Doctor of Philosophy

Detecting Anomalous Human Trajectories in Indoor Spaces

The Graduate School of the University of Ulsan

Department of Electrical, Electronic and Computer Engineering

DOI THI LAN

Detecting Anomalous Human Trajectories in Indoor Spaces

Supervisor: Prof. Seokhoon Yoon

A Dissertation

Submitted to
the Graduate School of the University of Ulsan
In partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Doi Thi Lan

Department of Electrical, Electronic and Computer Engineering

University of Ulsan, Korea

December 2023

Detecting Anomalous Human Trajectories in Indoor Spaces

This certifies that the dissertation of Doi Thi Lan is approved by:

Committee Chair: Prof. Jin-Ho Chung

Committee Member: Prof. Daehwan Kim

Committee Member: Prof. DaeHan Ahn

Committee Member: Dr. Woo-Sung Jung

Committee Member: Prof. Seokhoon Yoon, *Adviser*

Department of Electrical, Electronic and Computer Engineering

University of Ulsan, Korea

December 2023

Dedicated to

My beloved husband, sons, parents, parents-in-law, sister, brother, and cousin…

Acknowledgments

First, I would like to express my deep and sincere gratitude to my advisor, Prof. Seokhoon Yoon of the Department of Electrical, Electronic and Computer Engineering, University of Ulsan, for his dedicated guidance during my research. Due to my Professor's valuable comments and suggestions, I learned how to study successfully. Besides, under his guidance, I have accumulated a lot of valuable experience and knowledge and improved my carefulness and logical thinking. Without his support and guidance, I could not have been able to fulfil my research at the University of Ulsan.

Additionally, I would like to thank the committee members for reviewing my thesis and giving insightful comments that helped me improve the quality of my thesis. I also thank my lecturers and colleagues at Le Quy Don Technical University in Vietnam, who always support and encourage me to do research.

Next, I would like to thank my parents, parents-in-law, sister, brother, and friends for their support during my studies. Besides, I am thankful to the members of the Advanced Mobile Networks and Intelligence Systems Laboratory, who created a friendly working environment and supported me a lot during my studies at the Ulsan University.

Finally, I would like to express my deep gratitude to my husband and two little sons. They sacrificed a lot for me during my time away from home. Besides, I am very grateful to my cousin, who supported my husband care for my children. Their love and sacrifice inspired and motivated me to fulfill my research at the University of Ulsan.

Abstract

Anomalies in indoor human trajectories often relate to urgent situations (e.g., violent attacks, terrorism and fire). Detecting anomalous human trajectories can improve safety and instantly handle risks in working spaces. Therefore, this thesis studies methods for anomaly detection in indoor human trajectories. In particular, we first study abnormal indoor human trajectory types, which may occur in real-world environments. Next, a framework for detecting anomaly trajectories based on the density method is proposed. In addition, this thesis develops a trajectory clustering-based anomaly detection framework. Finally, we extend this work by studying a deep learning model to learn trajectory characteristics for detecting anomalies.

First, anomaly types of indoor human trajectories are studied. In particular, anomalous trajectories are essential for evaluating anomaly detection methods. However, since anomalous trajectories may not be available in datasets, they are needed to create for evaluation. In the literature, there are often two methods: giving a hypothesis for anomalies, and synthesizing anomalies and injecting them into datasets. In the first method, anomalies are assigned based on the difference in occurrence frequency and movement behaviour between trajectory groups in datasets. In the second, anomaly trajectory types are generated and injected into the datasets for evaluation.

Secondly, a density method-based framework is proposed for detecting anomalous trajectories. A trajectory is normal if it is near to other trajectories in datasets. In other words, the number of its neighbour trajectories is high. Thus, this framework detects anomalous trajectories based on their densities in datasets. In particular, two trajectories are neighbour to each other if their distance is smaller than a distance threshold. In detecting anomalies, a trajectory is detected as an anomaly if its density is smaller than a density threshold. We propose a new method for determining distance and density thresholds.

The next chapter proposes a framework based on a clustering algorithm to detect abnormal indoor trajectories. In this framework, the abnormality of a trajectory is determined based on the relationship between it and clusters in datasets. First, a distance metric is proposed to improve the accuracy of distance metric for indoor human trajectories. Then, the Epsilon parameter of Density-Based Spatial Clustering of Application with Noise (DBSCAN) is determined using a new DCVI metric. To find normal trajectory clusters in the dataset, DBSCAN is used. In detecting anomalous trajectory, if a trajectory does not belong

to any clusters, it is marked as an anomaly.

To take advantage of the power of neural networks in anomaly detection tasks, a Transformer encoder and self-organizing map-based deep learning model called TENSO is proposed to learn trajectory characteristics. In particular, to learn internal characteristics of normal trajectories, the Transformer encoder with the self-attention mechanism is used. In addition, to learn clusters of normal trajectory representations in latent space, the self-organizing map (SOM) is used. In the training phase, the TENSO model is trained using a total loss of trajectory reconstruction and SOM losses. In the anomaly detection phase, a test trajectory is evaluated to determine whether it is an anomaly based on trajectory reconstruction errors and the quantization error on the SOM.

Finally, proposed frameworks are evaluated using two real trajectory datasets: MIT Badge and sCREEN. The results show that the proposed frameworks detect trajectory anomalies effectively, especially the TENSO model-based framework.

Contents

List of Figures

List of Tables

Chapter 1

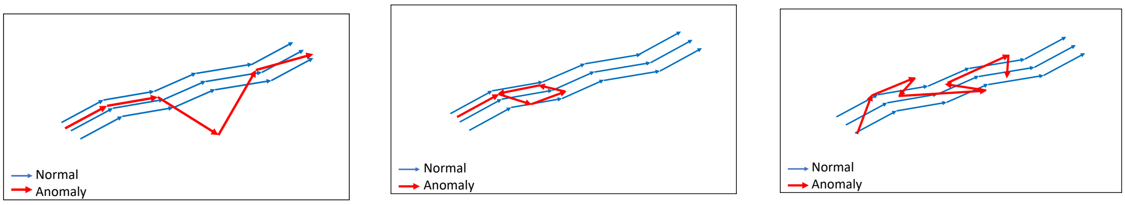
Introduction

1.1 Anomalous Indoor Human Trajectory Detection

Anomalous trajectory detection is one of the crucial tasks for a lot of applications. For taxi services, for example, abnormal trajectories are related to issues like traffic congestion, taxi driving fraud, and refusing to take passengers. Therefore, detecting anomalous taxi trajectories may improve the performance of this service [1–7]. Moreover, anomaly event detection plays a vital role in security surveillance in public spaces. The anomaly events, such as terrorism, and accidents, can be detected by analyzing the object trajectories in public places [8, 9]. In addition, to guarantee the safety and security of ships during a voyage, their location data are used to detect outlying trajectories and remind other ships to take the necessary avoidance actions [10–14]. Similarly, to ensure aircraft safety, data recorded from current flights are monitored and analyzed constantly. Once anomalous data patterns are detected, they are informed to the flight monitoring system to handle them instantly [15–19].

With indoor areas, trajectory data have opened many new research directions. In particular, customers' trajectories in shops can be investigated to determine their purchasing habits [20]. This enables owners to optimize product positioning and shop design. Human location prediction systems for indoor environments play a significant role in location-based services [21–26]. Additionally, abnormal human movements in indoor spaces often relate to serious situations like violent attacks, theft, and fire. Thus, detecting abnormal movements by people can improve safety in indoor environments. In this thesis, our objective is to identify abnormal human trajectories in indoor spaces.

In particular, this work studies three anomaly types (i.e., rare location visiting, wandering and route anomalies). A rare location is where humans rarely visit or are prohibited from visiting. Therefore, a person's behaviour may be considered abnormal if they access rare places. In addition, wandering anomalies can occur when people wander around objects with bad intentions. Route anomaly can be created when people take unusual routes to escape in urgent situations (e.g., fire and attacking violence). These anomaly types are illustrated in Figure 1.1.



(a) Rare location visiting anomaly. (b) Wandering anomaly. (c) Route anomaly.

Figure 1.1: Illustration of anomalous trajectory types.

There have been a lot of methods proposed to detect abnormal trajectories. However, detecting abnormal indoor human trajectories in urgent situations has a tight time constraint. Thus, this task requires anomaly detection as fast as possible to handle the occurring situations instantly. Existing methods that detect anomalous trajectories with no time limits are unsuitable [2, 27–29]. Thus, the thesis first proposed a density-based framework, which detects anomalies in a short time to satisfy time requirements. To do this, a time window is chosen to process trajectories. The window size is small enough to meet the time constraint. In addition, choosing anomaly threshold is based on the distribution of trajectories in this framework.

Next, trajectories in indoor spaces differ from those in outdoor spaces. For example, indoor trajectories are limited by rooms, corridors, and stairs. Therefore, anomaly detection methods proposed for outdoor trajectories are ineffective for indoor trajectory data [2, 30]. From the above aspects, the thesis develops a trajectory clustering-based framework to improve the performance of abnormal indoor human trajectory detection. Firstly, a distance metric for indoor trajectories is proposed. Then, the Epsilon (Eps) parameter of DBSCAN is chosen using a new cluster validation metric (DCVI). Anomaly detection is performed based on the relationship between test trajectory and trajectory clusters in datasets.

Finally, we propose a deep trajectory learning model-based detection method that combines traditional machine learning (clustering-based) and deep learning methods for detecting anomalies. Specifically, the proposed model learns internal trajectory characteristics using the Transformer encoder and clusters of trajectory representation in latent space using a self-organizing map (SOM). Anomaly detection is performed using the trajectory reconstruction errors and the quantization error on the SOM. Moreover, a mechanism for determining an appropriate anomaly threshold is proposed in our method, which is not provided in studies [2,27,30,31].

In evaluating the proposed frameworks, two real trajectory datasets are used: MIT Badge and sCREEN datasets. Since truth anomalies are not available in these dataset, they are generated for evaluation. Therefore, the need to be made for evaluation. There are two usual ways to generate anomalies for datasets in the literature. In the first method, a hypothesis is given based on the difference between trajectory groups. The second method is to generate anomalies and to inject them into datasets. The results show that the proposed frameworks effectively detect trajectory anomalies. Specifically, the framework based on the TENSO model outperforms existing baselines in terms of performance metrics.

1.2 Creating Anomalous Indoor Human Trajectory Types

This subsection briefly describes the two methods for creating anomalies in datasets.

First, a hypothesis is given for anomalies based on the difference in behavior between groups in the dataset [32,33]. The hypothesis is that if one group occurs more frequently than the other, this group is described as a normal group. The remaining group with less occurrence is described as an anomalous group.

The second way generates anomalies and injects them into datasets [34,35]. This study introduces three anomaly types: rare location visiting, wandering, and route anomalies, which can occur in real-world environments. The way for creating and injecting anomalies into datasets is detailed presented in Chapter 3.

1.3 Proposed Anomalous Trajectory Detection Frameworks

Anomaly trajectory detection methods can divided into two main categories: traditional method-based anomaly detection and deep learning model-based anomaly detection. The thesis uses both traditional method and deep learning model to develop anomalous trajectory detection frameworks.

1.3.1 Anomalous Human Trajectory Detection based on Traditional Methods

In the traditional detection methods, the characteristics of the trajectory (e.g., distance from other trajectories, density, and the relationship with clusters in the dataset) are discovered to detect anomalies. In this work, we first develop a framework to detect abnormal human trajectories using the edit distance on real sequence (EDR) and density method. Then, a trajectory clustering-based anomaly detection framework is proposed.

The trajectory outlier detection algorithm based on distance function and density method is introduced in [27, 31]. The authors proposed different distance measurements that are applied to calculate the similarity between two trajectories. In their algorithms, two parameters: distance threshold and neighbor density threshold are used to detect trajectory anomalies. However, they have not provided a method for choosing the parameter values that should be determined based on the dataset's distribution characteristics. Therefore, in our work, a method for determining the parameters based on trajectory data is proposed. Specifically, in this thesis, the density-based anomaly detection framework consists of two phases. In the first phase, we design a method to determine the algorithm parameters: distance threshold and density threshold using collected trajectories. In the second phase, a test trajectory is detected as normal or abnormal. To achieve this objective, neighbor density of the input trajectory is calculated using the distance threshold. Then, the test trajectory is marked as an anomaly if its density is less than the density threshold.

With the clustering-based framework, a distance metric is required to calculate the similarity between indoor trajectories. The distance between trajectories in indoor spaces is different from those in outdoor spaces because indoor trajectories are limited by entities, such as rooms, corridors, and stairs. Therefore, anomaly detection methods using existing distance functions, including Euclidean distance, longest common sub-sequence (LCSS) [36], dynamic time warping (DTW) [37], and edit distance on real sequence (EDR) [38], are ineffective for indoor trajectory data. In addition, indoor human movement behavior may be represented by location traces and the semantics of data points. For example, consider the behavior between two people. One stays in a meeting room, the other in a coffee room. Assume that these two rooms are next to each other. Their behavior is entirely different in this case, despite their close proximity. Therefore, semantic information should be considered when estimating the similarity between indoor trajectories. In this work, a new metric is proposed, which discovers both indoor walking distance and semantic information to determine the similarity between two trajectories.

In addition, clustering-based anomaly detection methods require input parameters for clustering algorithms, such as the number of clusters in K-Means, spectral clustering, and hierarchical clustering. With DBSCAN, two input parameters are required: the minimum number of points to form a new cluster (MinPts) and the radius to find the neighbors (Eps). Choosing an appropriate value of Eps is essential for performing DBSCAN; most studies determined this value manually. In this work, a DBSCAN cluster validity index (DCVI) is proposed to choose an appropriate value of Eps automatically . After the parameters are determined, DBSCAN is applied to find trajectory clusters in datasets. In detecting anomalies, if a trajectory does not belong to any clusters, it is marked as an anomaly.

1.3.2 Anomalous Human Trajectory Detection based on Deep Learning Model

Deep learning-based detection methods often focus on learning embedding vectors that capture spatial-temporal information in trajectories. Detecting anomalies is based on the embedding vectors [39–46]. However, these methods do not discover relationships between trajectory representations (e.g., distance, density, and clusters of trajectory representations) in latent space to use them for detecting abnormal trajectories. From the above aspects, we aim at building a novel deep learning model that learns normal trajectories' interior features and clusters of normal trajectory representations in latent space to detect anomalies.

In particular, a deep learning model called TENSO, which is based on the Transformer encoder and self-organizing map (SOM), is proposed in this work. To capture internal characteristics of normal trajectories in a latent space, the Transformer encoder is first used. Each trajectory is encoded by an output sequence that captures the correlation between points in the trajectory through a self-attention mechanism. To learn more helpful information in latent space, the TENSO model discovers both positional information of trajectories' points and their indoor semantic information (e.g., working rooms, meeting rooms, and corridors). Then, a decoder reconstructs the trajectories from their representations in the latent space.

To learn the clusters of normal trajectory representations in latent space, a SOM layer is used in this work. The SOM is a data clustering model containing interconnected neurons on a low-dimensional map [47–51]. Each neuron is represented by a prototype vector. Note that prototype vectors belong to the same space as the input data for the SOM. While training the SOM, the prototype vectors are updated and forwarded to input data. In TENSO, the SOM input data space contains latent representations of trajectories, that are the output of the Transformer encoder. Besides, the number of prototype vectors is chosen much less than the number of trajectories in this work. Thus, if trajectory representations are close to each other in the latent space, they are represented by the same prototype vector on the SOM. To train the TENSO model, a total loss of the trajectory reconstruction and SOM losses is used. After the model is trained, anomaly detection is performed when a new trajectory comes. In particular, the anomaly score (AS) of a trajectory is determined using the TENSO model. If the AS exceeds a given threshold, the trajectory is detected as an anomaly. In this work, the anomaly threshold is determined based on trajectories' anomaly scores in the training set. A new factor, the weighted sum (WS) of precision and recall, is proposed to choose the threshold value.

1.4 Evaluation Methods

This thesis aims at proposing anomaly detection frameworks in indoor human trajectories. To evaluate the proposed frameworks, two real trajectory datasets are used: MIT Badge [52] and sCREEN [20].

With the anomaly detection frameworks based on density and clustering methods, the datasets are divided into two parts. In particular, in the density-based framework, the first part is used to determine distance and density thresholds. Meanwhile, this part is used to find trajectory clusters in the clustering-based framework. The second part is used to evaluate the anomaly detection accuracy of both frameworks.

With the deep learning-based framework, the datasets are divided into three sets: the training set for training the proposed model, the validation set for choosing the best model, and the test set for evaluating results of the proposed framework.

In addition, three performance metrics are used in this thesis: recall, precision, and f1-score. Recall represents the anomaly detection ability of methods, and precision is the correction of detection methods. F1-score is the harmonic mean of precision and recall.

1.5 Thesis Organization

This dissertation is presented as follows. First, Chapter 2 starts by discussing related works. Then, Chapter 3 presents the methods for creating anomalous trajectory types. Chapter 4 provides a detailed introduction to the density-based framework. A detailed description of the clustering-based framework is presented in Chapter 5. Chapter 6 introduces a framework based on a deep learning model. Finally, in Chapter 7, we conclude this thesis by summarizing our contributions and discussing future works.

Chapter 2

Background and Related Works

2.1 Distance Metrics for Trajectory Data

A lot of distance functions are used to determine the level of similarity between two trajectories [53]. Euclidean distance measures the total distance between all pairs of corresponding points between two trajectories. However, Euclidean distance requires the same length of two considering trajectories, and it is sensitive to noises or equipment recording errors. Several other distance measurements have been proposed to improve Euclidean distance. For instance, LCSS in [36] can find similar common subsequences between the two trajectories of different lengths. This measurement is also robust to noise by giving the space and time thresholds to find similar points of two trajectories. Meanwhile, DTW in [37] has been successfully applied to time series data and trajectories. DTW is based on defining a cost for aligning two data points, and it finds the minimum cost to align all points between two trajectories. However, this measurement is sensitive to noises because the aligning cost is based on a real distance between two points [54]. EDR in [38] can remove noise effects by quantizing the distance between a pair of elements to two values, 0 and 1. This measurement does not require the same lengths of two considering trajectories by seeking the minimum number of edit operations to change one trajectory to another. The authors in [55] proposed an indoor semantic trajectory similarity measure (ISTSM) that was improved from the EDR. ISTSM integrated semantic labels and indoor walking distance to calculate the similarity of indoor semantic trajectories. On the other hand, the indoor walking distance was only considered if the two points were the same semantic. They directly assigned the penalty value of two different semantic points to a maximum value of 1. In other words, they ignored the space aspect between two points if they were different semantic labels.

Existing metrics do not efficiently discover the characteristics of indoor trajectories for determining the trajectories' distance. Thus, in the thesis, we propose a new distance metric for indoor trajectories, which uses both the indoor walking distance and semantic information for calculating distance of trajectories.

2.2 Distance-based Anomalous Trajectory Detection

The distance-based anomalous trajectory detection method calculates the similarity between trajectories using a distance function. A similarity threshold is given to identify abnormal trajectories [56, 57]. Zhu et al. introduced the time-dependent popular routesbased trajectory outlier detection (TPRO) method [56]. In this study, the most popular routes on each timestamp were used to detect the temporal anomalies. The trajectory datasets were divided into groups using a partitioning strategy. A reference trajectory represents each group. The edit distance between the reference trajectory of each group and the popular routes in the given city was then calculated. The anomalous trajectory groups were detected if the distance between its reference trajectory and the popular routes was larger than a given distance threshold. Saleem et al. presented the road segment partitioning towards anomalous trajectory detection (RPAT) algorithm [57]. The trajectories were divided into sub-trajectories based on the road segments. The speed, flow rate, and visited time were features used to calculate the score for each sub-trajectory. The score of each trajectory is the total of its sub-trajectory scores. Trajectory scores above a user-specified threshold are anomalies.

2.3 Extensible Markov Model-based Anomalous Trajectory Detection

The extensible Markov Model (EMM) combines an adaptive Markov chain and a clustering algorithm to detect anomalous trajectories [33]. The threshold nearest neighbour clustering algorithm (tNN) is used in this study. In particular, each cluster is modelled by a reference point. With a new data point, its distances and existing clusters are determined. If the smallest distance is greater than a given clustering threshold, a new cluster is created. In contrast, the point belongs to the cluster according to the smallest distance.

In the EMM model, each node is a cluster of location points, which is represented by a cluster model. Depending on the distance between a new point and clusters, the point is grouped either into one of the existing nodes or forms a new node. In detecting anomalies, a point is marked as an anomaly if it belongs to one of two situations: The point forms a new node, or the point belongs to a node whose occurrence probability or whose transition probability is lower than a given threshold. A trajectory is considered anomalous if it contains at least one abnormal point.

2.4 Density-based Anomalous Trajectory Detection

In the density-based anomalous trajectory detection method, the neighbor density of trajectories is estimated to detect anomalies. A previous study [27] proposed the trajectory outlier detection (TRAOD) algorithm by investigating the partition and detection strategy in finding sub-trajectory outliers. A new distance function was proposed, which comprises three components: Perpendicular distance, parallel distance, and angle distance. The density of sub-trajectories was determined using a distance threshold. A sub-trajectory is anomalous if its density is smaller than a threshold. The study in [31] also introduced a distance measure that uses intra-trajectory and inter-trajectory features. The distances between trajectories were first calculated to find the neighbor density of each trajectory. The anomalous trajectories were then detected based on the density threshold. In these studies, a mechanism for determining density threshold is not provided. Thus, the performance of anomaly detection can be affected.

By contrast, in our density-based anomalous trajectory detection framework, a new method is proposed to choose distance and density thresholds. The thresholds are determined based on the distribution of trajectories in datasets.

