftarrow \max_{g \in N, \ell \in [1,k]} \mathcal{K}_2(\delta(g), \mathsf{P}_{C_\ell})$
5: **repeat**
6: $\quad \tau \leftarrow \varrho \times \tau$
7: $\quad$ Expand cluster C_j to include unassigned point $g \in N$ for $j = \arg\max_{\ell \in [1,k]} \mathcal{K}_2(\delta(g), \mathsf{P}_{C_\ell})$ and $\mathcal{K}_2(\delta(g), \mathsf{P}_{C_j}) > \tau$
8: $\quad N \leftarrow G \setminus \cup_j C_j$
9: **until** $|N| = 0$ or $\tau < 0.00001$
10: Assign each unassigned point g to nearest cluster C via $\mathcal{K}_2(\delta(g), \mathsf{P}_C)$
11: Cluster $E_j \subset D$ corresponds to $C_j \subset G, j = 1, \dots, k$

$C_j, j = 1, \dots, k$ satisfying the following objective function using level-2 $\mathcal{K}_2$:

$$\arg\max_{C_1, \dots, C_k} \sum_{j=1}^{k} \sum_{g \in C_j} \mathcal{K}_2(\delta(g), \mathsf{P}_{C_j}|G), \tag{2}$$

where $\delta(\cdot)$ is a Dirac measure and P_C is the distribution that generates the set C of points in RKHS $\mathcal{H}_1$.

Note that TIDKC is developed based on IDKC [28] for

TABLE I: Time complexities of five main processes in TIDKC. t and ψ are parameters of Isolation Kernel [20]. d, k and b are numbers of dimensions, clusters and iterations, respectively.

1.	Build Isolation Kernel (IK)	$\mathcal{O}(dt\psi)$
2.	Getting the feature map	$\mathcal{O}(ndt\psi)$
3.	Cluster seeds selection	$\mathcal{O}(s^2t\psi)$
4.	Growing k clusters	$\mathcal{O}(bknt\psi)$
5.	Final clusters assignment	$\mathcal{O}(knt\psi)$
Total time cost for TIDKC		$\mathcal{O}((s^2+n(bk+d))t\psi)$

TABLE II: Real-world datasets. Ave $|T|$ indicates the average length of trajectories.

Dataset	#Points	#Traj	min–max $\|T\|$	Avg $\|T\|$	#Clusters
VRU_pedes_3	202,272	610	196–620	332	3
VRU_pedes_4	238,443	710	196–620	336	4
VRU_cyclists	76,051	265	52–1,257	287	3
TRAFFIC	15,000	300	50–50	50	11
CROSS	24,420	1,900	5–23	13	19
CASIA	147,883	1,500	16–612	96	15
Geolife	620,477	9,192	24–100	68	12

TABLE III: Clustering outcomes in terms of NMI.

	K-Medoids clustering					Spectral clustering					TIDKC		TGDKC		E^2DTC
	IDK	GDK	HD	DTW	t2vec	IDK	GDK	HD	DTW	t2vec	IDK	GDK	IDK	GDK	
VRU_pedes_3	.74	.72	.64	.52	.48	.85	.83	.85	.55	.58	.94	**.97**	.94	.80	.69
VRU_pedes_4	.55	.51	.40	.50	.41	.57	.60	.67	.62	.51	**.84**	.80	.70	.65	.51
VRU_cyclists	.75	.46	.64	.75	.56	.74	.58	**.94**	.74	.64	.83	.67	.61	.66	.86
TRAFFIC	.91	.91	.95	.96	.72	.96	.97	.69	.77	.61	**1.00**	.98	**1.00**	.96	.95
CROSS	.98	.98	.97	.97	.97	.97	.95	.85	.61	.73	**.99**	.98	.95	.95	.91
CASIA	.91	.92	.90	.92	.80	**.94**	.92	.86	.85	.83	.91	.90	.89	.89	.76
Geolife	.92	.89	.75	.82	.86	.95	.89	.93	.97	.90	**.99**	.96	.96	.97	.79
Average NMI	.81	.76	.73	.77	.70	.83	.80	.80	.73	.71	**.91**	.86	.85	.83	.80

trajectory clustering. IDKC is the first clustering algorithm that employs an adaptive distributional kernel without any optimisation. IDKC is able to identify complex clusters on massive datasets, but it works on multi-dimensional data points only. The key additional step is line 1 in Algorithm 1.