2.5 Clustering-based Anomalous Trajectory Detection

Clustering-based methods group trajectories into clusters using an appropriate clustering algorithm. Anomalies are detected if they do not belong to any clusters or belong to clusters that only have a few trajectories. Wang et al. [2] proposed an anomalous trajectory detection method using a hierarchical clustering algorithm to derive anomalous trajectories from a taxi GPS dataset. First, the trajectories that travel between the same source and destination were all extracted. The hierarchical clustering algorithm is then adopted to derive the clusters of trajectories using the edit distance. The clusters that have only a trajectory are finally marked as anomalies. Unlike in [2], the authors in [30] developed a two-phase anomalous trajectory detection framework: Online phase and offline phase. The offline phase finds the clusters of trajectories. In this step, the distance between trajectories was calculated using LCSS, and a hierarchical clustering algorithm was also adopted to group trajectories. In the second phase, a trajectory was marked as an anomaly if it did not belong to any clusters.

The challenge of clustering-based anomaly detection methods is determining the input parameters of an algorithm. For example, the hierarchical clustering algorithm requires the number of clusters as an input parameter. Previous studies [58, 59] proposed a few indices to find the number of clusters. These indices estimated the compactness of clusters and the separation between clusters for finding the number of clusters in datasets. DBSCAN requires two input parameters: Eps and MinPts; determining the Eps parameter is more difficult for DBSCAN. Many methods have been proposed to choose the Eps value. A combination of the elbow method and the kth nearest neighbour is used to determine the Eps value for DBSCAN [60]. They found k nearest neighbours of all data points in the dataset and sorted them in descending order of $k - \text{distance}$. An Eps value was chosen according to the cutoff point of the sorted $k - \text{dist}$ graph. On the other hand, the cutoff point cannot always be identified. A new method to choose the Eps value was proposed to overcome the disadvantage of the elbow method [61]. They automatically found the greatest slope change instead of observing the graph. The Eps value was determined according to the point with the greatest slope. The performance of this method still depended on the shape of the $k - \text{dist}$ graph. A new approach was proposed to determine the Eps value using empty circles [62]. They started by finding all empty circles in the dataset. The radius of the circles was sorted in descending order. The elbow value of this sorted radius was chosen as the Eps value. Nevertheless, similar to the reported method [60], it was difficult to determine the appropriate elbow value if it was unclear.

By contrast, in this work, the value of Eps is selected based on an evaluation of the clustering quality. Specifically, a novel DBSCAN clustering validity index called DCVI is proposed to choose the Eps value. DCVI measures the compactness of clusters, the separation between clusters, and the separation between clusters and outliers. The value of Eps is determined according to the maximum value of DCVI.

2.6 Deep Learning-based Anomalous Trajectory Detection

With recent neural network development, anomalous trajectory detection methods have started to focus on learning trajectory representations.

In particular, RNNs, which are used for learning sequential data, are widely applied for outlier detection tasks in trajectory data. The authors in [39] proposed a deep learning model called Anomalous Trajectory Detection using Recurrent Neural Network (ATD-RNN) to identify abnormal trajectories. This model learned the trajectory embedding, that kept normal trajectories' internal characteristics and their sequential information. ATD-RNN was trained by minimizing the cross-entropy loss function using a labeled dataset. With a new trajectory, the model predicted the probability of detecting the trajectory as an anomaly.

The study in [63] introduced an LSTM autoencoder-based Seq2Seq model to identify abnormal vehicle routes. An LSTM encoder mapped each input route into a vector, which captured the input route's characteristics. Then an LSTM decoder reconstructed the encoded route. An anomaly was detected if it was not reconstructed correctly by the autoencoder. Similarly, a framework was proposed for detecting dangerous driving behavior and hazardous roads using autoencoders [40]. In particular, the autoencoders were trained to learn a latent space that captures the input sequences' most representative characteristics. The difference between the input and reconstructed sequences also was used to indicate outliers. In the work, the authors studied autoencoders based on CNN and LSTM architectures for anomaly detection tasks.

In recent years, due to the occurrence of the Transformer architecture with an attention mechanism, capturing long-range dependence in sequential data is performed more effectively. Anomaly detection methods based on this neural network have also emerged gradually. For example, detecting anomalies in electrocardiogram (ECG) signals based on the Transformer was introduced in [64]. In this paper, the authors used the Transformer encoder to learn normal data patterns. Anomalies are detected using errors between the predicted and original ECG signals. The study in [41] also proposed an anomalous detection method based on the Transformer for improving safety and predicting risks in traffic. Their model used the Universal Transformer encoder to learn trajectories' embeddings by keeping information on trajectory points. Like the study [39], this model was also trained using the cross-entropy loss function, and a labeled dataset for abnormal and normal trajectories was required. Both above Transformer-based models learned to capture the interior features of normal samples in input data. However, they do not learn the relationship between representations of data in latent space to detect anomalies. Besides, the study in [64] does not provide an effective mechanism for choosing an appropriate anomaly threshold. This work still needs to set a specific value in the formulation of determining the anomaly threshold.

By contrast, in our proposed deep learning model, learning trajectory representations and the relationship between trajectory representations in the latent space is performed in a deep learning neural network. In particular, a model based on the Transformer encoder and SOM is proposed. The Transformer encoder with the self-attention mechanism learns the internal characteristics of trajectories. To learn the relationship between the trajectory representations in the latent space, a SOM layer is used in our model. Specifically, the SOM learns clusters of trajectory representations in the latent space. In addition, in detecting anomalies, an effective mechanism is provided for determining an anomaly threshold. The anomaly threshold is first chosen based on trajectories' anomaly scores in the training set. Then, a new metric (i.e., WS) is proposed to select the appropriate value of the anomaly threshold.

Chapter 3

Generating Anomaly Types in Indoor Trajectories

In this section, datasets for evaluating anomaly detection frameworks in this thesis are first introduced (i.e., MIT Badge and sCREEN datasets). Then, anomalous trajectory types are studies, and they are generated for evaluation.

3.1 Datasets

3.1.1 MIT Badge Dataset

The MIT Badge dataset includes the timestamped geographic locations of employees at an IT call center in Chicago from 26 March–17 April 2007 [52]. The location data of workers is estimated by the radio signal strength (RSSI) of the badge assigned to each worker. The in-house positioning system measures the RSSIs of each employee's badge at various base stations placed around the office. These signals are used to determine the instantaneous position of each badge. Thirty-six workers from three other groups participated in the data collection. The configuration group contains 28 workers. On the other hand, there are only seven in the pricing group and four in the coordinator group. Each recorded data point contains x and y coordinates at each timestamp. The sampling rate of data is 10 points per minute.

Each trajectory contains data points collected in a time window W. The data are collected within 17 days and divided into two parts for evaluation. The first part accounts for 70% of the total and contains 12 days. This part is used in the first phase of the framework to find the workers' normal trajectory clusters. The second part is the test dataset, with 30%

of the total that contains data within five days. The days for testing are chosen randomly from 17 collected days in the dataset.

3.1.2 sCREEN dataset

The sCREEN dataset was gathered in a German supermarket during business hours in July 2016 [20]. The location data of customers is collected using sensors installed in shopping carts and baskets. While customers shop for items, sensors send the ultrawideband signal regularly to anchors placed on the supermarket's ceiling. The location of the anchors is known. The real-time location system is based on the position of three anchors with the same timestamp to estimate customers' location. Similar to the MIT Badge dataset, each data point in the sCREEN dataset also contains information about the coordinates and timestamps of each point. Each trajectory is extracted from the location trace using the time window W. Assuming that the working hours in the supermarket are from 8:00 to 22:00. There are 175 carts and baskets to collect data for 29 days.

3.2 Generating Anomalous Trajectory Types

In this section, two methods for generating anomalies for evaluation are presented.

3.2.1 Hypothesized Anomalies

In the first method, a hypothesis is given for anomalies based on the difference in behavior between groups in the dataset. The hypothesis is that if one group occurs more frequently than the other, this group is described as a normal group. The remaining group with less occurrence is described as an anomalous group. In particular, in the MIT Badge dataset, the configuration group accounts for approximately 70% of the total (i.e., 28 workers), while the pricing group accounts for only 20% (i.e., 7 workers). Since the movement patterns of two of these groups are different, it is assumed that the movement of employees in the configuration group is normal and in the pricing group is abnormal. The sCREEN dataset does not contain different groups, so there is no hypothesis given for anomalies in this dataset. In other words, the first way to generate anomalies is not used in the sCREEN dataset.

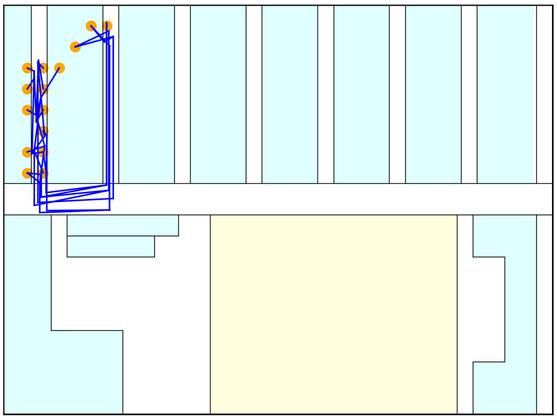


Figure 3.1: A normal trajectory

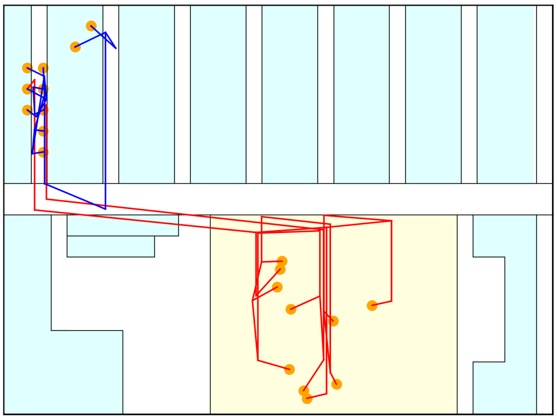


Figure 3.2: A rare location visiting anomaly

3.2.2 Synthesized Anomalies

In the second method, anomalies are generated anomalies and injected into datasets. In this thesis, three anomaly types are studied: rare location visiting, wandering and route anomalies.

- **Rare Location Visiting Anomaly.** In indoor spaces, the rare location refers to where humans have rarely visited or been prohibited from visiting. For example, in a factory, the prohibited places may be security control and engine rooms that workers can not enter. Moreover, the rare locations can also be places that workers visit only at a specific time. For example, the cafeteria can be a rare location with workers during working hours, even though they may come there for lunch. Similarly, customers in supermarkets or stores are also not permitted to access some locations, including security, staff areas, and warehouses. Therefore, if one person moves to rare locations, his/her behavior may be anomalous. Rare locations need to be identified based on the floor plan and historical trajectories of the dataset to generate this anomaly. First, the floor plan is divided into grid cells. Then, the probability of each trajectory visiting cells is calculated using the history trajectories. The cells with visiting probability are 0 chosen as rare locations. Rare location visit anomalies are generated from the normal trajectories by shifting some points of original trajectories to the rare locations. The number of shifted points is controlled by the parameter τ. For example, $\tau = 0.5$ means 50% of the normal trajectory is moved to rare locations. The starting point for shifting is randomly chosen from the original trajectory. As shown in Figures 3.1 and 3.2, examples of a normal trajectory and a rare location visiting anomaly, respectively. In these figures, the yellow area represents a rare location. The blue path is the normal part, and the red path represents the anomaly part.

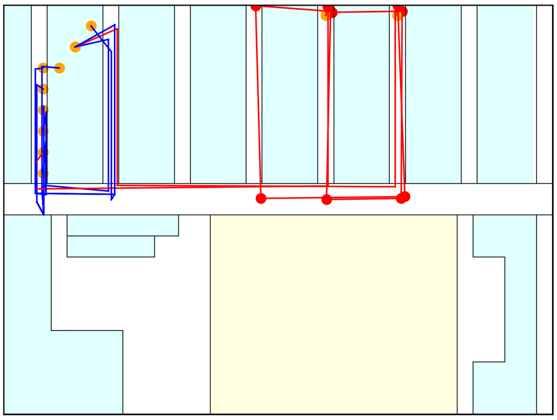


Figure 3.3: A wandering anomaly

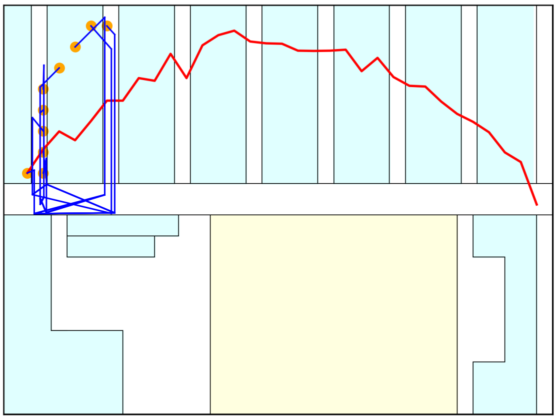


Figure 3.4: A route anomaly

- **Wandering Anomaly.**

A wandering anomaly can be created when a person wanders around objects many times, possibly with bad intentions. To create this anomaly, wandering routes in floor plans in datasets are found. Then, a wandering anomalies are synthesized from original trajectories based on found wandering routes. An example of wandering anomaly is depicted in Figure 3.3, which wanders around the working rooms (i.e., the red path). Parameter τ also is used to control the abnormality level.

- **Route Anomaly.** A route anomaly occurs when people take unusual routes to escape in urgent situations (e.g., fire and attacking violence). For example, when a fire occurs suddenly in indoor spaces, humans tend to move following random routes to escape. In this case, people tend to run to exits of the buildings. Thus, we choose the exit as the end point of the route anomaly. An example of a route anomaly is generated as Figure 4d, with the red path being abnormal. Like the two above anomaly types, the abnormality level is controlled by parameter τ.

3.3 Chapter Summary

In this chapter, we first introduced two datasets (i.e., MIT Badge and sCREEN datasets), which are used for the evaluation of anomaly detection methods in this work. Then, two methods for generating anomalies in datasets are presented in more detail. In the first method, a hypothesis is given to assign labels to trajectories in datasets. In the second method, anomalies are synthesized and injected into datasets for evaluation.

Chapter 4

Density-based Anomalous Human Trajectory Detection

4.1 Proposed Anomalous Trajectory Detection Framework based on Density

In this section, a density-based anomalous trajectory detection framework is proposed. Then, the proposed framework is evaluated and compared with other baselines using the MIT Badge and sCREEN datasets.

4.1.1 Definitions

First, three obvious definitions related to trajectory are provided. Then, EDR is introduced in detail that is used in this framework.

4.1.1.1 Definitions Related to Trajectory

- Definition 1. A trajectory T is a sequence of multi-dimensional points and denoted by $T = p_1, p_2, \ldots p_n$. In our work, p_i is a location point in 2-dimensional space with latitude $p_{i,x}$ and longitude $p_{i,y}$. n is the length of trajectory T and can differ from the lengths of other trajectories.

- Definition 2. Two trajectories S and R are neighbors to each other if $Dist_{(S,R)} \leq Thres_{Dist}$. Here $Dist_{(S,R)}$ is the distance between S and R. $Thres_{Dist}$ denotes the distance threshold used to determine whether two trajectories are neighbors or not.

- Definition 3. A trajectory S is an anomaly if and only if $Den_S \leq Thres_{Den}$, where Den_S is the neighbor number of S and is a given density threshold.

4.1.1.2 Definition of EDR

$$EDR(R,S) = \begin{cases} n, \text{if } m = 0 \\ m, \text{if } n = 0 \\ min\{EDR(Rest(R), Rest(S)) + Cost[replace(s_n, r_m)], \\ EDR(R, Rest(S)) + Cost[delete(s_n)], \\ EDR(S, Rest(R)) + Cost[insert(r_m)]\}, \text{otherwise} \end{cases} \tag{4.1}$$

where $Cost[delete(s_n)] = Cost[insert(r_m)] = 1$. $Cost[replace(s_n, r_m)] = 0$ if and only if $|s_{n,x} - r_{m,x} \leq \eta$ and $|s_{n,y} - r_{m,y} \leq \eta$ else $Cost[replace(s_n, r_m)] = 1$. η is a predefined matching threshold. The cost of insert or delete operations is always 1. Meanwhile, the cost of replace operation between s_n and r_m is 0 if two points matches. In contrast, the cost of replace operation is 1.

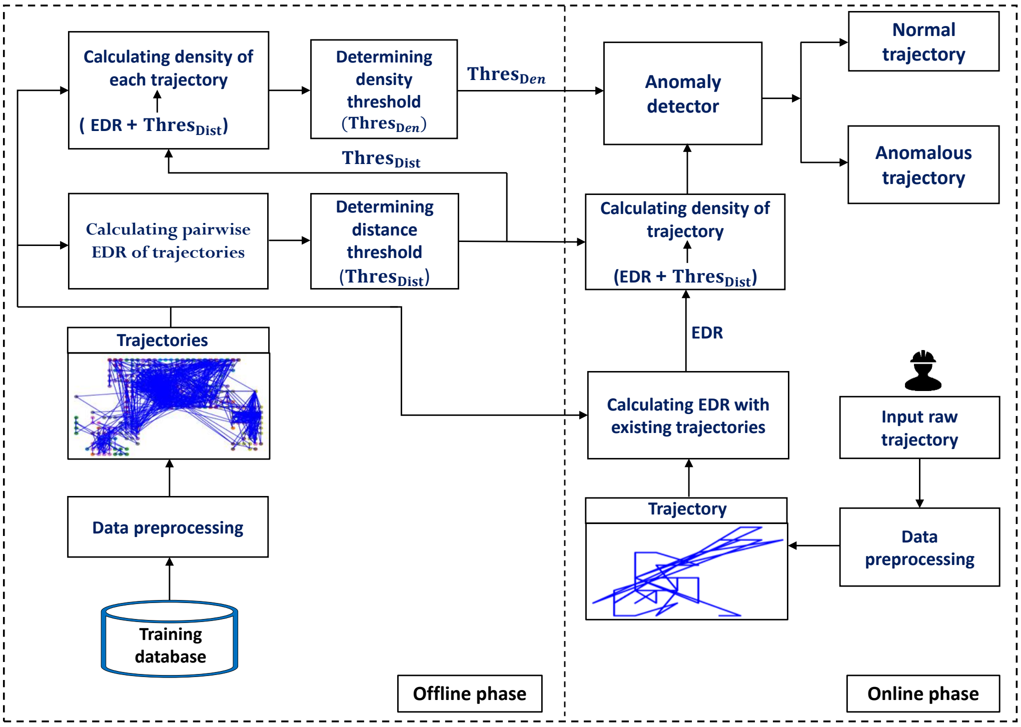


Figure 4.1: Anomalous Trajectory Detection Framework

4.1.2 Methodology

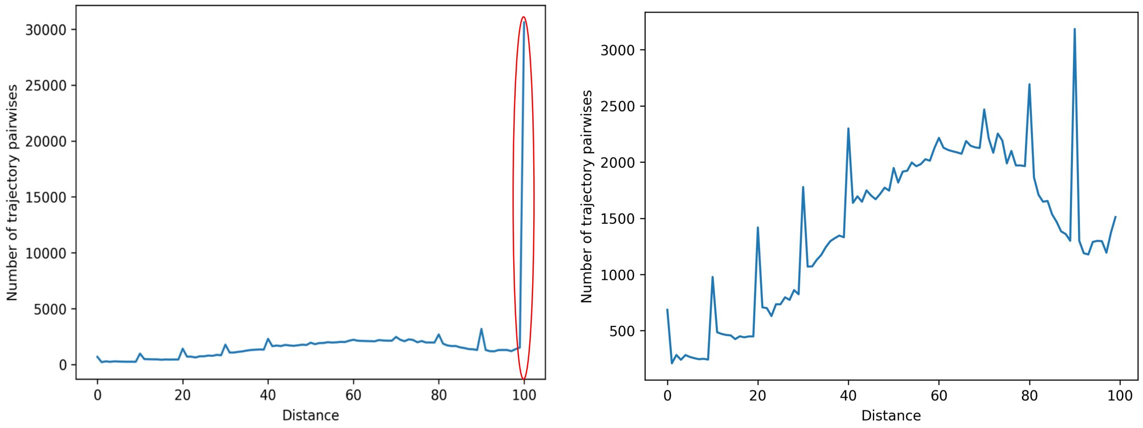
First, an anomalous trajectory detection framework is introduced. Then, a new method is proposed to determine distance and density thresholds.

4.1.2.1 Anomalous Trajectory Detection Framework

In this work, we want to design an abnormal detection approach. The abnormal trajectory detection framework is divided into two phases: offline and online as shown in Figure 4.1.

In during the offline phase, the parameters $Thres_{Dist}$ and $Thres_{Den}$ are determined based on collected trajectories in dataset. The dataset contains both normal and abnormal unlabeled trajectories. Before proceeding, these raw trajectories must be preprocessed. Since the objective is to detect the workers' abnormal behaviors as soon as possible, their trajectories are checked within short periods of time, which is known as timeslot. Hence, during the data preprocessing step, each trajectory is collected using timeslot.

In the online phase, each input trajectory is checked. First, we use EDR to calculate the distances between the input trajectory and all existing trajectories in the dataset. Then, the neighbor density of this trajectory is determined based on $Thres_{Dist}$. Finally, anomaly detector shows that the input trajectory is abnormal if its density is not higher than $Thres_{Den}$.



(a)

(b)

Figure 4.2: Distribution of pairwise distances: (a) Before preprocessing, (b) After preprocessing.

4.1.2.2 Determining Distance Threshold

The distance threshold $Thres_{Dist}$ is used to determine whether two trajectories are neighbor or not. Two trajectories are neighbors if their distance is no larger than $Thres_{Dist}$. This parameter has a direct impact on the algorithm performance. If $Thres_{Dist}$ is small, many trajectories are detected as anomalies, and the false alarm rate is high. In contrast, if

$Thres_{Dist}$ is large, there are very few anomalous trajectories found. Hence, choosing an appropriate value of $Thres_{Dist}$ is a crucial task in the density-based anomaly detection method.

From the dataset, the pairwise distances between trajectories are calculated. Then is chosen based on the mean value μ_{Dist} and the standard deviation σ_{Dist} of the pairwise distances. Since the distance threshold is used to determine whether two trajectories are neighbors or not, only trajectories related to each other are considered. From this angle, the distance value that does not represent the relationship between trajectories is removed before calculating $Thres_{Dist}$. As shown in Figure 4.2(a), there is a peak at the value of 100. This is the maximum EDR value of two trajectories in the dataset, and it represents those two completely different trajectories. In the preprocessing step, we remove this peak, and receive the result as shown in Figure 4.2(b). Before choosing the distance threshold, the trajectories' pairwise distances at all timeslots are collected. Therefore, all timeslots have the same distance threshold $Thres_{Dist}$. After calculating the mean value and the standard deviation of pairwise distances, we obtain μ_{Dist} and σ_{Dist}. The distance threshold is defined as follows:

$$Thres_{Dist} = \mu_{Dist} + \varphi_1 \times \sigma_{Dist}, \tag{4.2}$$

where φ_1 is a predefined parameter by user.

4.1.2.3 Determining Distance Threshold

The density threshold is used to determine whether a trajectory is detected as normal or abnormal. If density of a trajectory is no larger than $Thres_{Den}$, it is marked as an anomaly. The density threshold must also be chosen before using to detect anomaly. In MIT Badge dataset, the neighbor density of workers depends on the working hour. For example, the number of workers in during the first and last hours of a day often is less than during other hours. If we only use a $Thres_{Den}$ for all hours of the day, the method's performance may be affected. From this viewpoint, the density threshold is determined in accordance with timeslots. This means that each timeslot will receive an appropriate density threshold. The density threshold at timeslot T is denoted as $Thres^T_{Den}$. To determine $Thres^T_{Den}$, we calculate the density of each trajectory in the dataset at timeslot T. The value of $Thres^T_{Den}$ is also selected based on the mean value μ^T_{Den} and the standard deviation value σ^T_{Den} of trajectories' densities. $Thres_{Den}^{T}$ is defined as follows:

$$Thres_{Den}^{T} = \mu_{Den}^{T} + \varphi_2 \times \sigma_{Den}^{T}, \tag{4.3}$$

where φ_2 is a predefined parameter by user. Since the anomalous trajectory has little in common with the other trajectories, the density threshold $Thres_{Den}^{T}$ for detecting anomaly is smaller than the mean value μ_{Den}^{T} of trajectories' densities ($\varphi_2 < 0$).

4.1.3 Performance Evaluation

4.1.3.1 Experiment Setup

In this subsection, the proposed framework is evaluated using the MIT Badge and sCREEN datasets. In the MIT Badge dataset, the chosen timeslot is 10 minutes. This means that, in each timeslot, the maximum length of a trajectory is 100. Because data may be missing, this value may be less than 100. Each day is divided into 54 timeslots from 9:00 to 18:00. The experiments are conducted three times with different sets of test days. The output of the algorithm is the average of all run times. The average number of trajectories over timeslots is about 4000 in the first phase. The average number of trajectories for testing is about 400, with one half for the normal trajectories and the other half for abnormal trajectories.

In the sCREEN dataset, areas visited by customers in supermarkets do not depend on the time of day, so time is not divided into slots in the sCREEN dataset. All trajectories are grouped to process for detecting anomalies. The chosen timeslot is 5 minutes in this dataset.

4.1.3.2 Results

There are three parameters determined for the proposed framework: η in the EDR measurement, $Thres_{Dist}$ and $Thres_{Den}$. η is a threshold for determining whether two points of two trajectories are matched or not. If they are matched, the cost of replacement is 0; otherwise, it is 1. Based on the floor plan of the MIT Badge dataset, we recognize that two workers are close to each other if the Euclidean distance between them is less than 10 meters. The parameter η is then set to 1000 according to the scale in the floor plan of MIT Badge dataset. With the sCREEN dataset, the value of η is chosen at 4 (i.e, 4 meters).

In order to determine $Thres_{Dist}$ and $Thres_{Den}$, we must first define the values of φ_1 and φ_2, respectively. The precision and recall of algorithm depend on the values of φ_1 and φ_2.

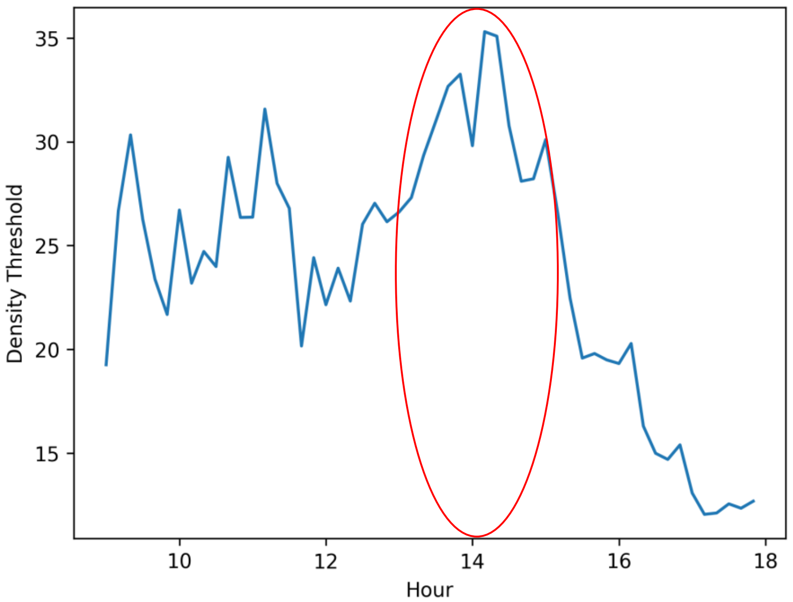


Figure 4.3: Density threshold distribution over time in the MIT Badge dataset

Table 4.1: Results on the MIT Badge dataset.

	EMM			Density		
	Recall	Precision	f1-score	Recall	Precision	f1-score
Pricing group	0.8955	0.6687	0.7635	0.9929	0.6394	0.7779
Rare location visiting anomaly	1	0.6082	0.7563	0.9964	0.6826	0.8098
Wandering anomaly	1	0.6082	0.7563	1	0.6834	0.8115
Route anomaly	1	0.6082	0.7563	1	0.6834	0.8115

To ensure the anomaly detection ability as well as the precision of the algorithm, we choose: $\varphi_1 = -0.4$ and $\varphi_2 = -0.6$ in this work.

In the MIT Badge dataset, the density threshold $Thres_{Den}^{T}$ is determined for each timeslot T, and distribution of the density threshold over all timeslots is shown in Figure 4.3. As can be seen, the density threshold is low during the first hours and quickly reduces during lunch and last hours. Meanwhile, this value is the highest from 1.00 pm to 3.00 pm in the range of red ellipse.

Table 4.2: Results on the sCREEN dataset.

	EMM			Density		
	Recall	Precision	f1-score	Recall	Precision	f1-score
Rare location visiting anomaly	1	0.5828	0.7364	1	0.7225	0.8389
Wandering anomaly	1	0.5828	0.7364	1	0.7225	0.8389
Route anomaly	1	0.5828	0.7364	1	0.7225	0.8389

The results of detecting anomaly types are shown in Tables 4.1 and 4.2. The proposed framework is compared with EMM model-based anomalous trajectory detection framework.

As can be seen in Tables 4.1 and 4.2, the density-based framework outperforms the EMM-based framework over all anomalies. With hypothesized anomalies in the MIT Badge dataset, the proposed framework achieves at 77.79% and is 1.26% higher than the EMM-based framework in terms of f1-score.

The density-based and EMM-based frameworks detect correctly synthesized anomalies with recall of 1 in both datasets. In addition, the EMM-based framework only achieves about 76% and 74% of f1-score in the MIT Badge and the sCREEN datasets, respectively. In contrast, f1-score of the proposed framework is about 81% and 84% on the MIT Badge and the sCREEN datasets, respectively.

4.2 Chapter Summary

In this chapter, we introduced first two datasets: MIT Badge and sCREEN, which are used to evaluate the proposed frameworks in this thesis. In addition, anomalous trajectory types are synthesized and injected to the datasets for evaluation.

Next, we proposed an anomalous human trajectory detection framework based on the density method. The framework is divided into two phases: offline and online. Task of the first phase is to determine the values for parameters: distance threshold and density threshold. Meanwhile, in the second phase, a new trajectory is checked that it is an anomaly or not. We also validate the algorithm using MIT Badge and sCREEN datasets, and receive the better results the EMM-based framework. In the next chapter, we plan to improve the distance function by combining the semantic information and the space information for determining the distance of two trajectories. In addition, anomalous trajectory detection is performed based on a clustering algorithm.

Chapter 5

Clustering-based Anomaly Detection in Indoor Human Trajectories

In this chapter, we proposes a two-phase framework for detecting indoor human trajectory anomalies based on density-based spatial clustering of applications with noise (DBSCAN). The first phase of the framework groups datasets into clusters. In the second phase, the abnormality of a new trajectory is checked. A new metric called the longest common subsequence using the indoor walking distance and a semantic label (LCSS_IS) is proposed to calculate the similarity between trajectories, extending from LCSS. Moreover, a DBSCAN cluster validity index (DCVI) is proposed to improve the trajectory clustering performance. The DCVI is used to choose the epsilon parameter for DBSCAN. The proposed method is evaluated using two real trajectory datasets: MIT Badge and sCREEN.

5.1 Definitions

In this section, definitions of LCSS measure and DBSCAN are provided.

5.1.1 Longest Common Subsequence (LCSS) Measure

LCSS is a similarity measure for character strings [65–68]. This measure tries to identify the longest common subsequence between two considering strings. In another study [36], the original LCSS was extended to compare two trajectories of moving objects. Let R and S be two trajectories with length m and n, respectively, where $R = ((r_{x,1}, r_{y,1}), (r_{x,2}, r_{y,2}), ..., (r_{x,m}, r_{y,m}))$ and $S = ((s_{x,1}, s_{y,1}), (s_{x,2}, s_{y,2}), ..., (s_{x,n}, s_{y,n}))$. For a trajectory R, let $Rest(R)$ be the trajectory $Rest(R) = ((r_{x,1}, r_{y,1}), (r_{x,2}, r_{y,2}), ..., (r_{x,m-1}, r_{y,m-1}))$. LCSS of R and S trajectories is defined in the following equation [30].

$$LCSS(R,S) = \begin{cases} 0, \text{if } m = 0 \text{ or } n = 0 \\ LCSS(Rest(R), Rest(S)) + 1, \\ \text{if } |r_{x,m} - s_{x,n}| < \eta \text{ and } |r_{y,m} - s_{y,n}| < \eta \text{ and } \mid m - n \mid \leq \nu \\ max\{LCSS(Rest(R), S), LCSS(R, Rest(S))\}, \text{otherwise} \end{cases} \tag{5.1}$$

where η and ν are pre-defined parameters that depend on the application and the dataset. η is a distance threshold to determine whether two points are matched. If the difference in the X dimension ($|r_{x,m} - s_{x,n}|$) and the difference in the Y dimension ($|r_{y,m} - s_{y,n}|$) between two points r_m and s_n are smaller than η, the two points are matched. ν controls how far in time to match one point of R to a point of S.

LCSS can calculate the similarity between two trajectories of varying lengths by finding the matched points between two trajectories. This measurement is also robust to noise by giving the distance and time thresholds to find close points of two trajectories. However, in the LCSS measure, the proximity of two points from two trajectories is determined using the differences on X and Y dimensions. This measure ignores the movement constraints of indoor spaces, which are limited by entities such as rooms, corridors, and stairs. The actual travel distance between two points in indoor space is longer than the distance used in LCSS. LCSS is inappropriate for determining the similarity of indoor trajectories. Moreover, LCSS only uses the space distance to determine the trajectories' similarity and ignores the semantic information of points. Therefore, this paper proposes a new similarity measure called LCSS_IS, which uses indoor walking distance and semantic labels to determine the similarity of trajectories.

5.1.2 DBSCAN

DBSCAN aims to seek high-density clusters and detect outliers in datasets [60, 69–71]. The clustering algorithm can also handle datasets with noise and clusters of any shape. In addition, DBSCAN does not require the number of clusters as an input parameter, while some of the clustering algorithms require it [72]. Nevertheless, this algorithm requires two input parameters: Eps and MinPts. The former is the radius to find neighbors, and the latter is the minimum number of points to form a cluster.

Several related definitions of DBSCAN are first stated.

- Definition 1: Point q is an $Eps - neighbor$ of a point p if $dist(p, q) \leq Eps$. $dist(p, q)$ is the distance between two points p and q.
- Definition 2: Point q is a core point of the cluster if the number of its $Eps - neighbors$ is greater than or equal to $MinPts - 1$.
- Definition 3: Point q is a border point of the cluster if it is not a core point, but it is an $Eps - neighbor$ of the core point.
- Definition 4: Outliers are points that are neither core points nor border points of the clusters.

DBSCAN labels the data points in the dataset as core points if they satisfy Definition 2. Core points are $Eps - neighbor$ to each other belonging to the same cluster. From obtained core points, border points are labeled if they satisfy Definition 3. Outliers are data points that do not belong to any clusters [62].

5.2 Methodology

In this section, a two-phase framework is first proposed to detect human trajectory anomalies using DBSCAN. To improve the accuracy of the similarity measure between trajectories, a new measure called LCSS_IS is then proposed for indoor human trajectories. In LCSS_IS, the indoor walking distance and semantic information are combined to evaluate the similarity between trajectories rather than relying solely on coordinates' differences between points as in the original LCSS. Unlike ISTSM, which only uses the indoor walking distance for points that have the same semantic label, LCSS_IS uses it in both cases of the same and different semantic labels. Finally, to improve the clustering performance of DBSCAN, a cluster validity index is proposed to determine the Eps value, an important parameter of DBSCAN.

5.2.1 Clustering-Based Anomalous Trajectory Detection Framework

This section introduces an overview of the two-phase framework for detecting human trajectory anomalies. This section also discusses how to cluster trajectories using DBSCAN and detect anomalous trajectories.

As shown in Figure 5.1, phase 1 of the framework groups the trajectories in the database into clusters. First, raw location traces from the database are preprocessed to extract a

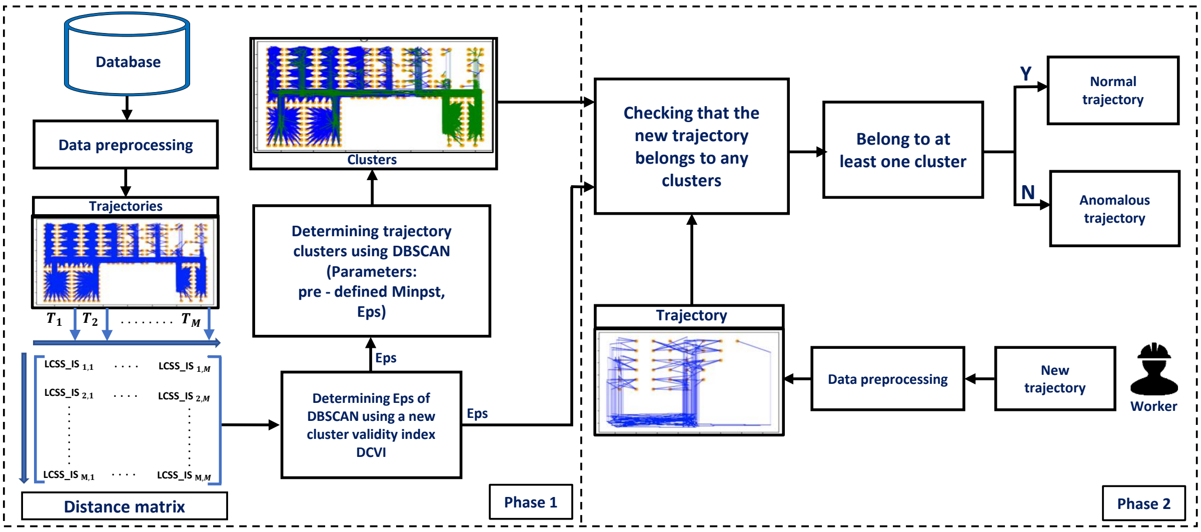

Database
Data preprocessing
Trajectories
T_1 T_2 .... T_M
LCSS_IS 1,1 .... LCSS_IS 1,M
LCSS_IS 2,1 .... LCSS_IS 2,M
LCSS_IS M,1 .... LCSS_IS M,M
Distance matrix
Clusters
Determining trajectory clusters using DBSCAN (Parameters: pre - defined Minpst, Eps)
Eps
Determining Eps of DBSCAN using a new cluster validity index DCVI
Eps
Phase 1
Checking that the new trajectory belongs to any clusters
Belong to at least one cluster
Y
N
Normal trajectory
Anomalous trajectory
Trajectory
Data preprocessing
New trajectory
Worker
Phase 2

Figure 5.1: A two-phase framework for detecting human anomalous trajectory.

set of trajectories. Trajectories that have lost many data points from equipment recording errors are removed. Then, a distance matrix of processed trajectories is calculated using a new distance function LCSS_IS that is extended based on LCSS. Finally, we cluster the trajectories using DBSCAN. This step proposes a new cluster validity index DCVI to choose an appropriate Eps value for DBSCAN.

Anomalous trajectory detection is performed in phase 2. A new trajectory is checked to determine whether it belongs to any clusters obtained from the database in phase 1. The Eps-neighbors of the new trajectory in clusters are then determined using the Eps value. The new trajectory belongs to the cluster if the number of neighbors is greater than or equal to $(MinPts - 1)$. The new trajectory is detected as an anomaly if it does not belong to any clusters. Otherwise, it is a normal trajectory.

5.2.2 Similarity Measure for Indoor Human Trajectories

5.2.2.1 Longest Common Subsequence using Indoor Walking Distance and a Semantic Label (LCSS_IS)

To determine LCSS_IS between two trajectories, a navigation graph is first constructed to calculate the indoor walking distance between two points. The navigation graph represents the topology of a floor plan of an indoor space [73–76]. In this work, the navigation graph is constructed using a connectivity base graph model [77]. The connectivity base graph is defined by vertices and edges. Entities of the floor plan, such as rooms, stairs, and hallways, are decomposed and represented by vertices. Edges are used to connect vertices in the graph. An edge corresponds to a connection between two partitions in the floor plan. Moreover, to represent the proximity between entities in indoor space, each edge is assigned a value according to the distance of entities. The connectivity base graph model has a low computation complexity while maintaining indoor trajectory modeling efficiency.

In the present work, depending on the floor plans of indoor spaces in datasets, each vertex represents a particular space. For example, in the MIT Badge dataset, a navigation graph is constructed as shown in Figure 5.2. In this floor plan, the space is divided into small working cubicles for workers. Therefore, each vertex is considered to cover two opposite cubicles and the part of the corridor between these cubicles. Moreover, since the movement characteristics are important for anomaly detection in human movement, vertices are positioned along the corridors. In Figure 5.2a, vertex 10 covers a space, represented by a small rectangle in the upper left corner of the floor plan. The graph vertices are connected based on the walking routes in the indoor area. To measure the indoor walking distance between the two positions in the floor plan, two vertices in the navigation graph that are the closest to the two positions in the floor plan are determined. The shortest path between the two vertices is defined as the indoor walking distance using the Dijkstra shortest path algorithm [78]. For example, in Figure 5.2, vertices 12 and 18 are the closest to the A and B points, respectively. Therefore, the distance between points A and B is the shortest path from vertex 12 to 18. This path is marked with red arrows as shown in Figure 5.3.

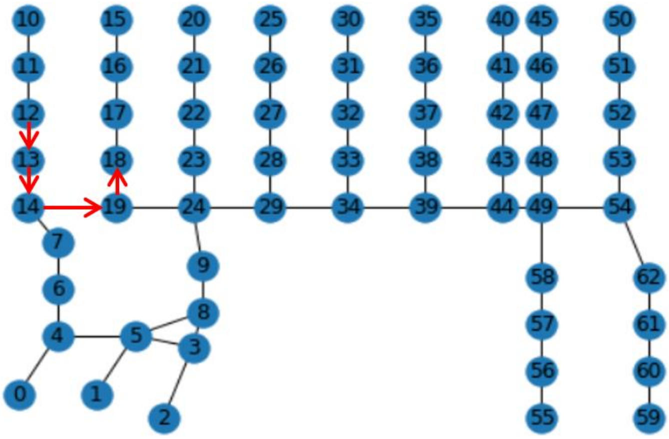


Figure 5.2: Indoor navigation graph.

Next, semantic labels need to be assigned to the trajectory points to compute the distance between indoor trajectories using LCSS_IS. Each point is matched with a label using the

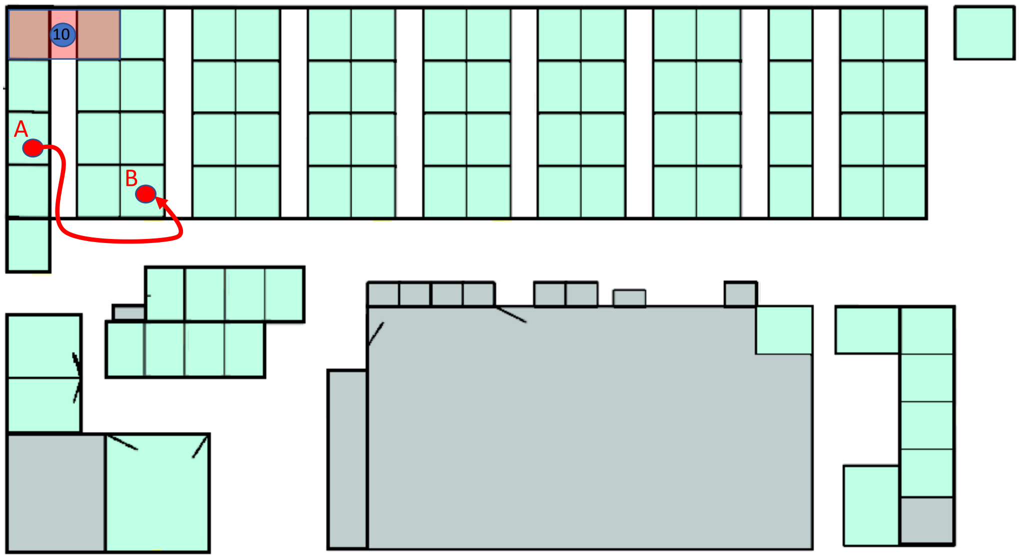


Figure 5.3: Floor plan.