Table I presents the time complexities of the key procedures of TIDKC. Building IK and mapping n points using IK's feature map cost $\mathcal{O}(nd\psi t)$, where $\psi \ll n$. The cluster seeds selection can be based on a subset of fixed size $s \ll n$ for a large dataset. The cluster growing process is based on the distributional kernel with $\mathcal{O}(n)$. The number of iterations for lines 5 to 9 in Algorithm 1 is bounded by the growth rate (ϱ in line 6). Thus, the total time complexity of TIDKC is linear to the dataset size n since all other parameters are constant.

V. Empirical Evaluation

A. Experimental design and settings

In this section, we aim to answer the following questions:

1) Is distributional kernel an effective means to represent trajectories and compute the similarity between trajectories in clustering tasks?
2) Does TIDKC have any advantage over other clustering methods on trajectory clustering, in terms of clustering performance and time complexity?

To answer the first question, we compare IDK and GDK with traditional trajectory distance measures[2], including Hausdorff distance [17], DTW distance [1] and the deep learning method t2vec [9]. They are evaluated in the same clustering algorithm. To answer the second question, we compare TIDKC and TGDKC[3] with three existing clustering methods, i.e., K-Medoids clustering [3], Spectral clustering [3] and an end-to-end deep learning clustering method E^2DTC [5]. We search their parameters following the original papers and report their best performances. As both TIDKC and TGDKC require a multi-dimensional representation of trajectories and do not accept a distance matrix as input, only IDK and GDK are used for trajectory similarity measurements.

Table II presents the datasets used in the experiments. Most of the datasets are trajectories of vehicles or pedestrians recorded by GPS or surveillance cameras. They vary in size, lengths of trajectories and number of clusters.

B. Clustering performance

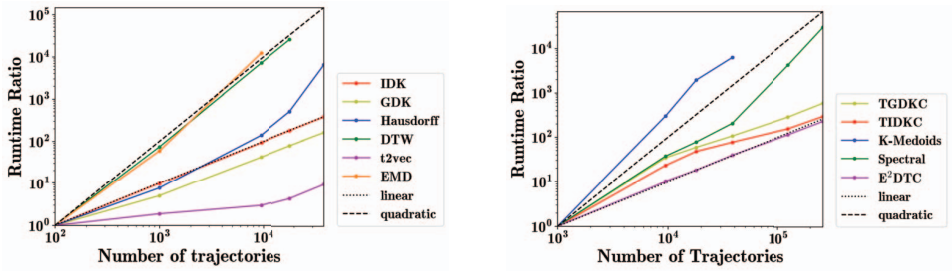
The clustering performances of five clustering methods using different distance measures are shown in Table III in terms of NMI (Normalised Mutual Information) [4].

We have the following observations:

1) Comparing the five distance measures, IDK has the best average NMI using either K-Medoids or Spectral clustering.
2) Deep representing method t2vec performed the worst on most datasets using either K-Medoids or Spectral clustering. It has mixed results between these two clustering methods on different datasets.
3) TIDKC with either IDK or GDK outperformed all other clustering methods on almost all datasets. The end-to-end deep learning method, E^2DTC, has some improvement over Spectral clustering and K-Medoids, only if the latter two use DTW and t2vec; but E^2DTC always performed worse than TIDKC with one exception.

[2]Fréchet and Earth mover's distance are omitted because they took extremely long time to run when a trajectory has a large number of points.

[3]To produce TGDKC (Trajectory Gaussian Distributional Kernel Clustering), IDK is replaced with GDK as $\mathcal{K}_2$ in Algorithm 1, where the cluster growing step is guided by GDK instead.



(a) Scaleup test for different distance/kernel measures. (b) Scaleup test for different clustering methods.

Fig. 3: Scaleup tests. All measures and clustering methods ran on CPU, except t2vec and E^{2}DTC which ran on GPU. All clustering methods use IDK as the trajectory representation and distance measurement, except E^{2}DTC.