Table 5.1: List of semantic labels in datasets.

Dataset	MIT Badge			sCREEN		
Semantic labels	Working space	Rest space	Travel space	Category of products	Entry and exit	Travel space
	Configuration	Kitchen	Corridors	Fruit and vegetables	Entrance	Main pathway
	Pricing	Coffee	X	Dairy	Checkout	Racetrack
	Coordinator	Centers	X	Frozen products	X	X
	Printer, Copy	X	X	Drinks	X	X
	Meeting	X	X	Bakery	X	X
	Base space	X	X	Beauty	X	X

point coordinates and entity labels in indoor spaces. A list of semantic labels, defined in the MIT Badge and the sCREEN datasets, is shown in Table 5.1.

A semantic trajectory is then defined as follows:

$$T = \{((x_1, y_1), s_1, t_1), ((x_2, y_2), s_2, t_2), ..., ((x_i, y_i), s_i, t_i), ..., ((x_n, y_n), s_n, t_n)\} \tag{5.2}$$

where (x_i, y_i) and s_i are coordinates and semantic labels of point i, respectively, at the timestamp t_i. n is the length of trajectory T. LCSS_IS of R and S trajectories is defined in Equation (5.3).

$$LCSS\_IS(R,S) = \begin{cases} 0, \text{if } m = 0 \text{ or } n = 0 \\ max\{LCSS\_IS(Rest(R),S), LCSS\_IS(R,Rest(S)), \\ LCSS\_IS(Rest(R),Rest(S)) + \lambda + d\}, \\ \text{if } IndoorDist(r_m,s_n) < \eta \text{ and } \mid m - n \mid \leq \nu \\ max\{LCSS\_IS(Rest(R),S), LCSS\_IS(R,Rest(S)), \\ LCSS\_IS(Rest(R),Rest(S)) + \lambda\}, \text{otherwise} \end{cases} \tag{5.3}$$

where the parameter d belongs to the range of $(0,1)$. $\lambda = 1 - d$ if two points r_m and s_n are the same semantic, otherwise $\lambda = 0$. In this work, d represents the spatial proximity between the two points while λ indicates their semantic similarity. When the sum of d and λ equals 1, two points are spatially close and have the same semantics. In addition, the value of d may be chosen based on the application. For example, $d > 0.5$ if the focus is on the spatial aspect over the semantic information and otherwise. When $d = 0.5$, the spatial proximity and semantic similarity are considered equally. $IndoorDist(r_m, s_n)$ is the indoor walking distance between two points r_m and s_n, which is used to measure the spatial proximity between them. Similar to LCSS, η and ν are pre-defined parameters. If η is a distance threshold to determine if two points are close to one another, ν controls how far in time two points are matched.

In addition, two variants of LCSS_IS are also considered: LCSS using indoor walking distance called LCSS_IWD in Equation (5.4) and LCSS using a semantic label called LCSS_SL in Equation (5.5).

$$LCSS\_IWD(R,S) = \begin{cases} 0, \text{if } m = 0 \text{ or } n = 0 \\ LCSS\_IWD(Rest(R),Rest(S)) + 1, \\ \text{if } IndoorDist(r_m,s_n) < \eta \text{ and } \mid m - n \mid \leq \nu \\ max\{LCSS\_IWD(Rest(R),S), LCSS\_IWD(R,Rest(S))\}, \text{otherwise} \end{cases} \tag{5.4}$$

$$LCSS\_SL(R,S) = \begin{cases} 0, \text{if } m = 0 \text{ or } n = 0 \\ max\{LCSS\_SL(Rest(R), S), LCSS\_SL(R, Rest(S)), \\ LCSS\_SL(Rest(R), Rest(S)) + \lambda + d\}, \\ \text{if } |r_{x,m} - s_{x,n}| < \eta \text{ and } |r_{y,m} - s_{y,n}| < \eta \text{ and } \mid m - n \mid \leq \nu \\ max\{LCSS\_SL(Rest(R), S), LCSS\_SL(R, Rest(S)), \\ LCSS\_SL(Rest(R), Rest(S)) + \lambda\}, \text{otherwise} \end{cases} \tag{5.5}$$

Equation (5.4) shows that LCSS_IWD evaluates the similarity between two points r_m and s_n only using indoor walking distance. In contrast, the similarity between these two points in the LCSS_SL and LCSS_IS metrics is considered from both spatial and semantic aspects. Nevertheless, to estimate the spatial proximity between two points, LCSS_SL uses their coordinates' differences between the points as in the original LCSS, while LCSS_IS explores their indoor walking distance. In other words, LCSS_IS is a combination of LCSS_IWD and LCSS_SL.

The above metrics are normalized between 0 and 1, as reported elsewhere [39]. For example, the normalization of LCSS_IS is defined as follows:

$$Norm_{LCSS\_IS(R,S)} = \frac{LCSS\_IS(R,S)}{min(m,n)} \tag{5.6}$$

from $Norm_{LCSS\_IS(R,S)}$, the distance between two trajectories is determined: $Dist_{(R,S)} = 1 - Norm_{LCSS\_IS(R,S)}$. When the distance equals 0, two trajectories are the same, while 1 indicates that they are entirely different.

5.2.3 Determination for Eps Value

In these two parameters of DBSCAN, MinPts can be chosen more easily based on the user's knowledge of the dataset [60]. In contrast, Eps has a larger effect on the result of clusters and outliers. If Eps is chosen incorrectly, DBSCAN fails to discover clusters of the dataset. For example, when Eps is small, the number of clusters increases, and only a small number of data points are grouped. Moreover, there are many data points found as outliers. When Eps is large, the number of clusters decreases, and there will be few outliers. Therefore, choosing an appropriate Eps value is a challenge for DBSCAN. This study focuses on choosing the value of Eps.

Eps is a vital input parameter of DBSCAN. The value of Eps directly affects the number of clusters and outliers. If Eps is chosen incorrectly, DBSCAN fails to discover clusters of the data set. For example, when Eps is small, the number of clusters increases, and only a small number of data points is grouped. Moreover, there are many data points found as outliers. In contrast, when Eps is large, the number of clusters decreases, and there is a small number of found outliers. Therefore, choosing the optimal Eps is a challenge for the DBSCAN algorithm.

Inspired by existing cluster validity indices [58], this paper proposes a DBSCAN cluster validity index called DCVI. This index is used to choose the Eps value by estimating the quality of DBSCAN. Unlike other clustering algorithms, the clustering results of DBSCAN contain clusters and outliers. Therefore, the quality of DBSCAN is estimated based on measuring the separation between clusters, the separation between outliers and clusters, and the compactness within each cluster. DCVI is defined as the following equation:

$$DCVI = \frac{Min_{l=\{1,\dots,\ K-1\},k=\{l+1,\dots,\ K\}} InterCD(C_l, C_k) + Min_{o=\{1,2\dots,\ O\},k=\{1,\dots,\ K\}} OCD(o, C_k)}{(Min_{k=\{1,\dots,\ K\}} IntraCD(C_k))^{\rho}} \tag{5.7}$$

In Equation (5.7), $InterCD(C_l, C_k)$ is the distance between clusters C_l and C_k. $OCD(o, C_k)$ is the distance between outlier o and cluster C_k. $IntraCD(C_k)$ is the intra-cluster distance of cluster C_k. ρ controls the contribution of the intra-cluster distance to DCVI.

In this work, $InterCD(C_l, C_k)$ is the average distance between points in cluster C_l and points in cluster C_k. The minimum distance between outlier o and all points in cluster C_k is chosen to calculate $OCD(o, C_k)$. $IntraCD(C_k)$ is the average distance between points within cluster C_k. DBSCAN can be considered successful for grouping data points when the outliers are far from clusters, the clusters are separated from each other, and the compactness of clusters is strong. Therefore, the Eps value is chosen when DCVI is maximum.

Figure 5.4 presents a way to determine the Eps value based on DCVI. First, the Eps value is changed from the minimum value to the maximum value of the trajectory distance. According to each Eps value, clusters of the dataset are found using DBSCAN. Then, the value of DCVI is calculated at this Eps value. Finally, the Eps value is chosen according to the maximum value of DCVI.

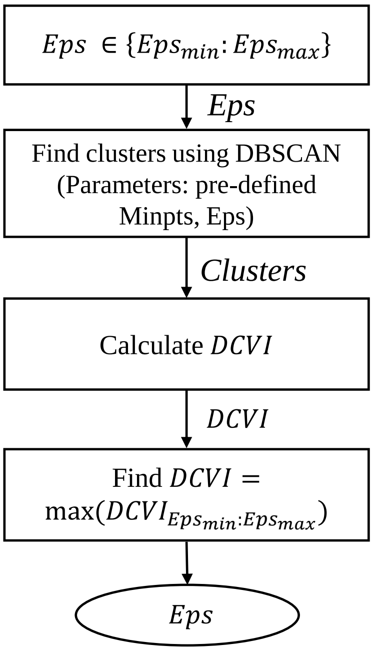

$Eps \in \{Eps_{min}: Eps_{max}\}$
Eps
Find clusters using DBSCAN (Parameters: pre-defined Minpts, Eps)
$Clusters$
Calculate $DCVI$
$DCVI$
Find $DCVI = \max(DCVI_{Eps_{min}:Eps_{max}})$
Eps

Figure 5.4: Flowchart for choosing the Eps value using DCVI.

5.3 Performance Evaluation

The proposed framework based on DBSCAN is evaluated using two datasets: The MIT Badge dataset and the sCREEN dataset. The data preprocessing in the datasets is performed as in the subsection 4.1.3.1.

5.3.1 Parameter Determination for Distance Metric and DBSCAN

5.3.1.1 Parameter Determination for Distance Metric

This study proposes the LCSS_IS measurement presented in Section 5.2.2 to measure the distance between trajectories. In this metric, the parameter d is set to $d = 0.5$. The value of parameter λ is $\lambda = 1 - d = 0.5$ when two points are the same semantic labels. This implies that the weights assigned to spatial closeness and semantic similarity are equivalent when estimating the similarity of the points. Moreover, parameter $\eta = 3500$ with the MIT Badge dataset and $\eta = 5$ with the sCREEN dataset. In this work, the value of η is set based on the floor plan scale and the knowledge about human movement behavior in indoor spaces. Here, η is a distance threshold for determining the spatial closeness between points. The parameter ν shows how far in time a point from one trajectory is matched to a point in another trajectory. In this study, the value of ν is set to the maximum point number of two considering trajectories. This means there is no time constraint when estimating the similarity between two points of two trajectories. Since the trajectories are extracted in a short time window, each point from one trajectory is matched to all points in another trajectory over the whole time window.

5.3.1.2 Determining the Eps Value for DBSCAN

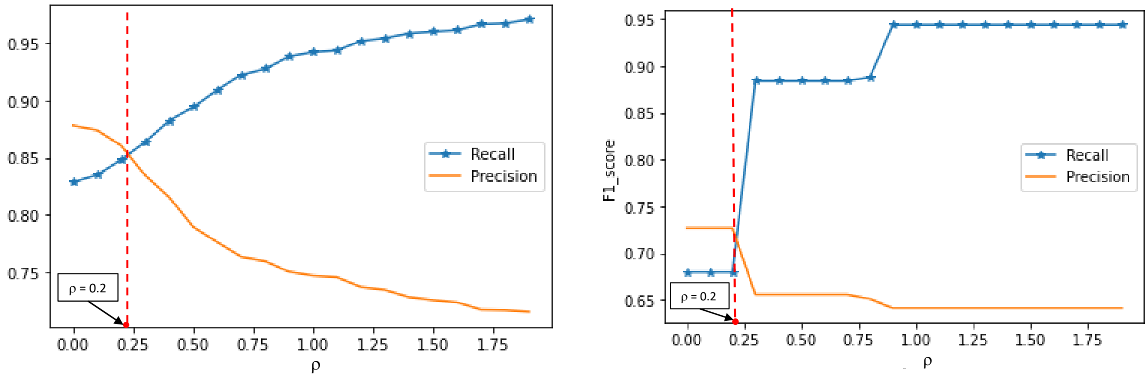
DBSCAN requires two input parameters: MinPts and Eps. Based on previous studies [72, 77], MinPts was chosen to be twice the dimensions of data points in the feature space, i.e., $MinPts = 2 \times dim$. In this work, with the Euclidean, EDR, LCSS, and LCSS_IWD measures, the feature space dimensions for the location points include the x-coordinate, y-coordinate, and timestamp. In contrast, the data space has four dimensions with LCSS_SL, LCSS_IS, and ISTSM because the semantic attribute of the location point is added. Therefore, the MinPts is set to six for the distance metrics in the three-dimensional space and eight for the others in the four-dimensional space.

The value of Eps is chosen using the DCVI metric in Equation (5.7). Note that the ρ parameter, which controls the role of intra-cluster distance, needs to be determined before calculating DCVI. In this work, ρ is chosen using the performance of the algorithm when detecting noise. Noise is first generated with a completely different distribution with anomaly types. This step ensures that the determination of the ρ parameter is independent of the algorithm performance evaluation when detecting anomalies. Noise is then injected into the dataset as anomalies. The algorithm performance for detecting noise is estimated with the different ρ values. Finally, the value of ρ, which corresponds to the best performance of the algorithm is selected.

- **Noise creation**. Noise points are added to the original trajectory as the following equation [36]:

  $$(x_{noise}, y_{noise}) = (x + randn \times X, y + randn \times Y) \tag{5.8}$$

  where (x, y) and (x_{noise}, y_{noise}) are coordinates of points before and after adding noise, respectively. $randn$ is a random value created from Gaussian distribution with mean 0 and variance 1. X and Y are constants that control the proximity between the noise point and the original point. $X = 7000$ and $Y = 5000$ with the MIT Badge dataset and $X = 10$ and $Y = 10$ with the sCREEN dataset. In addition, the ratio of noise point number in each trajectory is chosen randomly in the range $[0.3, 1]$. The starting point for adding noise is chosen randomly from the original trajectory.



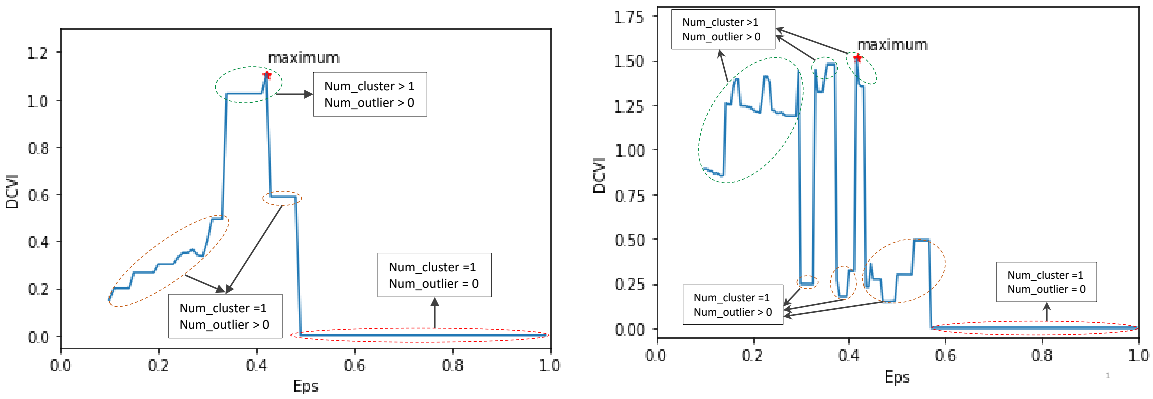
(a)

(b)

Figure 5.5: Selecting ρ value based on algorithm performance when detecting noise: (**a**) MIT Badge dataset. (**b**) sCREEN dataset.

- **Choosing ρ parameter**. The value of ρ is selected based on the trade-off between the algorithm's recall and precision. In this work, ρ corresponds to the intersection point of recall and precision. This means that the algorithm needs to ensure anomaly detection ability while maintaining precision. Figure 5.5a,b show the performance of the algorithm when detecting noise for the MIT Badge and the sCREEN datasets, respectively. The value of ρ, which equals 0.2 for both datasets, is determined from these figures.

- **Choosing Eps parameter**. After selecting ρ, the Eps value is determined using the DCVI metric, as shown in Figure 5.4. The Eps value is chosen according to the maximum value of DCVI. Note that, in the MIT Badge dataset, data are divided into timeslots, and the algorithm is performed following each timeslot. Therefore, there are 54 Eps values according to 54 timeslots. Here, this paper shows only one chosen Eps value according to slot 0, which is 0.42 at the red point in Figure 5.6a. On the other hand, with the sCREEN dataset, data are not divided into timeslots, so only one Eps value (i.e., 0.41 at the red point as in Figure 5.6b) is selected. In Equation (5.7), it should be noted that, if the number of clusters and outliers are equal to 1 and 0, respectively, $InterCD(C_l, C_k)$ and $OCD(o, C_k)$ are equal to 0. As in Figure 5.6, if Eps is too large, there is no outlier, and all trajectories are grouped into one cluster. In this case, DCVI equals the minimum value at 0 according to the part shown by the



(a)

(b)

Figure 5.6: Selecting the Eps value: (**a**) MIT Badge dataset. (**b**) sCREEN dataset.

red ellipse. In addition, if the number of clusters is higher than one and the number of outliers is greater than 0, DCVI is large according to the portion indicated in the green ellipses. The yellow ellipses show that the DCVI is small at Eps values, where only one cluster is found, and the number of outliers is greater than 0.

5.3.2 Result Analysis

In this work, we use three metrics: recall, precision, and F1-score to estimate the algorithm performance. The proposed method is evaluated and compared with four baselines.

- **EMM.** The work in [33] detects anomalous trajectories using an EMM.
- **Density method.** This method detects a trajectory as an anomaly if its density is smaller than a given threshold. The density method has been used in studies [31,33,79].
- **Hierarchical clustering.** The hierarchical clustering-based anomaly detection method is proposed in [30]. This work detects abnormal trajectories based on their closeness with extracted clusters from datasets.
- **Spectral clustering.** From a previous study [30], the hierarchical clustering algorithm is replaced with the spectral clustering algorithm for finding clusters from datasets. Anomalies are found in the same way in [30].

The baselines, except for EMM, also require a distance measure to calculate similarity in the trajectories. In this work, these baselines use four existing distance measures: Euclidean, EDR, LCSS, and ISTSM. Our DBSCAN-based anomaly detection method is evaluated using seven existing and proposed distance metrics: Euclidean, EDR, LCSS, ISTSM, LCSS_IWD, LCSS_SL, and LCSS_IS. The following subsections present the outcomes of methods over the various anomaly types.

5.3.2.1 Anomaly as Pricing Group

This subsection only estimates the performance of the methods using the original MIT Badge dataset. Because there is no labeled anomaly in the dataset, a hypothesis is given for creating anomalies. In particular, in the MIT Badge dataset, the configuration group accounts for approximately 70% of the total while the pricing group is only 20%. Therefore, it is assumed that the workers' movement in the former is normal, while the workers' movement in the latter is abnormal. Moreover, the number of normal and abnormal trajectories for evaluating the algorithm is chosen equally. The algorithm is fairly estimated owing to the balance between abnormal and normal sample numbers in the test dataset. Table 5.2 lists the results of baselines and the proposed method for detecting anomalies as the pricing group.

In particular, EMM achieves a high recall value of 89.55% compared with a precision of only 66.87%. Because EMM evaluates the abnormality at the level of the trajectory point, and a trajectory is detected as an anomaly if it contains at least one anomalous point. This means that EMM prioritizes seeking the abnormality of the trajectory point. Therefore, there are many normal trajectories detected as anomalies, and EMM precision is low.

The methods, which use distance metrics, evaluate the abnormality at the trajectory level. With these methods, LCSS obtains a higher precision than EDR and ISTSM. One possible explanation is that LCSS aims to find the longest common subsequence between two trajectories, ignoring the unmatched points. Therefore, LCSS tends to seek out the normality of a trajectory rather than its abnormality. This explains why the methods obtain better results in precision. Moreover, in four distance metrics (Euclidean, EDR, LCSS, and ISTSM), the biggest disadvantage of Euclidean, compared with the remaining metrics, is that it can not be applied directly to the trajectories with different lengths. In this work, to use the Euclidean metric, the missed points are interpolated based on existing points. The linear interpolation method was used [80]. After interpolation, the Euclidean distance may estimate the similarity of trajectories quite well because this metric uses the absolute difference between points. Therefore, the performance of methods using Euclidean

Table 5.2: Results with anomaly as pricing group.

Method	Measure	Recall	Precision	F1-score
EMM		0.8955	0.6687	0.7635
Density	Euc	0.7634	0.7241	0.7408
	EDR	0.9929	0.6394	0.7779
	LCSS	0.8221	0.7692	0.7927
	ISTSM	0.7914	0.7639	0.7738
Hierachical Clustering	Euc	0.8702	0.7206	0.787
	EDR	0.8923	0.6738	0.7676
	LCSS	0.7826	0.7082	0.7435
	ISTSM	0.9288	0.6701	0.7783
Spectral Clustering	Euc	0.8681	0.7198	0.7858
	EDR	0.8823	0.6964	0.7782
	LCSS	0.816	0.7227	0.7664
	ISTSM	0.9207	0.6737	0.7777
Proposed	Euc	0.7897	0.8214	0.8052
	EDR	0.9035	0.7479	0.8179
	LCSS	0.8363	0.8085	0.8216
	ISTSM	0.8637	0.7083	0.7742
	LCSS_IWD	0.6856	0.8817	0.771
	LCSS_SL	0.9539	0.8022	0.8708
	LCSS_IS	0.8997	0.8838	**0.8903**

is comparable to EDR, LCSS, and ISTSM.

ISTSM, which uses space and semantic aspects to determine the trajectories' similarity, obtains a higher recall than precision for all methods. One possible explanation is that ISTSM prioritizes seeking the difference between trajectories. In ISTSM, if two points have different semantic labels, evenly close in space, a maximum value of 1 is assigned to the substitution cost of the two points. Therefore, trajectories may be detected as anomalies more easily, and recall is high. However, precision is low, so F1-score for this measure is low.

The proposed method with LCSS_IS outperforms the other methods in the F1-score. LCSS_IS, which is extended from LCSS, shows an improved ability to distinguish between two trajectories using the indoor walking distance and semantic labels to determine points' similarity. This means that LCSS_IS uses the spatial proximity and semantic information to estimate the distance of the trajectories rather than that solely based on the space aspect as LCSS, EDR, and Euclidean. Unlike ISTSM, which only uses the indoor walking distance for the same labels and ignores it for the different labels, LCSS_IS uses the indoor walking distance for both the same and different labels. Moreover, the performance of the two variants, LCSS_IWD and LCSS_SL, are also evaluated. In LCSS_IWD, the indoor walking distance is used to determine the similarity between the two points. Meanwhile,

Table 5.3: Results with rare location visiting anomaly on the MIT Badge dataset.

τ (%)		0.5			1		
Method	Measure	Recall	Precision	F1-score	Recall	Precision	F1-score
EMM		1	0.6082	0.7563	1	0.6082	0.7563
Density	Euc	0.9528	0.7746	0.8537	0.9528	0.7746	0.8537
	EDR	0.9964	0.6826	0.8098	1	0.6834	0.8115
	LCSS	0.9911	0.8334	0.9052	1	0.8346	0.9096
	ISTSM	0.716	0.7853	0.7478	1	0.7267	0.8417
HC	Euc	0.9379	0.7475	0.8285	1	0.7465	0.8532
	EDR	0.8953	0.6542	0.7554	1	0.6787	0.8084
	LCSS	0.9132	0.7085	0.797	1	0.7265	0.8412
	ISTSM	0.968	0.6401	0.7706	1	0.6894	0.8159
SC	Euc	0.9383	0.746	0.8277	1	0.745	0.8522
	EDR	0.8766	0.6993	0.7763	1	0.7255	0.8402
	LCSS	0.8973	0.7368	0.8074	1	0.8365	0.9098
	ISTSM	0.8411	0.6448	0.7299	1	0.6837	0.8119
Proposed	Euc	0.9941	0.7908	0.8792	1	0.8351	0.9083
	EDR	0.8401	0.7961	0.8153	1	0.8228	0.9025
	LCSS	1	0.8585	0.923	1	0.8585	0.923
	ISTSM	0.8792	0.7189	0.7886	1	0.8141	0.897
	LCSS_IWD	0.741	0.9072	0.8145	0.983	0.927	0.9543
	LCSS_SL	0.9977	0.8114	0.8944	1	0.8117	0.8956
	LCSS_IS	0.9629	0.91	0.9356	1	0.9137	**0.9545**

LCSS_SL uses both the space and semantic aspects to estimate the similarity. Note that, in LCSS_SL, the spatial proximity is determined using the coordinates' norm between two points as in LCSS. As can be seen in Table 5.2, LCSS_IWD achieves high precision, while LCSS_SL improves recall. Since LCSS_IS is a combination of LCSS_IWD and LCSS_SL, the performance of the proposed method with LCSS_IS is improved in the recall, precision and F1-score.

5.3.2.2 Synthetic Anomalies

This subsection estimates the performance of the methods for two synthetic anomaly types: Rare location visits and route anomalies. To estimate the anomaly detection ability of the methods over various anomaly types, in the test dataset, abnormal trajectories are replaced according to their type, and the normal trajectories remain.

Tables 5.3 and 5.4 depict the results of detecting rare location visit anomalies on the MIT Badge and the sCREEN datasets, respectively. In the experiments, $\tau = \{0.5, 1\}$, which is the ratio of the shifted point number to rare locations in each trajectory. The outcome of methods is better when τ increases. This is because the abnormality is higher when a trajectory remains at a rare location over a longer time. Tables 5.5 and 5.6 show the

Table 5.4: Results with rare location visiting anomaly on the sCREEN dataset.

τ (%)		0.5			1		
Method	Measure	Recall	Precision	F1-score	Recall	Precision	F1-score
EMM		1	0.5828	0.7364	1	0.5828	0.7364
Density	Euc	1	0.672	0.8038	1	0.672	0.8038
	EDR	0.748	0.6608	0.7017	1	0.7225	0.8389
	LCSS	1	0.7102	0.8305	1	0.7102	0.8305
	ISTSM	0.884	0.6637	0.7582	1	0.6906	0.817
HC	Euc	1	0.5176	0.6821	1	0.5176	0.6821
	EDR	1	0.5435	0.7042	1	0.5435	0.7042
	LCSS	1	0.7102	0.8305	1	0.7102	0.8305
	ISTSM	1	0.6443	0.7837	1	0.6443	0.7837
SC	Euc	1	0.5252	0.6887	1	0.5252	0.6887
	EDR	1	0.5495	0.7093	1	0.5495	0.7093
	LCSS	0.968	0.7634	0.8536	1	0.7692	0.8695
	ISTSM	1	0.6631	0.7974	1	0.6631	0.7974
Proposed	Euc	1	0.7937	0.885	1	0.7937	0.885
	EDR	0.672	0.7636	0.7149	1	0.8278	0.9058
	LCSS	0.996	0.8111	0.8941	1	0.8117	0.8961
	ISTSM	1	0.6684	0.8013	1	0.6684	0.8013
	LCSS_IWD	1	0.8065	0.8929	1	0.8065	0.8929
	LCSS_SL	1	0.7553	0.8606	1	0.7553	0.8606
	LCSS_IS	0.928	0.8722	0.8992	1	0.8803	**0.9363**

performance of methods in both datasets when detecting route anomalies.

In the evaluated baselines, EMM achieves the highest recall value with synthetic anomaly types on both datasets. As previously stated, a trajectory is detected as an anomaly by EMM if the trajectory has at least one abnormal point. Moreover, synthetic anomalies contain points with high abnormality, and the points are identified easily by EMM. Therefore, EMM may detect all synthetic anomalies. However, because a trajectory is detected easily as an anomaly by EMM, the precision of this method is low. Hence the F1-score is also low.

The baselines, which use distance metrics, also obtain a very high recall value with all types of synthetic anomalies on both datasets. Because the synthetic anomalies have a strong abnormality at the trajectory level compared to the normal trajectories, they may be detected by the baseline methods. Nevertheless, the precision of the baselines is low. As the baselines detect anomalies using thresholds (i.e., the density threshold in the density method and the distance threshold between a trajectory and clusters in hierarchical and spectral clustering), their performance is affected by the value of the thresholds. However, choosing an appropriate threshold value for detecting anomalies is a challenge. With baselines, thresholds are chosen based on the knowledge about datasets. Therefore, the methods may not obtain the best performance in precision.

Table 5.5: Results with route anomalies on the MIT Badge dataset.

Route anomaly		Detour anomaly			Random route anomaly		
Method	Measure	Recall	Precision	F1-score	Recall	Precision	F1-score
EMM		1	0.6082	0.7563	1	0.6082	0.7563
Density	Euc	0.815	0.8229	0.8184	0.815	0.8229	0.8184
	EDR	1	0.6834	0.8115	1	0.6834	0.8115
	LCSS	0.9974	0.8328	0.9076	0.9937	0.8323	0.9057
	ISTSM	1	0.7267	0.8417	0.9988	0.6853	0.8128
HC	Euc	1	0.7465	0.8532	1	0.7465	0.8532
	EDR	0.976	0.6734	0.7968	0.99	0.6765	0.8036
	LCSS	0.9903	0.7272	0.8382	0.9846	0.7261	0.8354
	ISTSM	1	0.6894	0.8159	0.9843	0.6439	0.7784
SC	Euc	1	0.745	0.8522	1	0.745	0.8522
	EDR	0.9751	0.6338	0.7682	0.9914	0.7239	0.836
	LCSS	0.9954	0.7555	0.8582	0.9988	0.7562	0.8599
	ISTSM	1	0.6837	0.8119	0.9048	0.6614	0.7639
Proposed	Euc	1	0.8351	0.9083	1	0.8351	0.9083
	EDR	0.8862	0.8703	0.8761	0.9224	0.8839	0.9007
	LCSS	1	0.8585	0.923	1	0.8424	0.9135
	ISTSM	1	0.8141	0.897	0.9685	0.7052	0.8147
	LCSS_IWD	0.9983	0.9291	0.9624	0.9928	0.9287	0.9597
	LCSS_SL	1	0.8117	0.8956	1	0.8114	0.8954
	LCSS_IS	0.994	0.9132	0.9515	0.9885	0.9132	**0.9489**

Table 5.6: Results with route anomalies on the sCREEN dataset.

Route anomaly		Detour anomaly			Random route anomaly		
Method	Measure	Recall	Precision	F1-score	Recall	Precision	F1-score
EMM		1	0.5828	0.7364	1	0.5828	0.7364
Density	Euc	1	0.6964	0.821	1	0.6964	0.821
	EDR	1	0.7225	0.8389	1	0.7225	0.8389
	LCSS	1	0.7102	0.8305	1	0.7102	0.8305
	ISTSM	1	0.6812	0.8104	1	0.6812	0.8104
HC	Euc	1	0.5176	0.6821	1	0.5176	0.6821
	EDR	1	0.5435	0.7042	1	0.5435	0.7042
	LCSS	1	0.7102	0.8305	1	0.7102	0.8305
	ISTSM	1	0.6443	0.7837	1	0.6443	0.7837
SC	Euc	1	0.5252	0.6887	1	0.5252	0.6887
	EDR	1	0.5495	0.7093	1	0.5495	0.7093
	LCSS	1	0.7692	0.8695	1	0.7692	0.8695
	ISTSM	1	0.6631	0.7974	1	0.6631	0.7974
Proposed	Euc	1	0.7937	0.885	1	0.7937	0.885
	EDR	1	0.8278	0.9058	1	0.8278	0.9058
	LCSS	1	0.8117	0.8961	1	0.8117	0.8961
	ISTSM	1	0.6684	0.8013	1	0.6684	0.8013
	LCSS_IWD	1	0.8065	0.8929	1	0.8065	0.8929
	LCSS_SL	1	0.7553	0.8606	1	0.7553	0.8606
	LCSS_IS	1	0.8803	0.9363	1	0.8803	**0.9363**

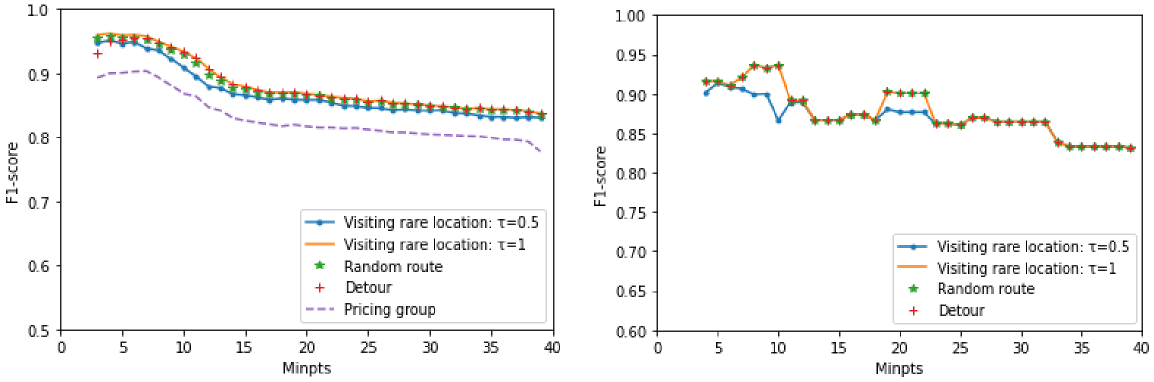
In contrast, the proposed method improves precision value compared to the baselines while achieving a high recall value. This is because our method detects anomalies using the Eps value. The appropriate value of Eps is determined based on estimating the clustering quality of DBSCAN using the DCVI metric. Therefore, the performance of the proposed method is improved significantly compared with the baselines in precision and F1-score with synthetic anomalies. Moreover, in the proposed method, LCSS_IS outperforms other distance metrics in the F1-score, except for detecting route anomalies on the MIT Badge dataset. In this case, the F1-score obtains the highest value of approximately 96% with the variant

LCSS_IWD. With LCSS_IS, this method also achieves a high F1-score of approximately 95% and outperforms the other baselines.

Because normal trajectories are maintained in the test dataset when estimating the method performance over other anomaly types, the result of *False Positive* samples does not change for each method. Therefore, if the result of *True Positive* samples of a method is the same as with different anomaly types, the F1-score of the method is equal according to these anomaly types. For example, the F1-score for detecting detour and random route anomalies is the same following each method in the sCREEN dataset.

5.3.2.3 Effect of the MinPts Parameter on Performance

This subsection presents the performance of the proposed method over various anomaly types when varying the MinPts parameter of DBSCAN. It is known that MinPts has less influence than Eps on the clustering quality of DBSCAN. The value of MinPts often is chosen based on the user's knowledge about datasets [79] (i.e., MinPts can be set to twice the dimensions of the data). In this experiment, the MinPts change in the range of $[3, 40]$. With a given value of MinPts, an appropriate value of Eps is determined using the DCVI metric. Then, the chosen Eps value is used for clustering trajectories and detecting anomalies. Figure 5.7a,b show the results in terms of the F1-score for the MIT Badge and sCREEN datasets, respectively. Note that only the LCSS_IS measure is used in this experiment. The trend of changing performance is similar to all anomaly types. The proposed method achieves the best results around MinPts = 8 on the two datasets. When $MinPts > 10$, the performance of the proposed method is decreased. This may be explained by the fact that many trajectories are detected as noise when the MinPts that are chosen are too large, which may affect the clustering quality of DBSCAN. Therefore, the effectiveness of detecting anomalies is degraded.



(a)

(b)

Figure 5.7: Performance of the proposed method when varying *MinPts*: (**a**) MIT Badge dataset. (**b**) sCREEN dataset.

5.4 Chapter Summary

This paper proposed a two-phase framework for detecting indoor human trajectory anomalies based on DBSCAN. This proposed method discovered trajectory clusters in the dataset. A newly coming trajectory is detected as an anomaly if it does not belong to any clusters of trajectories. A novel measure called LCSS_IS was proposed to determine the similarity of the trajectories, which was extended from the original LCSS for discovering the features of indoor human movement. In particular, the indoor walking distance and semantic information were combined to estimate the similarity of trajectories. Therefore, LCSS_IS measured the distance of trajectories in indoor spaces more precisely than existing distance metrics. Furthermore, a novel cluster validity index, DCVI, was proposed to choose the Eps parameter for DBSCAN. DCVI was designed to measure the separation between clusters, the separation between clusters and outliers, and the compactness within each cluster. An appropriate Eps value was determined corresponding to the maximum value of DCVI. The proposed method was evaluated on two real datasets: MIT Badge and sCREEN. The different anomalous trajectory types were also detected in this work. The proposed method showed impressive performance and outperformed the baselines.

In this work, there are a few limitations. First, several features of trajectory data, such as speed and moving direction, were not considered for calculating the distance between trajectories. Second, the proposed method was not evaluated on datasets with complex floor plans, such as buildings with many floors. We plan to extend this work to address the above limitations in a future study.

Chapter 6

Deep Learning-based Anomalous Trajectory Detection

To identify anomalous human trajectories, understanding features of their movement plays an important role. Therefore, in this work, a Transformer encoder and self-organizing map-based model called TENSO is proposed to learn trajectory characteristics for detecting anomalies. In particular, the proposed model learns the internal characteristics of normal trajectories and clusters of normal trajectory representations in a latent space. To learn the internal characteristics of normal trajectories, the encoder of Transformer with a self-attention mechanism first encodes trajectories into sequences of embedding vectors of trajectory points in the latent space. Then, a decoder reconstructs the trajectories from the latent space. In addition, to learn clusters of normal trajectory representations in the latent space, the self-organizing map (SOM) layer is used, which gets its input as the output of the Transformer encoder. In the training phase, the TENSO model is trained using a total loss of trajectory reconstruction and SOM losses. In the anomaly detection phase, a test trajectory is evaluated to determine whether it is an anomaly based on trajectory reconstruction errors and the quantization error on the SOM. In this phase, a new metric is proposed which, namely WS, is the weighted sum of recall and precision to choose the appropriate threshold for detecting anomalies. The TENSO model-based framework is evaluated using two real trajectory datasets: MIT Badge and sCREEN.

6.1 Definitions

This section presents definitions of Transformer encoder and SOM in detail as follows.

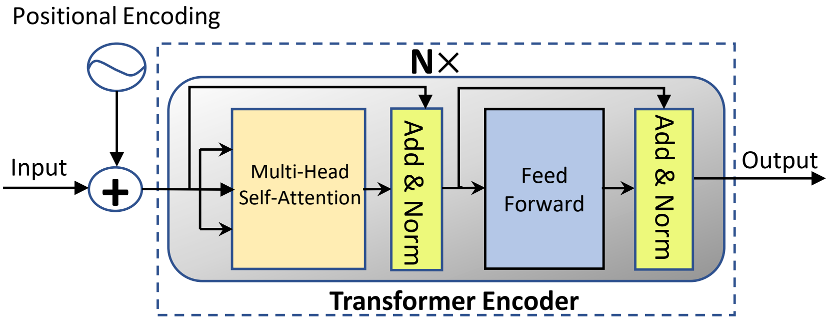


Figure 6.1: Transformer encoder

6.1.1 Transformer encoder

Transformer is a network architecture based on an attention mechanism, and consists of two main parts: an encoder and a decoder [81]. This neural network can effectively replace recurrent neural network (RNN) and convolutional neural network (CNN) architectures in sequential tasks. In this work, we only consider the Transformer encoder, which is presented in Figure 6.1. It contains two sub-layers: a layer of multi-head self-attention and a position-wise fully connected feed-forward network.

- **Multi-head self-attention.** A single attention layer called Scaled Dot-Product Attention is the main calculation layer in multi-head attention. It consist of three inputs: queries, keys of dimension d_k, and values of dimension d_v. To determine the weights of the values, they compute the dot products of the query with each key, divide each result by $\sqrt{d_k}$, and then apply a softmax function. The queries, keys, and values are packed as separate matrices: Q, K, and V. The equation for a single attention is

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{6.1}$$

Note that in the Transformer encoder, the keys, queries, and values are obtained from the same place, and the attention is known as self-attention. Since performing parallel self-attention layers on multiple subspaces of the input sequence is better than on a single space, the input sequence is projected h times with learned linear projections before applying self-attention layers. This mechanism is known as multi-head self-attention in the Transformer encoder and is performed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O, \tag{6.2}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and the linear projections are matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

- **Position-wise feed-forward network.** This sub-layer is applied to each position of the output sequence of multi-head self-attention. It consists of a dense layer with ReLU activation and a linear layer:

  $$\text{FFN}(x) = W_2 \times \text{ReLU}(W_1 x + B_1) + b_2, \tag{6.3}$$

  where x is the input of the sub-layer, with W_2, W_1, b_1, and b_2 as parameters of the network.

The output from each of these two sub-layers of the Transformer encoder are normalized and added by an Add & Norm layer.

6.1.2 Self-organizing map

The SOM is a competitive learning neural network known as a topology-preserving clustering model [82, 83]. This network has two layers: an input layer, and an output layer of interconnected neurons (often called units or nodes) in a two-dimensional grip map.

Assume that a set of input data samples is defined $X = \{x_i\}_{1 \leq i \leq L}, x_i \in \mathbb{R}^D$. A SOM has V units, and each unit is represented by a corresponding prototype vector $\{m_v\}_{1 \leq v \leq V}$. Prototype vectors belong to the same space as input data (i.e., $\mathbb{R}^D$). In each iteration of SOM training, all prototype vectors are updated. The level of the update depends on the correlation between prototype vectors and the input data vector, which is determined by a distance metric. When an input data sample comes, the prototype vector with the smallest distance is determined. The unit on the map that corresponds to this prototype vector is called the best matching unit (BMU). A definition for BMU b_i of x_i is

$$b_i = \underset{v}{\text{argmin}}(\text{Dist}(x_i, m_v)), \tag{6.4}$$

where Dist(.) is the chosen distance metric.