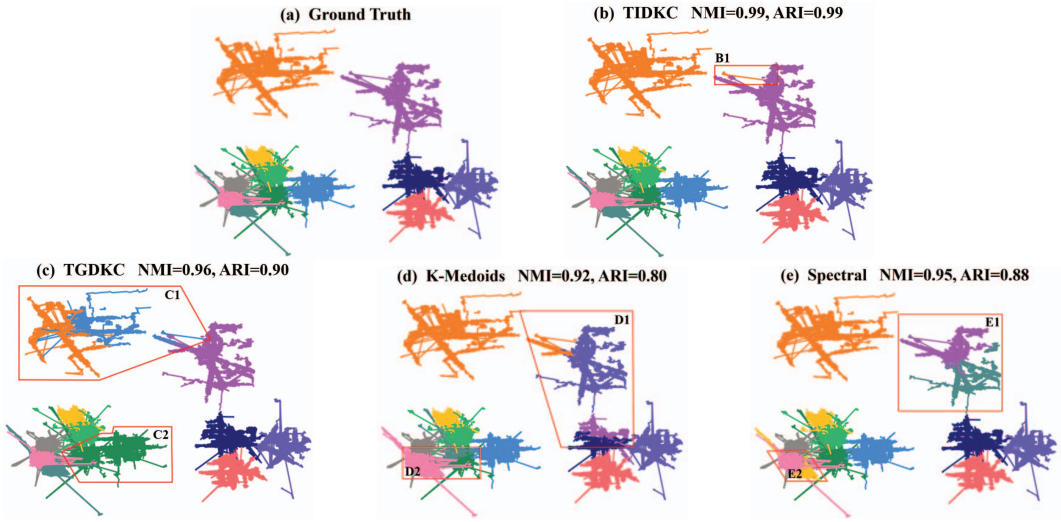


Fig. 4: Clustering outcomes of clustering methods using the same representation of IDK on the Geolife dataset. Areas with clustering errors are indicated by the red boxes.

In a nutshell, TIDKC, in combination with IDK to represent and compute the similarities of trajectories, offers the best method for trajectory clustering.

C. Scaleup tests

We conduct scale-up tests on both distance measures and clustering methods[4]. The dataset of scaleup tests is constructed based on the Geolife dataset. Figure 3a shows the scaleup test result of different distance measures. Because of the use of GPU, t2vec has the lowest runtime ratio. Apart from t2vec, GDK with the Nyström method, has the best scaleup ratio, followed by IDK with linear runtime. Hausdorff and DTW distances have at least quadratic runtime, making them the slowest measures.

Figure 3b shows the scaleup test result of different clustering methods. K-Medoids is the slowest with quadratic runtime. Spectral clustering also has super-linear time complexity, especially when the dataset is large. TIDKC and TGDKC have similar linear runtime ratios with TIDKC being slightly faster. TIDKC and TGDKC have slightly different runtimes though they use the same algorithm. This is because the runtime of TGDKC or TIDKC depends on the number of iterations required to complete the task, mainly due to the kernel (GDK or IDK) employed. E^{2}DTC has the lowest runtime ratio with the use of GPU.

[4]Multi-thread processing is applied in Hausdorff and DTW distances, but not others. All ran in CPU, except that t2vec and E^{2}DTC ran in GPU.

D. Clustering Visualisation

We provide exactly the same IDK similarities on the Geolife dataset to TIDKC, TGDKC, K-Medoids and Spectral clustering to investigate their clustering capabilities, which is shown in Figure 4. Any clustering mistakes are a direct result of a clustering method since the trajectory similarities on this dataset, provided by IDK, enable an ideal clustering method to produce a perfect clustering outcome.

Our observations are given as follows:

1) TIDKC does very well, clustering almost every trajectory correctly, except for the trajectory in region B1.
2) TGDKC is less accurate in dealing with clusters of different densities, compared to TIDKC. For clusters in

sparse regions like region C1 in subfigure (c), TGDKC tends to divide the points in the same cluster into different subclusters. For dense clusters, TGDKC might merge two different clusters into one, as shown in region C2 in subfigure (c).

3) K-Medoids clustering relies heavily on the initialization of centroids and can only handle the clustering of globular shapes (in Hilbert space induced by IDK). As a result, it splits one cluster into two subclusters for two clusters in region D1; and merges two clusters into one in region D2, as shown in subfigure (d).
4) Spectral clustering has clustering errors in region E1 and region E2, shown in subfigure (e). These errors can be attributed to the fundamental limitation of SC identified previously (see [13] for details).

VI. Conclusion

In this paper, we found that IDK is a powerful tool for trajectory representation and similarity measurement as well as trajectory clustering. The proposed TIDKC makes use of distributions at two levels of IDK. The first level IDK is used to represent each trajectory as a distribution. The second level IDK is used to represent the distribution of a cluster of trajectories, enabling clusters with irregular shapes and varied densities to be discovered.

Our extensive evaluation confirms that IDK is more effective in capturing complex structures in trajectories than traditional and deep learning-based distance measures, such as DTW, Hausdorff distance and t2vec. Furthermore, the proposed TIDKC has superior clustering performance in terms of NMI than existing trajectory clustering algorithms, including K-Medoids clustering, Spectral clustering, TGDKC and E^{2}DTC. With linear runtime, IDK is faster than point-based trajectory distance measures like DTW, Hausdorff distance and distribution-based method EMD. TIDKC runs orders of magnitude faster than other methods except deep learning-based that was accelerated by GPU.

Acknowledgements

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

References

[1] Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Foundations of Data Organization and Algorithms, pp. 69–84. Springer Berlin Heidelberg (1993)
[2] Borg, I., Groenen, P.J., Mair, P.: Applied multidimensional scaling. Springer Science & Business Media (2012)
[3] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
[4] Cover, T.M.: Elements of information theory. John Wiley & Sons (1999)
[5] Fang, Z., Du, Y., Chen, L., Hu, Y., Gao, Y., Chen, G.: E^{2}DTC: An end to end deep trajectory clustering framework via self-training. In: Proceedings of the 37th IEEE International Conference on Data Engineering. pp. 696–707 (2021)
[6] Gao, M., Shi, G.Y.: Ship-handling behavior pattern recognition using AIS sub-trajectory clustering analysis based on the t-SNE and spectral clustering algorithms. Ocean Engineering **205**, 106919 (2020)
[7] Hung, C.C., Peng, W.C., Lee, W.C.: Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. The VLDB Journal **24**, 169–192 (2015)
[8] Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 593–604 (2007)
[9] Li, X., Zhao, K., Cong, G., Jensen, C.S., Wei, W.: Deep representation learning for trajectory similarity computation. In: Proceedings of the 34th IEEE International Conference on Data Engineering. pp. 617–628. IEEE (2018)
[10] Liu, C., Torralba, A., Freeman, W.T., Durand, F., Adelson, E.H.: Motion magnification. ACM Transactions on Graphics **24**(3), 519–526 (2005)
[11] Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al.: Kernel mean embedding of distributions: A review and beyond. Foundations and Trends in Machine Learning **10**(1-2), 1–141 (2017)
[12] Myers, C., Rabiner, L., Rosenberg, A.: Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **28**(6), 623–635 (1980)
[13] Nadler, B., Galun, M.: Fundamental limitations of spectral clustering. Advances in Neural Information Processing Systems **19** (2006)
[14] Rubner, Y., Tomasi, C., Guibas, L.J.: A metric for distributions with applications to image databases. In: Proceedings of the Sixth International Conference on Computer Vision. pp. 59–66 (1998)
[15] Schreck, T., Bernard, J., Von Landesberger, T., Kohlhammer, J.: Visual cluster analysis of trajectory data with interactive kohonen maps. Information Visualization **8**(1), 14–29 (2009)
[16] Song, X., Kanasugi, H., Shibasaki, R.: Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 2618–2624 (2016)
[17] Taha, A.A., Hanbury, A.: An efficient algorithm for calculating the exact Hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(11), 2153–2163 (2015)
[18] Ting, K.M., Liu, Z., Zhang, H., Zhu, Y.: A new distributional treatment for time series and an anomaly detection investigation. Proceedings of the VLDB Endowment **15**(11), 2321–2333 (2022)
[19] Ting, K.M., Xu, B.C., Washio, T., Zhou, Z.H.: Isolation distributional kernel: A new tool for point and group anomaly detections. IEEE Transactions on Knowledge and Data Engineering **35**(03), 2697–2710 (2023)
[20] Ting, K.M., Zhu, Y., Zhou, Z.H.: Isolation kernel and its effect on svm. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 2329–2337 (2018)
[21] Wang, S., Bao, Z., Culpepper, J.S., Cong, G.: A survey on trajectory data management, analytics, and learning. ACM Computing Surveys **54**(2), 1–36 (2021)
[22] Wang, S., Bao, Z., Culpepper, J.S., Sellis, T., Qin, X.: Fast large-scale trajectory clustering. Proceedings of the VLDB Endowment **13**(1), 29–42 (2019)
[23] Xu, H., Zhou, Y., Lin, W., Zha, H.: Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4328–4336 (2015)
[24] Yang, W., Wang, S., Sun, Y., Peng, Z.: Fast dataset search with earth mover's distance. Proceedings of the VLDB Endowment **15**(11), 2517–2529 (2022)
[25] Yao, D., Cong, G., Zhang, C., Bi, J.: Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In: Proceedings of the 35th IEEE International Conference on Data Engineering. pp. 1358–1369. IEEE (2019)
[26] Yao, D., Hu, H., Du, L., Cong, G., Han, S., Bi, J.: TrajGAT: A graph-based long-term dependency modeling approach for trajectory similarity computation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 2275–2285 (2022)
[27] Zhang, Z., Huang, K., Tan, T., Yang, P., Li, J.: Red-sfa: Relation discovery based slow feature analysis for trajectory clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 752–760 (2016)
[28] Zhu, Y., Ting, K.M.: Kernel-based clustering via isolation distributional kernel. Information Systems **117**, 102212 (Jul 2023)