On the grid map, the Manhattan distance $d(i, j)$ between two nodes i and j is determined. A temperature parameter, T, and a neighborhood function, $K^T(d)$, of SOM are also defined. Temperature parameter T at iteration $iter$ is determined as follows:

$$T(iter) = T_{\max}\left(\frac{T_{\min}}{T_{\max}}\right)^{\frac{iter}{Num_{iters}}}, \tag{6.5}$$

where $T_{\max}$ and $T_{\min}$ are the initial and final temperatures, respectively. Num_{iters} is the number of iterations. The function $K^T(d)$ determines the neighborhood radius around a unit on the map, and can be defined as a Gaussian neighborhood function as follows:

$$K^T(d) = e^{-\frac{d^2}{T^2}}. \tag{6.6}$$

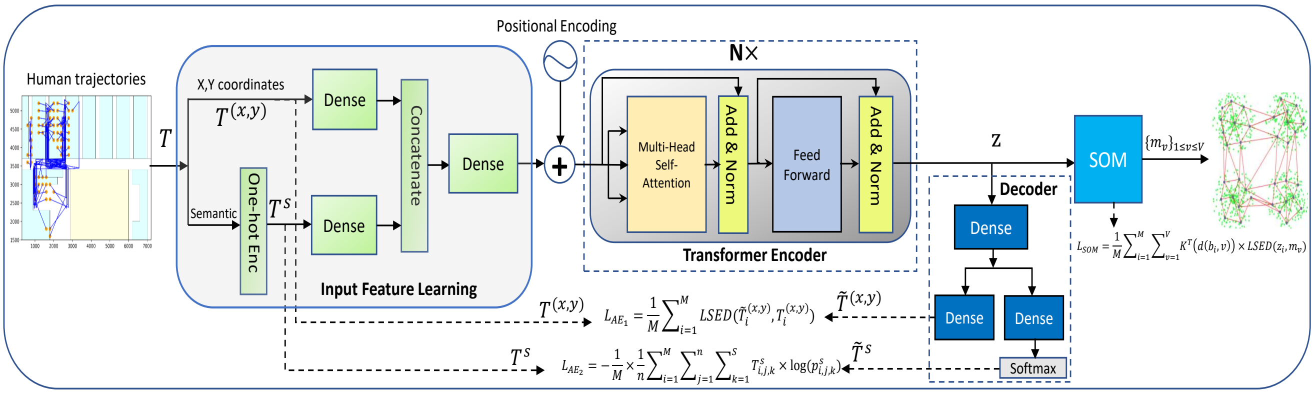
In its training phase, the SOM gets each input sample x_i and updates all prototype vectors by forwarding them to closer x_i. The weight for updating is the value of the neighborhood function around the BMU of x_i. Thus, close neighboring units will be updated more than farther ones. The updating procedure is presented in the following expression:

$$m_v \leftarrow m_v + \chi \times K^T(d(b_i, v))(x_i - m_v), \tag{6.7}$$

where χ is the learning rate.

6.2 Methodology

6.2.1 Model based on the Transformer Encoder and SOM



Human trajectories
T
X,Y coordinates
$T^{(x,y)}$
Semantic
One-hot Enc
T^S
Dense
Dense
Concatenate
Dense
Input Feature Learning
Positional Encoding
N×
Multi-Head Self-Attention
Add & Norm
Feed Forward
Add & Norm
Transformer Encoder
z
SOM
$\{m_v\}_{1\le v\le V}$
$L_{SOM} = \frac{1}{M}\sum_{i=1}^{M}\sum_{v=1}^{V} K^T(d(b_i, v)) \times LSED(z_i, m_v)$
Decoder
Dense
Dense
Dense
Softmax
$T^{(x,y)}$
$L_{AE_1} = \frac{1}{M}\sum_{i=1}^{M} LSED(\tilde{T}_i^{(x,y)}, T_i^{(x,y)})$
$\tilde{T}^{(x,y)}$
T^S
$L_{AE_2} = -\frac{1}{M}\times\frac{1}{n}\sum_{i=1}^{M}\sum_{j=1}^{n}\sum_{k=1}^{S} T_{i,j,k}^S \times \log(p_{i,j,k}^S)$
$\tilde{T}^S$

Figure 6.2: The architecture of the TENSO model

6.2.1.1 Overview of the Proposed Model

In this work, we build a model based on the Transformer encoder and SOM with two main tasks: learning normal trajectory representations, and learning their clusters in latent space. The first task is based on the Transformer encoder, and the second is based on the SOM layer.

Since the proposed TENSO model captures interior features of normal trajectories, anomalous trajectories may not be well reconstructed. Moreover, anomalous trajectory representations in the latent space may not belong to clusters of normal representations, which are also learned by the TENSO model. Thus, the proposed model can detect anomalous trajectories using reconstruction errors and the relationship between representations and clusters of normal trajectory representations in latent space. The architecture of the TENSO model is divided into three main parts as seen in Figure 6.2. The first part learns the input features of the trajectory (the green blocks). The second part includes the Transformer encoder and a decoder. The Transformer encoder learns trajectory representations and encodes them via output sequences in a latent space. A decoder with dense layers is used to reconstruct original trajectories. The third part is the SOM layer, which gets the input from the output of the Transformer encoder. This part learns normal trajectory representation clusters in latent space. Detailed descriptions of the parts are presented as follows.

6.2.1.2 Learning Input Features of Trajectories

To learn the correlation between a trajectory's points and to encode them using the Transformer encoder, the input trajectory points need to be embedded as vector representations. In particular, each trajectory point contains positional and semantic information. Since each trajectory's position belongs to a low-dimensional space (i.e., x and y coordinates), it is projected to a high-dimensional space. This helps the model learn more positional information about the trajectory points. This step is performed using a dense layer. Besides, since the x and y coordinate values may have different ranges, they are normalized to the range of [0,1] by using min-max normalization [84].

To use semantic information for learning trajectory representations, semantic labels need to be embedded as dense vectors. First, semantic labels of trajectory points are converted to one-hot vectors. Then, the one-hot vectors are projected to a high-dimensional space using a dense layer. Two vector representations of semantic and positional information are concatenated and projected in a higher dimension space to obtain the trajectory point's final vector representation.

6.2.1.3 Learning Trajectory Representations

In this subsection, the Transformer encoder and a decoder are presented. First, the Transformer encoder is used to learn the correlation between a trajectory's points and its sequential information. Then, the original trajectory reconstruction from the output sequence of the Transformer encoder is obtained using the decoder.

In the TENSO model, the Transformer encoder gets input as a sequence of trajectory points' embeddings. With the self-attention mechanism, each trajectory's point is encoded as a vector representation that captures the correlation of this point with all trajectory points. Since the attention mechanism does not retain the order of trajectory points by itself, positional encoding is added to the input of the Transformer encoder. The output of the Transformer encoder is a sequence of points' vector representations that captures internal characteristics and the sequential nature of the trajectory. The Transformer encoder consists of N stacked identical layers. Each layer comprises two main sub-layers: multi-head self-attention and position-wise fully connected feed forward layer [81].

A simple decoder with dense layers is used to reconstruct the original trajectory, which ensures the trade-off between the trajectory reconstruction's effectiveness and the computational cost of the model. The decoder consists of one hidden layer and two outputs. The first output reconstructs the trajectory's positional information (i.e., x and y coordinates), and the second reconstructs the semantic label information (i.e., the probabilities that trajectory points' labels belong to semantic labels in the dataset).

6.2.1.4 SOM-based Normal Trajectory Representation Cluster Modeling

This subsection describer the SOM layer for learning normal trajectory representation clusters in latent space, which is trained jointly with the remaining parts of the TENSO model.

The correlations between trajectory representations in latent space may be useful for abnormal trajectory detection tasks. In particular, if a trajectory whose representation does not belong to any clusters in latent space, this trajectory may be an anomaly. Thus, the proposed model learns normal trajectory representation clusters in the latent space to detect anomalous trajectories, which can be performed by the SOM layer.

The input of the SOM is gotten from the output of the Transformer encoder. After training the SOM, the normal trajectory representation clusters in the latent space are modeled by prototype vectors of the SOM. Depending on the dataset and the application, the number of prototype vectors may vary. A small number of units on a map learns a general distribution of datasets, whereas a SOM with many units may represent more detailed datasets. In our work, the number of units is chosen as 100 (i.e., a square map of 10×10). This value is much smaller than the number of trajectories in the training sets (i.e., 10,537 trajectories in the MIT Badge dataset and 11,107 trajectories in the sCREEN dataset).

6.2.1.5 Loss Functions

In the TENSO model, all parameters are trained and updated jointly. A total loss function is designed that comprises three different terms:

$$\text{Total Loss} = L_1 + \gamma_1 \times L_2 + \gamma_2 \times L_{\text{SOM}}, \tag{6.8}$$

where L_1 is the loss for reconstructing x-y coordinates, L_2 is the loss for reconstructing semantic labels, and L_{SOM} is the loss of the SOM; γ_1 and γ_2 are parameters that control the role of component loss functions in the total loss function.

In L_1, we use lock-step Euclidean distance (LSED) to determine the x-y coordinate reconstruction error between the decoded and original trajectories. L_1 is defined as

$$L_1 = \frac{1}{M}\sum_{i=1}^{M} \text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}), \tag{6.9}$$

where M is the number of trajectories used for each parameter update, referred to as batch size. $\tilde{T}_i^{(x,y)}$ and $T_i^{(x,y)}$ are the decoded and original trajectories i^{th}, respectively, which only contain x-y coordinate information of the trajectory points.

$\text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)})$ is defined as follows [53]:

$$\text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}) = \frac{1}{n}\sum_{j=1}^{n} \text{dist}(\tilde{p}_j^{(x,y)}, p_j^{(x,y)}), \tag{6.10}$$

where $\text{dist}(\tilde{p}_j^{(x,y)}, p_j^{(x,y)})$ is the Euclidean distance between two points $\tilde{p}_j^{(x,y)}$ and $p_j^{(x,y)}$, which are the decoded and original points j^{th} of the trajectory i^{th}, and n is the point number of each trajectory.

In TENSO, since the model needs to predict the semantic label for each trajectory point, L_2 is chosen as the cross-entropy loss function, which is used for training models in multiclass classification. The equation for L_2 is

$$L_2 = -\frac{1}{M} \times \frac{1}{n} \sum_{i=1}^{M} \sum_{j=1}^{n} \sum_{k=1}^{S} T^s_{i,j,k} \times \log(p^s_{i,j,k}), \tag{6.11}$$

where S is the number of semantic label classes in the datasets, $T^s_{i,j,k}$ is the value of the semantic label information at the class k^{th} of the point j^{th} of the original trajectory i^{th} (0 if a negative instance and 1 if a positive instance). $p^s_{i,j,k}$ is the probability that the semantic label of point j^{th} of the original trajectory i^{th} belongs to the semantic label class k^{th}.

For the loss of the SOM layer, L_{SOM} is defined by

$$L_{\text{SOM}} = \frac{1}{M} \sum_{i=1}^{M} \sum_{v=1}^{V} K^T(d(b_i, v)) \times \text{LSED}(z_i, m_v). \tag{6.12}$$

The terms in equation 6.12 are explained in Table 6.1.

Table 6.1: Explanation of terms

Term	Meaning
M	The number of input trajectories
V	The number of units on the map
z_i	Latent representation of input trajectory i^{th}
b_i	Best matching unit of z_i on the SOM
m_v	Prototype vector of unit v on the SOM
$d(b_i, v)$	Manhattan distance between b_i and v on the SOM
$\text{LSED}(z_i, m_v)$	Lock-step Euclidean distance between z_i and m_v
$K^T(.)$	The neighbor function

The training procedure of the proposed model is presented in **Algorithm 1**.

6.2.2 Anomalous Trajectory Detection

This subsection discusses abnormal trajectory detection based on the TENSO model. In the training step of TENSO, normal historical trajectories are used. Then, a new trajectory is detected whether it is an anomaly. In particular, the anomaly score of each trajectory is determined based on the trained TENSO model. The trajectory is marked as an anomaly if the anomaly score exceeds a given threshold. Determination of the anomaly score and threshold are presented in detail as follows.

Algorithm 1 Training procedure for TENSO

```
Input: - $X_{Train} = \{T_1, ..., T_{N_{Train}}\}$
       - Number of epochs ($num_{epoch}$)
       - Batch size ($M$)
       - Temperatures $T_{\max}$, $T_{\min}$
Output: - Trained TENSO
Determine the number of iterations:
$Num_{iters} = num_{epoch} \times [\frac{N_{Train}}{M}]$
for $iter \leftarrow 1$ to $Num_{iters}$ do
    Get a batch of trajectories: $\{T_i\}_{i=1}^{M}$
    Determine the latent representation batch: $\{z_i\}_{i=1}^{M}$
    Determine the reconstructed terms of trajectories: $\{\tilde{T}_i^{(x,y)}\}_{i=1}^{M}$ and $\{\tilde{T}_i^{s}\}_{i=1}^{M}$
    Find BMUs of the latent representation batch: $\{b_i\}_{i=1}^{M}$
    Update temperature parameter $T$ using equation 6.5
    Determine neighborhood function $K^T(d(b_i, v))$ using equation 6.6
    Update all parameters of TENSO using the total loss in equation 6.8
end for
```

6.2.2.1 Anomaly Score

To determine the anomaly score, the trajectory reconstruction errors and the quantization error of latent representation on the SOM are used. The trajectory's reconstruction errors consist of two terms: the x-y coordinate reconstruction error and the semantic label reconstruction error.

In two terms of the reconstruction errors, the x-y coordinate reconstruction error is measured using LSED distance, which is defined as the following equation.

$$\mathrm{RE}_1 = \mathrm{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}), \tag{6.13}$$

where $\tilde{T}_i^{(x,y)}$ is the x-y coordinate reconstruction of $T_i^{(x,y)}$.

To obtain the semantic label reconstruction error, the output of the softmax function from the decoder is processed. This output is a sequence of vectors. Vector j^{th} contains probabilities that the semantic label of point j^{th} belongs to label classes in dataset. The error label detection probability of point j^{th} in each trajectory may be determined as $(1 - p^s_{j,k=\text{Target label}})$. $p^s_{j,k=\text{Target label}}$ is correct label detection probability for point j^{th}. We define the trajectory's semantic label reconstruction error as the average value of trajectory point error detection probabilities, which is determined as follows:

$$\mathrm{RE}_2 = \frac{1}{n}\sum_{j=1}^{n}(1 - p^s_{j,k=\text{Target label}}). \tag{6.14}$$

The quantization error is the distance between the latent representation of the trajectory and its BMU's prototype vector on the SOM [85]. In this work, quantization error is also determined using LSED distance as follows:

$$\mathrm{QE_{SOM}} = \mathrm{LSED}(z_i, m_{b_i}), \tag{6.15}$$

where z_i is the latent representation of T_i, and m_{b_i} is the prototype vector of BMU b_i of z_i on the SOM.

The anomaly score of each trajectory is determined as

$$\mathrm{AS} = (\alpha \times \mathrm{RE}_1 + \beta \times \mathrm{RE}_2) + (1 - \alpha - \beta) \times \mathrm{QE_{SOM}}, \tag{6.16}$$

where α and β control the role of terms in the equation for AS. In this work, we set $\alpha = \beta = 0.25$. With these values, the role of the quantization error on the SOM and the reconstruction error by the Transformer encoder and decoder is the same (i.e., the weight is 0.5 for each term). Besides, the trajectory reconstruction errors for x-y coordinates and the semantic label are considered equally (i.e., the weight is 0.25 for each term).

Since the probability $p^s_{j,k=\text{Target label}}$ belongs to the range of [0,1], the value of RE_2 is also in the range of [0,1] in equation 6.14. However, the values of RE_1 and QE are determined using the LSED metric, and their ranges may differ from [0,1]. Thus, we normalize RE_1 and QE to the range of [0,1] using min-max normalization. This ensures all terms in the AS equation have the same range.

6.2.2.2 Anomaly Threshold

To detect abnormal trajectories, a threshold is used. In this work, a new method is proposed for determining the anomaly threshold based on anomaly scores of trajectories in the training set. The anomaly threshold is defined as follows:

$$\mathrm{Thres} = \mu_{\mathrm{AS_t}} + \theta \times \sigma_{\mathrm{AS_t}}, \tag{6.17}$$

where $\mu_{\mathrm{AS_t}}$ and $\sigma_{\mathrm{AS_t}}$ are the mean and the standard deviation (SD) of the trajectory anomaly scores in the training set; θ is a parameter determined by the new WS metric. Note that since the training set only contains normal trajectories, Thres should be larger than $\mu_{\mathrm{AS_t}}$. Thus, θ should be positive.

The value of θ is chosen based on the proposed framework's performance on the validation set. The new factor that is the weighted sum of recall and precision, is used for finding the appropriate value of θ. The equation for WS is defined as

$$\text{WS} = \delta \times \text{Recall} + (1 - \delta) \times \text{Precision}, \tag{6.18}$$

where δ is a parameter that controls the trade-off between recall and precision in WS. If δ is larger than 0.5, the role of recall is more important than precision; otherwise, precision is more important. In our work, δ is set at 0.5.

Algorithm 2 Determine the anomaly threshold

```
Input: - Trained TENSO model
       - $X_{\text{Train}} = \{T_{\text{train}_1}, ..., T_{\text{train}_{N_\text{T}}}\}$
       - $X_{\text{Val}} = \{T_{\text{val}_1}, ..., T_{\text{val}_{N_\text{V}}}\}$
       - Values $\theta_{\text{min}}$, $\theta_{\text{max}}$
Output: The chosen anomaly threshold: $\text{Thres}_{\text{chosen}}$
for $i \leftarrow 1$ to $N_\text{T}$ do
    Determine $\text{RE}_1, \text{RE}_2, \text{QE}_{\text{SOM}}$ using TENSO with $T_{\text{train}_i}$
    Calculate $AS_{T_{\text{train}_i}}$ by equation 6.16
end for
for $i \leftarrow 1$ to $N_\text{V}$ do
    Determine $\text{RE}_1, \text{RE}_2, \text{QE}_{\text{SOM}}$ using TENSO with $T_{\text{val}_i}$
    Calculate $AS_{T_{\text{val}_i}}$ by equation 6.16
end for
$\mu_{\text{AS}_\text{t}} \leftarrow \text{mean}\{\text{AS}_{T_{\text{train}_i}}\}_{i=\{1,...,N_\text{T}\}}$
$\sigma_{\text{AS}_\text{t}} \leftarrow \text{SD}\{\text{AS}_{T_{\text{train}_i}}\}_{i=\{1,...,N_\text{T}\}}$
for $\theta \leftarrow \theta_{\text{min}}$ to $\theta_{\text{max}}$ do
    $\text{Thres} \leftarrow \mu_{\text{AS}_\text{t}} + \theta \times \sigma_{\text{AS}_\text{t}}$
    for $i \leftarrow 1$ to $N_\text{V}$ do
        if $\text{AS}_{T_{\text{val}_i}} \leq \text{Thres}$ then
            $T_{\text{val}_i} \leftarrow$ Normal trajectory
        else
            $T_{\text{val}_i} \leftarrow$ Abnormal trajectory
        end if
    end for
    Determine the number of True Positive (TP) samples in $X_{\text{Val}}$
    Determine the number of False Positive (FP) samples in $X_{\text{Val}}$
    Calculate recall and precision using TP, FP
    Calculate WS by equation 6.18
end for
$\theta_{\text{chosen}} \leftarrow \underset{\theta}{\text{argmax}}\ \text{WS}$
$\text{Thres}_{\text{chosen}} \leftarrow \mu_{\text{AS}_\text{t}} + \theta_{\text{chosen}} \times \sigma_{\text{AS}_\text{t}}$
```

With an anomaly detection framework, it is expected that both recall and precision of

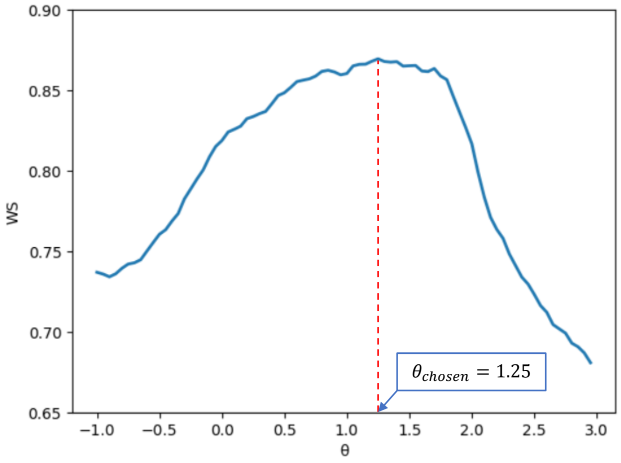


Figure 6.3: Choosing the value of θ

the framework achieve high values. In this case, the WS of recall and precision also obtains a high value. This only can be achieved if the anomaly threshold is selected appropriately. If a small value of threshold is chosen, the framework tends to detect trajectories as anomalies. Thus, recall obtains a high value, whereas precision is small. In contrast, if the anomaly threshold is large, a trajectory is marked normal more easily. Therefore, recall is small and precision is high. In both cases of threshold, the WS of recall and precision is small. From these aspects, an appropriate threshold is selected that the framework achieves a maximum value for WS. In this work, θ is chosen according to the maximum value of WS on the validation set. It is expected that this value also yields a good performance with the test set. A detailed description for choosing θ and the anomaly threshold is in **Algorithm 2**. Figure 6.3 depicts the selected value for θ as 1.25, corresponding to the maximum WS value for the validation set in the MIT Badge dataset. The chosen value of θ is larger than 0, which is appropriate considering our previous statement about θ.

6.2.3 Result Analysis

6.2.3.1 Baselines

In this work, we compare the proposed method with four baselines.

- **Hierarchical clustering**. In this baseline, the hierarchical clustering algorithm was first applied to find the normal trajectory clusters in the dataset [53]. Then, a new trajectory can be detected as an anomaly based on its relationship with the found clusters.

- **DBSCAN**. The work in [80] detected anomalous trajectories using DBSCAN. In this work, a distance metric was proposed to measure the distance of trajectories, and the Epsilon parameter of DBSCAN was determined using a new DCVI metric. To find normal trajectory clusters in the dataset, DBSCAN was used. In detecting anomalous trajectory, if a trajectory did not belong to any clusters, it was marked as an anomaly.

- **LSTM-AE**. In [39], an autoencoder-based framework for detecting anomalies was proposed. The authors applied two autoencoders (CNN-AE and LSTM-AE) to learn the latent spaces from input sequences. We compare the proposed framework with the LSTM-AE-based framework in this baseline.

- **LSTM-VAE**. The authors in [81] proposed an anomaly detection framework with multi-sensor measurement values for maritime components. Their model contained a LSTM-based variational autoencoder (LSTM-VAE). The architecture of VAE constrained the latent space of input data to a standard normal distribution. Besides, they used LSTM network in VAE to capture the temporal dependencies in the input sequences. In detecting anomalies, the reconstructed sequences from VAE was used to find the anomaly score. They determined a anomaly threshold based on the mean and standard deviation of anomaly scores for normal samples in the validation set.

- **Transformer**. The study in [64] introduced an Transformer-based anomaly detection framework for ECG signals. In this work, a deep learning model contains an embedding layer and a standard Transformer encoder to learn latent representations of original signals for detecting anomalies. Besides, an anomaly threshold was also determined based on predicted errors of normal signals in the training set.

6.2.3.2 Implementation Details

The proposed model is implemented in Python 3.9.5 using the Tensorflow Keras library. To train the model, the Adam optimizer is used with a learning rate of 0.001. The number of training epochs is 40, and the batch size is 128. The model's hyper-parameters are selected based on the framework's performance on the validation set. Besides, parameters γ_1 and γ_2 in the total loss function are chosen based on the learning curves of components in this function. These experiments for selecting γ_1, γ_2 and the model hyper-parameters will be discussed in subsections 6.2.3.5 and 6.2.3.6, respectively. The set of selected parameters is listed in Table 6.2.

Table 6.2: List of parameters

Parameters		Value	Studied in experiments
Input feature learning	Coordinates projecting dimension	64	✓
	Semantic projecting dimension	32	✓
Transformer encoder	D-model	128	✓
	Num of heads	4	✓
	Num of layers	4	✓
Decoder	Hidden dimension of decoder	64	✗
SOM	Map size of SOM	10x10	✓
	$T_{\max}$	10	✗
	$T_{\min}$	0.1	✗
Total loss function	γ_1	0.01	✓
	γ_2	0.001	✓
	Learning rate	0.001	✗
	Epochs	40	✗
	Batch size	128	✗

In the training phase, the best model is chosen at the epoch with the minimum validation loss. This means that choosing the model is based on the best result from learning trajectory representations and their clusters in latent space. With a given hyper-parameter set, selecting the best model in the training phase is independent of downstream tasks.

In the evaluation phase, the proposed framework uses both the MIT Badge and the sCREEN datasets. Besides, this work is compared with five baselines: hierarchical clustering-based, DBSCAN-based, LSTM-AE-based, LSTM-VAE-based, and Transformer methods. In the LSTM-AE-based method, since a mechanism for determining the anomaly threshold was not provided, we used the mechanism from the LSTM-VAE-based method to select an appropriate threshold.

Besides, two variants of the TENSO model are evaluated in this subsection. In particular, the first variant learns a latent space of normal trajectories using the Transformer encoder. Then, the SOM layer is used to learn normal trajectory representation clusters in the latent space. In this variant, the decoder is dropped, and trajectories are not reconstructed from the latent space. The variant is called TE-SOM. In the anomaly detection phase, the TE-SOM-based framework only uses the distance between the trajectory representation and its nearest cluster in the latent space.

In contrast, the second variant based on the Transformer encoder and a decoder (TE-Decoder) only learns normal trajectory representations. The SOM for learning normal trajectory representation groups is dropped in TE-Decoder. To detect anomalies, the TE-

Decoder-based framework only uses trajectory reconstruction errors. The results of all methods for each anomaly type are presented below.

6.2.3.3 Detecting pricing group as anomaly

Since the occurrence frequency and movement behaviors in worker groups are different in the MIT Badge dataset, a hypothesis is given to assign abnormal and normal labels to trajectories in this dataset [33]. Specifically, if the occurrence frequency of a group is much less than other groups, it can be considered an anomaly group. In the MIT Badge dataset, the pricing group accounts for only about 18% of the total, and the configuration group is approximately 72%. Therefore, workers' trajectories in the pricing and configuration groups are assigned as abnormal and normal samples, respectively. Table 6.3 shows the results of all methods when detecting pricing group as anomaly in the MIT Badge dataset. We can see that TENSO achieves 89.64% in terms of f1-score and outperforms all baselines.

In particular, the TENSO-based framework is significantly better than clustering-based methods (i.e., about 19% better than hierarchical clustering and 6% better than DBSCAN). With the clustering-based baselines, anomaly detection was only based on the relationship between the test sample and clusters in the dataset. In other words, they did not discover internal characteristics of trajectories. In contrast, TENSO jointly learns the correlation between points within each trajectory and trajectory representation clusters in the latent space to detect anomalies. Therefore, the proposed framework discovers more helpful information about trajectories, and improves accuracy in detecting anomalies. Note that the DBSCAN-based method in [86] used a time window of 10 minutes to collect data for each trajectory. In this work, to detect anomalies earlier, a shorter time window of only 2 minutes is used. Since the trajectory length is smaller, determining whether the trajectory is normal or abnormal becomes more difficult, and leads to lower performance.

Besides, our method detects anomalies more effectively than the existing deep learning-based methods. In particular, TENSO outperforms LSTM-AE, LSTM-VAE and Transformer at 10.55%, 12.31% and 9.83%, respectively, in terms of f1-score. These baselines focused on learning a latent space that captures the internal characteristics of trajectories. Anomalies were detected using reconstructed trajectories through the models. However, the three baselines did not discover the clusters of trajectory representations in the latent space for anomaly detection, which is performed in the TENSO model. This explains that the TENSO-based framework achieves better performance than the baselines.

Table 6.3: Results for detecting pricing group as anomaly in MIT Badge dataset.

Method	Recall	Precision	f1-score
Hierarchical clustering	0.7368	0.6708	0.7009
DBSCAN	0.8563	0.8149	0.835
LSTM-AE	0.8798	0.7183	0.7909
LSTM-VAE	0.7695	0.7772	0.7733
Transformer	0.7924	0.8039	0.7981
TE-SOM	0.9275	0.7595	0.8351
TE-Decoder	0.8066	0.8744	0.8392
TENSO	0.9009	0.892	**0.8964**

The TE-SOM and TE-Decoder variants achieve similar results in terms of f1-score (i.e., about 84%). In other aspects, TE-SOM obtains the highest recall at 92.75 % while only achieving approximately 76% for precision. In contrast, TE-Decoder outperforms TE-SOM in precision by about 11%. Thus, combining the two variants in TENSO helps achieve the best anomaly detection performance where both recall and precision are high.

6.2.3.4 Detecting synthesized anomalies

Three anomaly types (i.e., rare location visiting anomaly, wandering anomaly and route anomaly) are injected into both datasets for the evaluation. We also change the τ parameter to control the abnormality level of synthesized trajectories. In particular, τ is set to 0.5, 0.75 and 1 according to the time for anomaly part in each trajectory being 1, 1.5 and 2 minutes, respectively.

Tables 6.4 and 6.5 present the results of all methods when detecting three synthesized anomaly types in the MIT Badge and sCREEN datasets, respectively. The proposed framework achieves the best performance in these experiments compared with all the baselines. In particular, with the MIT Badget dataset, the TENSO-based framework detects all rare location visiting and wandering anomalies for three τ settings and achieves an f1-score of 94.83%. With route anomalies, the f1-score is achieved at 87.07%, 94.49% and 94.71% according to τ be 0.5, 0.75 and 1, respectively. With the sCREEN dataset, the proposed framework also achieves results similar to the MIT Badge dataset. All rare location visiting anomalies are identified, and the f1-score is approximately 96% in this case. The results for detecting route and wandering anomalies are also high, with f1-score at about 91% and 93%, respectively, for τ of 0.5. When τ increases to 0.75 and 1, f1-score is about 95% for both anomaly types. In both datasets, as can be seen that when τ increases, the anomaly

Table 6.4: Results for detecting synthesized anomalies in the MIT Badge dataset

Synthesized anomalies	τ (%)	0.5			0.75			1		
	Method	Recall	Precision	F1-score	Recall	Precision	F1-score	Recall	Precision	F1-score
Rare location visiting anomaly	Hierarchical clustering	0.9715	0.6892	0.8064	1	0.6954	0.8203	1	0.6954	0.8203
	DBSCAN	0.6579	0.7718	0.7104	1	0.8371	0.9113	1	0.8371	0.9113
	LSTM-AE	0.9988	0.7432	0.8522	1	0.7434	0.8528	1	0.7434	0.8528
	LSTM-AE	0.9628	0.8136	0.882	0.9895	0.8177	0.8954	0.9994	0.8192	0.9004
	Transformer	0.9988	0.8378	0.9112	0.9994	0.8379	0.9116	1	0.838	0.9119
	TE-SOM	1	0.7729	0.872	1	0.7729	0.872	1	0.7729	0.872
	TE-Decoder	1	0.8961	0.9452	1	0.8961	0.9452	1	0.8961	0.9452
	TENSO	1	0.9017	**0.9483**	1	0.9017	**0.9483**	1	0.9017	**0.9483**
Route anomaly	Hierarchical clustering	0.5713	0.566	0.5686	0.938	0.6817	0.7896	0.9418	0.6825	0.7915
	DBSCAN	0.544	0.7366	0.6258	0.9789	0.8342	0.9008	0.9957	0.8365	0.9092
	LSTM-AE	0.8854	0.7195	0.7939	0.9882	0.7412	0.8471	0.995	0.7425	0.8504
	LSTM-AE	0.8841	0.8003	0.8402	0.9281	0.808	0.8639	0.9622	0.8135	0.8816
	Transformer	0.8742	0.8189	0.8456	0.9368	0.8289	0.8796	0.9851	0.836	0.9044
	TE-SOM	0.9039	0.7547	0.8227	0.964	0.7665	0.854	0.9962	0.7723	0.8701
	TE-Decoder	0.8296	0.8774	0.8529	0.9368	0.8899	0.9128	0.9857	0.8948	0.9381
	TENSO	0.855	0.8869	**0.8707**	0.9932	0.9011	**0.9449**	0.9975	0.9014	**0.9471**
Wandering anomaly	Hierarchical clustering	0.6406	0.5939	0.6164	0.7429	0.6291	0.6813	0.7745	0.6387	0.7001
	DBSCAN	0.7534	0.7948	0.7735	0.9994	0.8371	0.911	1	0.8371	0.9113
	LSTM-AE	0.9994	0.7433	0.8525	1	0.7434	0.8528	1	0.7434	0.8528
	LSTM-AE	0.8618	0.7962	0.8277	0.9449	0.8107	0.8727	0.995	0.8186	0.8982
	Transformer	0.9473	0.8305	0.8851	0.9752	0.8346	0.8994	0.9777	0.8349	0.9007
	TE-SOM	0.9993	0.7728	0.8717	1	0.7729	0.872	1	0.7729	0.872
	TE-Decoder	1	0.8961	0.9452	1	0.8961	0.9452	1	0.8961	0.9452
	TENSO	1	0.9017	**0.9483**	1	0.9017	**0.9483**	1	0.9017	**0.9483**

detection accuracy of all methods is also larger for all anomaly types. Since abnormality in trajectories is higher when τ increases, they are detected more easily. Thus, the anomaly detection results are higher.

In the clustering-based baselines, when $\tau = 0.5$, the anomaly detection performance is much lower than the deep learning-based methods in both datasets. For example, at $\tau = 0.5$, the TENSO-based framework outperforms the hierarchical clustering-based method by about 30% for route and wandering anomalies in the MIT Badge dataset, and for all anomaly types in the sCREEN dataset. This can be explained that the clustering-based detection methods' performance was affected by the distance metric of trajectories. In these baselines, the distance metrics (i.e., the longest common subsequence (LCSS) in the hierarchical clustering-based method and the extension of LCSS (LCSS_IS) in the DBSCAN-based method) were used. LCSS and LCSS_IS aim at finding the largest number of similar points between two trajectories, ignoring unmatched points. In other words, these metrics tend to find normality rather than abnormality [86]. Thus, if the anomaly part in synthesized trajectories is small, anomaly detection performance of clustering-based methods is low.

Table 6.5: Results for detecting synthesized anomalies in the sCREEN dataset

Synthesized anomalies	τ (%)	0.5			0.75			1		
	Method	Recall	Precision	F1-score	Recall	Precision	F1-score	Recall	Precision	F1-score
Rare location visiting anomaly	Hierarchical clustering	0.9677	0.5176	0.6745	1	0.5258	0.6892	1	0.5258	0.6892
	DBSCAN	0.7948	0.739	0.7659	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	1	0.7942	0.8853	1	0.7942	0.8853	1	0.7942	0.8853
	LSTM-AE	1	0.8019	0.8901	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	1	0.871	0.931	1	0.871	0.931	1	0.871	0.931
	TE-SOM	0.9625	0.819	0.885	0.9916	0.8234	0.8997	0.998	0.8243	0.9029
	TE-Decoder	1	0.9093	0.9525	1	0.9093	0.9525	1	0.9093	0.9525
	TENSO	1	0.9214	**0.9591**	1	0.9214	**0.9591**	1	0.9214	**0.9591**
Route anomaly	Hierarchical clustering	0.9375	0.5096	0.6603	0.9977	0.5252	0.6882	1	0.5258	0.6869
	DBSCAN	0.6772	0.707	0.6918	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	0.9008	0.7766	0.8342	0.9011	0.7767	0.8343	0.9034	0.7771	0.8356
	LSTM-AE	0.7654	0.756	0.7654	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	0.936	0.8634	0.8982	0.9997	0.871	0.9309	1	0.871	0.931
	TE-SOM	0.9974	0.8242	0.9026	0.9974	0.8242	0.9026	0.9974	0.8242	0.9026
	TE-Decoder	0.8406	0.894	0.8665	0.9594	0.9059	0.9318	0.9948	0.9089	0.9499
	TENSO	0.9201	0.9151	**0.9177**	0.989	0.9206	**0.9536**	0.9925	0.9208	**0.9553**
Wandering anomaly	Hierarchical clustering	0.8671	0.4901	0.6262	0.9392	0.5101	0.6611	0.9524	0.5136	0.6673
	DBSCAN	0.8075	0.7421	0.7734	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	0.9302	0.7821	0.8498	1	0.7942	0.8853	1	0.7942	0.8853
	LSTM-AE	0.9484	0.7934	0.864	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	0.9945	0.8704	0.9283	0.998	0.8708	0.93	1	0.871	0.931
	TE-SOM	0.9107	0.8107	0.8578	0.9326	0.8143	0.8694	0.9974	0.8242	0.9026
	TE-Decoder	0.9971	0.9091	**0.9511**	1	0.9093	**0.9525**	1	0.9093	0.9525
	TENSO	0.944	0.9171	0.9304	0.9674	0.9189	0.9426	0.9948	0.921	**0.9565**

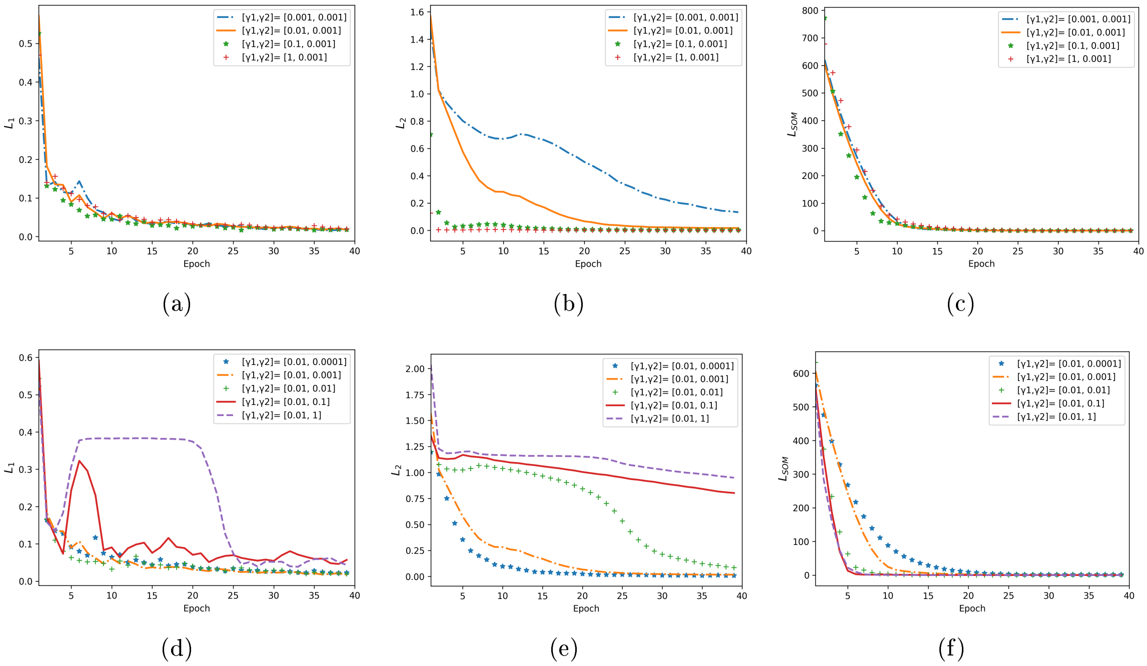
Moreover, as stated previously, the clustering-based baselines only use the relationship between the test trajectory and the clusters of normal trajectories in the dataset to detect anomalies. They did not explore the internal characteristics of the trajectories. Therefore, their performance is still lower than the proposed framework even when abnormality in synthesized trajectories increases (e.g., $\tau = \{0.75, 1\}$).

In the existing deep learning-based methods, the best model (i.e., Transformer) achieves 91.19% and 93.1% in terms of f1-score for detecting rare location visit anomaly in the MIT Badge and sCREEN datasets, respectively. The results are still lower than the TENSO-based framework by 3.64% with the MIT Badge dataset and by 2.55% with the sCREEN dataset. The reason for the performance difference is that the baselines were solely based on trajectory reconstruction errors to detect anomalies. In contrast, the proposed framework uses both trajectory reconstruction error and quantization error on SOM. Tables 6.4 and 6.5 also show that the Transformer-based method is better than LSTM-AE-based and LSTM-VAE-based methods on both datasets in terms of f1-score. This can be explained that the Transformer architecture is more effective than LSTM architecture in learning trajectory representations for detecting anomalies.

With the two variants of the proposed model, TE-Decoder achieves high precision while TE-SOM maintains high recall for all anomaly types. Therefore, the combination of TE-Decoder and TE-SOM in TENSO significantly improves detection results with both datasets, as shown in Tables 6.4 and 6.5. Besides, as can be seen that TE-Decoder also has an attractive performance in f1-score for all anomalies. This variant is even better than the TENSO-based method when detecting the wandering anomaly in the sCREEN dataset: about 2% and 1% higher for f1-score when $\tau = 0.5$ and 0.75, respectively. This means that learning internal characteristics and sequential information of trajectories based on the Transformer encoder and decoder plays an important role in detecting anomalies. In addition, as can be seen from Tables 6.4 and 6.5, the TE-Decoder variant also outperforms Transformer-based method, although both models also use the Transformer architecture. This is because an effective way to select the anomaly threshold is provided in the TE-Decoder. In contrast, in the Transformer-based method, a fixed value is used in the formulation of determining the anomaly threshold, which affects the performance of detecting anomalies.

From Tables 6.4 and 6.5, it can be shown that the deep learning-based methods can detect easily the rare location visiting anomaly. This is possibly explained that this anomaly type contains rare locations, which the deep learning models do not see in the training phase. Thus, the rare location visiting anomaly is not reconstructed well by the models, and they are detected more easily than two remaining anomaly types. For example, the lowest recall still achieves 96.28% for LSTM-VAE in the MIT Badge dataset and 96.25% for TE-SOM in the sCREEN dataset. With the TENSO-based framework, all anomalies of this type are detected.

Note that if anomaly detection ability of a method does not change over different anomaly types, the f1-score of the method is the same for the anomaly types. Since the anomaly detection ability is maintained, the number of true positive samples does not change over the anomaly types. Besides, since normal samples are maintained when evaluating the methods for all anomaly types, the number of false positive samples is also kept. Thus, the f1-score, determined using the number of true positive and false positive samples, is the same for the anomaly types. For example, in the MIT Badge dataset, since all samples of rare location visiting and wandering anomalies are correctly detected by the TENSO-based framework, the results in terms of the f1-score for this method are the same for the two anomaly types.



(a) (b) (c)

(d) (e) (f)

Figure 6.4: Component loss functions when changing γ_1 and γ_2: (a) L_1 when changing γ_1. (b) L_2 when changing γ_1. (c) L_{SOM} when changing γ_1. (d) L_1 when changing γ_2. (e) L_2 when changing γ_2. (f) L_{SOM} when changing γ_2.

6.2.3.5 Sensitivity of Parameters in Loss Functions

This subsection discusses the γ_1 and γ_2 parameters that controls the trade-off between components in the total loss function of the proposed model. The experiments are performed with the MIT Badge dataset.

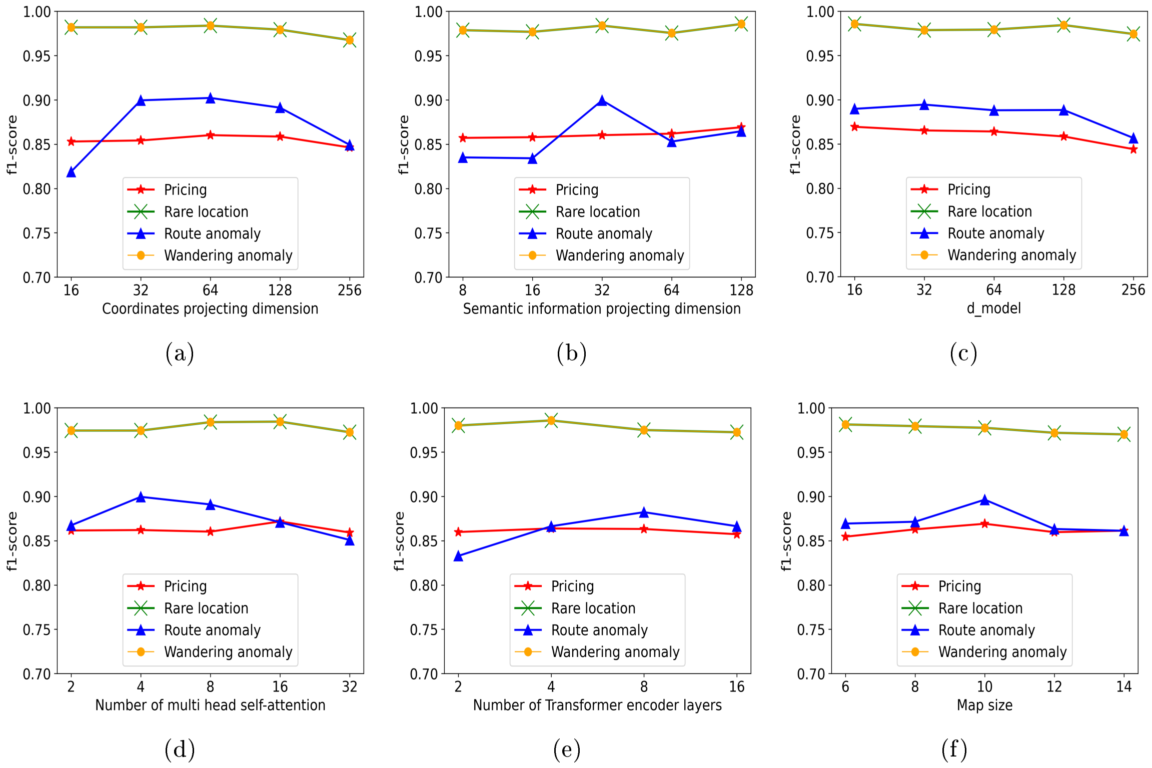
The influence of these parameters on L_1, L_2, and L_{SOM} (i.e., the loss function for reconstructing x-y coordinates, the loss function for reconstructing semantic information, and the loss of the SOM, respectively) is shown in Figure 6.4. As can be seen in Figure 6.4 that L_1 has the smallest amplitude. The amplitude of L_{SOM} is the largest and is much larger than the two remaining components. Note that the weight of L_1 in the total loss function is 1. Thus, to ensure the trade-off between components in the total loss function, the weight of L_2 (i.e., γ_1) should be smaller than 1, and the weight of L_{SOM} (i.e., γ_2) should be smaller than γ_1.

To evaluate the effect of γ_1 on the learning curves, γ_2 is kept to a value smaller than 1 (e.g., 0.001), and we change γ_1 to different values $\{0.001, 0.01, 0.1, 1\}$. The top row of Figure 6.4 shows the evolution of three learning curves in the total loss function when changing γ_1. In Figure 6.4(a), the learning curves of L_1 according to values of γ_1 are only slightly different. This can be explained that since the amplitude of L_2 is not much larger than L_1, the effect of changing the weight of L_2 on the learning curve of L_1 is unclear. Besides, as can be seen in Figure 6.4(c) that optimization of the SOM loss is ensured with each value of γ_1. As previously shown, the amplitude of L_2 is much smaller than L_{SOM}. Thus, the optimizing process of L_{SOM} is also not influenced by γ_1. In contrast, since γ_1 is the weight of L_2, it directly affects the learning curve of L_2. In particular, the optimizing process is slow when the weight is too small (e.g., when $\gamma_1 = 0.001$). When the weight is near or equal to 1, the learning curve is not smooth, and quickly drops to the optimal value in the first few training epochs. Therefore, to ensure optimization of all components in the total loss function, an appropriate γ_1 value of 0.01 is chosen.

Next, the evolution of learning curves when changing γ_2 is shown in the bottom row of Figure 6.4. Note that in the experiments, $\gamma_1 = 0.01$. As can be seen that the L_1 and L_2 reconstruction losses are clearly affected when γ_2 is changed. That is possibly explained that the value of L_{SOM} is much higher than L_1 and L_2. Thus, when the weight of L_{SOM} changes, it influences these losses remarkably. In particular, with a value for γ_2 of 0.1 or 1, the optimizing process of L_1 is improper, as seen in Figure 6.4(d). It decreases quickly after the first few epochs and then increases significantly in the subsequent epochs before reaching the optimal value. When γ_2 is reduced to one of the $\{0.0001, 0.001, 0.01\}$ values, the optimization of this function is ensured. Similarly, the L_2 reconstruction loss function is not optimized correctly with γ_2 from the set $\{0.01, 0.1, 1\}$ as seen in Figure 6.4(e). With the SOM loss, γ_2 has more influence than γ_1 on the learning curve. The reason is that γ_2 is the weight of L_{SOM}, directly affecting this learning curve. However, optimization of L_{SOM} is still properly ensured with all the values of γ_2. From the above analysis, we select 0.001 as the γ_2 value, which is appropriate for the three loss functions.

6.2.3.6 Effects of hyper-parameters

A discussion about the effects of model hyper-parameters on the proposed framework's performance is given in this subsection. Specifically, Figure 6.5 shows the effects of six hyper-parameters: the x-y coordinate projecting dimension, semantic information projecting dimension, d_model (i.e., the input dimension of the Transformer encoder), the number for multi-head self-attention, the number of layers in the Transformer encoder, and the map size of the SOM layer. The experiments are performed on the validation set of the MIT Badge dataset, and both hypothesized and synthesized anomalies are used for evaluation. Note



(a) (b) (c)

(d) (e) (f)

Figure 6.5: Influence of parameters on the performance in MIT Badge dataset: (a) Coordinates projecting dimension. (b) Semantic projecting dimension. (c) D-model. (d) Number of multi head self-attention. (e) Number of Transformer encoder layers. (f) Map size.

that the synthesized anomalies are generated with $\tau = 1$ in the experiments. The proposed framework's performance is presented in terms of the f1-score.

First, as can be seen in Figures 6.5(a) and 6.5(b) that the route anomaly detection ability by the framework is clearly affected when changing semantic information and coordinates projecting dimensions of trajectory points. In contrast, with the remaining anomalies, the framework performance is stable over the different values for these hyper-parameters. Thus, selecting semantic information and coordinate projecting dimensions is based on the performance from detecting the route anomaly. In particular, the semantic information and the coordinate projecting dimensions are set to 32 and 64, respectively. The proposed framework achieves the best performance on the route anomaly at these values.

Next, Figures 6.5(c), 6.5(d) and 6.5(e) present the effects on framework's performance from hyper-parameters on the Transformer encoder (i.e., the d_model, the number for multi-head self-attention, and the number of Transformer encoder layers, respectively). In Figure 6.5(c), framework performance is stable over all anomaly types when the d_model is not greater than 128. If d_model = 256, the performance decreases. One possible reason is that when the d_model is too large, the model becomes more complex and causes the overfitting problem. In the case, the value of the d_model is set at 128. From Figures 6.5(d) and 6.5(e), the number for multi-head self-attention and the number of Transformer encoder layers are set at 4. These values are chosen to ensure the trade-off between effectiveness and computational cost.

Finally, the effect of map size on the SOM is shown in Figure 6.5(f). As can be seen in Figure 6.5(f), the framework's performance when changing the map size is also stable for hypothesized anomalies (i.e., pricing group's trajectories) and the two synthesized anomaly types (i.e., rare location and wandering anomalies). Therefore, choosing the map size is also based on the proposed framework's performance when detecting route anomaly. From Figure 6.5(f), the map size is chosen at 10 according to the best performance on the route anomaly.

6.3 Chapter Summary

This work proposed an anomalous indoor human trajectory detection framework using a deep learning model. Our proposed model was based on the Transformer encoder and SOM, which jointly learned normal trajectories' representations and their clusters in the latent space. In particular, the Transformer encoder with a self-attention mechanism was used to learn the correlations between points within each trajectory and its sequence information. A decoder reconstructed the input trajectory from its latent representation. In addition, the SOM layer was used to learn clusters of normal trajectory representations in the latent space. In detecting anomalous trajectories, the anomaly score of the test trajectory was determined by using the trained model. The trajectory's anomaly score contained the trajectory reconstruction errors from the latent space and the quantization error on the SOM. If the anomaly score exceeded a set threshold, the test trajectory was detected as an anomaly. In this work, we also proposed a novel metric called WS, the weighted sum of recall and precision for determining the anomaly threshold. Especially an appropriate anomaly threshold was selected according to the maximum value for WS with the validation set. Our framework was estimated using two real trajectory datasets: MIT Badge and sCREEN. The results depicted that the proposed framework achieved attractive performance, and outperformed existing methods in anomaly trajectory detection.

In the future, we will extend the proposed model to learn more information about trajectories (e.g., speed, direction) to improve anomaly detection performance. Besides, the model will be developed to detect the anomaly trajectories even if the trajectories are not complete.

Chapter 7

Concluding Remarks

7.1 Summary of the Contribution

This dissertation studies problems in anomalous human trajectory detection in indoor spaces. First, we studied abnormal indoor human trajectory types. Then, a density-based framework for detecting anomalous trajectories was proposed. Next, a trajectory clustering-based anomalous indoor human trajectory detection framework was introduced to improve the performance of detecting anomalies. Finally, a deep learning model was developed to learn the characteristics of trajectories for detecting abnormalities.

Anomalous trajectory types in indoor human movements are crucial for evaluating proposed anomalous detection frameworks. Thus, we studied abnormal human trajectory types in indoor spaces, which can occur in real-world environments. Then, these anomaly types were synthesized and injected into datasets for evaluation. In this chapter, we also introduced the density-based anomalous trajectory detection framework. This framework proposed a new method for determining distance and density thresholds. A trajectory was detected as an anomaly if its density was smaller than a density threshold. The results show that our framework detected anomalies efficiently and outperformed the EMM-based anomalous detection framework.

In the next chapter, the clustering-based anomalous trajectory detection framework was studied. In particular, a new distance metric called LCSS_IS, extending from LCSS, was first proposed to improve the performance of measuring the distance between indoor human trajectories. Then, to improve the performance of clustering trajectories in datasets, a new metric (DCVI) was introduced for choosing the Eps parameter in DBSCAN. To detect abnormal trajectories, clusters in datasets were first discovered using DBSCAN. Then, a trajectory was marked as an anomaly if it did not belong to any clusters in datasets. The experiments showed that the proposed framework was better than baselines (i.e., EMM-based, density-based, hierarchical clustering-based and spectral clustering-based). In addition, the proposed distance metric was also more efficient than existing distance metrics (i.e., Euclidean, EDR, LCSS, ISTSM).

To learn more trajectory characteristics for detecting anomalous trajectories, a deep learning model called TENSO was proposed. In TENSO, a Transformer encoder was used to learn a latent space of trajectories, which captured the internal characteristics of trajectories. From the latent space, a SOM layer was used to learn clusters of trajectory representations. To train the model, a total loss function of reconstruction and SOM losses was proposed. In addition, to detect anomalous trajectories, a new method for determining anomaly thresholds was proposed. A trajectory was detected as an anomaly if its anomaly score was larger than the anomaly threshold. The performance of the TENSO-based framework was evaluated and compared with traditional methods (i.e., hierarchical clustering-based, DBSCAN-based) and models (i.e., LSTM-AE-based, LSTM-VAE-based and variants of TENSO). The results showed that our model achieved the best performance.

In summary, the main contributions of this work are as follows:

- We studied characteristics of abnormal human trajectory types in indoor spaces, which can occur in urgent situations. Then, these anomaly types were generated and injected into datasets for evaluating abnormal trajectory detection frameworks.

- We proposed a density-based anomalous trajectory detection framework. In this method, anomaly thresholds are determined based on distribution of trajectories in datasets.

- Trajectory clustering-based anomalous indoor trajectory detection is proposed to improve anomaly detection ability. A new metric called LCSS_IS is proposed for indoor human trajectories. Besides, a new cluster validity index called DCVI is proposed to choose the Eps parameter of DBSCAN. In detecting anomalies, a trajectory is detected based its relationship and clusters in datasets.

- A deep learning model was proposed to learn trajectory characteristics for detecting abnormal trajectories. In our model, both internal characteristics of trajectories and clusters of trajectory representations in latent space was discovered. To learn the internal characteristics of trajectories, the Transformer encoder was used. Then, a

SOM layer was used to learn clusters of trajectory representations from the latent space of trajectories.

- For the problem of choosing a threshold to detect anomalies, we developed a new metric called WS, which is a weighted sum of recall and precision. To find an appropriate anomaly threshold value, normal trajectory scores in the training set were first determined. Then, the framework's performance was evaluated on a validation set using WS. An anomaly threshold was chosen according to the maximum value of WS.

- Various experiments have been conducted to evaluate the proposed anomaly detection frameworks. Specifically, the proposed frameworks were evaluated using two real trajectory datasets: MIT Badge and sCREEN. The results show that our frameworks efficiently detected anomalous human trajectory types in indoor spaces. Specifically, the TENSO-based framework achieved an attractive performance and outperformed all evaluated frameworks.

7.2 Future Works

For studying anomalous human trajectory types in indoor spaces, we discover characteristics of three anomaly types: rare location visiting anomaly, wandering and route anomalies. These anomaly types can occur in some urgent situations in workplaces, such as fire, violent attacks, theft, and terrorism. In the future, we want to study more anomalous human trajectory types in indoor spaces. For example, one person can suffer from an accident, and they stop at the location for a longer period.

In the clustering-based anomalous detection framework, there are still a few limitations. First, several features of trajectory data, such as speed and moving direction, were not considered for calculating the distance between trajectories. Second, the proposed method was not evaluated on datasets with complex floor plans, such as buildings with many floors. We plan to extend this framework to address the above limitations in a future study.

For learning characteristics using deep learning models, we also want to extend the proposed model to learn more information about trajectories (e.g., speed, direction) to improve anomaly detection performance. Besides, the model will be developed to detect the anomaly trajectories even if the trajectories are not complete.

Bibliography

[1] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, A. Cano, and J. C.-W. Lin, "A two-phase anomaly detection model for secure intelligent transportation ride-hailing trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4496–4506, 2020.

[2] Y. Wang, K. Qin, Y. Chen, and P. Zhao, "Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi gps data," *ISPRS International Journal of Geo-Information*, vol. 7, no. 1, p. 25, 2018.

[3] P. Han, A. L. Ellefsen, G. Li, F. T. Holmeset, and H. Zhang, "Fault detection with lstm-based variational autoencoder for maritime components," *IEEE Sensors Journal*, vol. 21, no. 19, pp. 21 903–21 912, 2021.

[4] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iboat: Isolation-based online anomalous trajectory detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 806–818, 2013.

[5] D. Chen, Y. Du, S. Xu, Y.-E. Sun, H. Huang, and G. Gao, "Online anomalous taxi trajectory detection based on multidimensional criteria," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[6] S. Qian, B. Cheng, J. Cao, G. Xue, Y. Zhu, J. Yu, M. Li, and T. Zhang, "Detecting taxi trajectory anomaly based on spatio-temporal relations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6883–6894, 2021.

[7] L. Xu, Y. Luo, Q. Yu, X. Zhang, W. Zhang, and Z. Lu, "Anomalous taxi trajectory detection using popular routes in different traffic periods," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, 2023.

[8] S. Wu, B. E. Moore, and M. Shah, “Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes,” in *2010 IEEE computer society conference on computer vision and pattern recognition.* IEEE, 2010, pp. 2054–2060.

[9] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby, “Detecting anomalies in people’s trajectories using spectral graph analysis,” *Computer Vision and Image Understanding*, vol. 115, no. 8, pp. 1099–1111, 2011.

[10] T. Zhang, S. Zhao, and J. Chen, “Ship trajectory outlier detection service system based on collaborative computing,” in *2018 IEEE World Congress on Services (SERVICES).* IEEE, 2018, pp. 15–16.

[11] Y. Huang and Q. Zhang, “Identification of anomaly behavior of ships based on knn and lof combination algorithm,” in *AIP Conference Proceedings*, vol. 2073, no. 1. AIP Publishing LLC, 2019, p. 020090.

[12] J. Weng, G. Li, and Y. Zhao, “Detection of abnormal ship trajectory based on the complex polygon,” *The Journal of Navigation*, vol. 75, no. 4, pp. 966–983, 2022.

[13] R. Zhen, Y. Jin, Q. Hu, Z. Shao, and N. Nikitakos, “Maritime anomaly detection within coastal waters based on vessel trajectory clustering and naïve bayes classifier,” *The Journal of Navigation*, vol. 70, no. 3, pp. 648–670, 2017.

[14] J. Weng, G. Li, and Y. Zhao, “Detection of abnormal ship trajectory based on the complex polygon,” *The Journal of Navigation*, vol. 75, no. 4, pp. 966–983, 2022.

[15] R. Fujimaki, T. Yairi, and K. Machida, “An approach to spacecraft anomaly detection problem using kernel feature space,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 401–410.

[16] J. Oehling and D. J. Barry, “Using machine learning methods in airline flight data monitoring to generate new operational safety knowledge from existing data,” *Safety science*, vol. 114, pp. 89–104, 2019.

[17] K. Sheridan, T. G. Puranik, E. Mangortey, O. J. Pinon-Fischer, M. Kirby, and D. N. Mavris, “An application of dbscan clustering for flight anomaly detection during the approach phase,” in *AIAA Scitech 2020 Forum*, 2020, p. 1851.

[18] K. Qin, Q. Wang, B. Lu, H. Sun, and P. Shu, “Flight anomaly detection via a deep hybrid model,” *Aerospace*, vol. 9, no. 6, p. 329, 2022.

[19] M. Memarzadeh, B. Matthews, and I. Avrekh, “Unsupervised anomaly detection in flight data using convolutional variational auto-encoder,” *Aerospace*, vol. 7, no. 8, p. 115, 2020.

[20] M. Paolanti, D. Liciotti, R. Pietrini, A. Mancini, and E. Frontoni, “Modelling and forecasting customer navigation in intelligent retail environments,” *Journal of Intelligent & Robotic Systems*, vol. 91, no. 2, pp. 165–180, 2018.

[21] P. Wang, J. Yang, and J. Zhang, “Location prediction for indoor spaces based on trajectory similarity,” in *2021 4th International Conference on Data Science and Information Technology*, 2021, pp. 402–407.

[22] ——, “A spatial-temporal-semantic method for location prediction in indoor spaces,” *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[23] K. Tran, D. Q. Phung, B. Adams, and S. Venkatesh, “Indoor location prediction using multiple wireless received signal strengths.” in *AusDM*, 2008, pp. 187–192.

[24] K. A. Nguyen and Z. Luo, “Reliable indoor location prediction using conformal prediction,” *Annals of Mathematics and Artificial Intelligence*, vol. 74, pp. 133–153, 2015.

[25] P. Wang, J. Yang, and J. Zhang, “Location prediction for indoor spaces based on trajectory similarity,” in *2021 4th International Conference on Data Science and Information Technology*, 2021, pp. 402–407.

[26] C. Koehler, “Indoor location prediction through modeling of human spatiotemporal behavior,” Ph.D. dissertation, Carnegie Mellon University, 2019.

[27] J.-G. Lee, J. Han, and X. Li, “Trajectory outlier detection: A partition-and-detect framework,” in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 140–149.

[28] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, “ibat: detecting anomalous taxi trajectories from gps traces,” in *Proceedings of the 13th international conference on Ubiquitous computing*, 2011, pp. 99–108.

[29] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008.

[30] N. B. Ghrab, E. Fendri, and M. Hammami, "Abnormal events detection based on trajectory clustering," in *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV).* IEEE, 2016, pp. 301–306.

[31] Z. Zhu, D. Yao, J. Huang, H. Li, and J. Bi, "Sub-trajectory-and trajectory-neighbor-based outlier detection over trajectory streams," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2018, pp. 551–563.

[32] P. Banerjee, P. Yawalkar, and S. Ranu, "Mantra: a scalable approach to mining temporally anomalous sub-trajectories," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1415–1424.

[33] M. SZEKÉR, "Spatio-temporal outlier detection in streaming trajectory data," 2014.

[34] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *2020 IEEE 36th International Conference on Data Engineering (ICDE).* IEEE, 2020, pp. 949–960.

[35] P.-R. Lei, "A framework for anomaly detection in maritime trajectory behavior," *Knowledge and Information Systems.*

[36] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering.* IEEE, 2002, pp. 673–684.

[37] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[38] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 491–502.

[39] L. Song, R. Wang, D. Xiao, X. Han, Y. Cai, and C. Shi, "Anomalous trajectory detection using recurrent neural network," in *Advanced Data Mining and Applications:*

14th International Conference, ADMA 2018, Nanjing, China, November 16–18, 2018, Proceedings 14. Springer, 2018, pp. 263–277.

[40] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *2018 19th IEEE international conference on mobile data management (MDM)*. IEEE, 2018, pp. 125–134.

[41] Y. Zhang, N. Ning, P. Zhou, and B. Wu, "Ut-atd: Universal transformer for anomalous trajectory detection by embedding trajectory information." in *DMSVIVA*, 2021, pp. 70–77.

[42] G. Bouritsas, S. Daveas, A. Danelakis, and S. C. Thomopoulos, "Automated real-time anomaly detection in human trajectories using sequence to sequence networks," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–8.

[43] Y. Cheng, B. Wu, L. Song, and C. Shi, "Spatial-temporal recurrent neural network for anomalous trajectories detection," in *Advanced Data Mining and Applications: 15th International Conference, ADMA 2019, Dalian, China, November 21–23, 2019, Proceedings 15*. Springer, 2019, pp. 565–578.

[44] C. Ma, Z. Miao, M. Li, S. Song, and M.-H. Yang, "Detecting anomalous trajectories via recurrent neural networks," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 370–382.

[45] R. Jiao, J. Bai, X. Liu, T. Sato, X. Yuan, Q. A. Chen, and Q. Zhu, "Learning representation for anomaly detection of vehicle trajectories," *arXiv preprint arXiv:2303.05000*, 2023.

[46] Z. Wang, G. Yuan, H. Pei, Y. Zhang, and X. Liu, "Unsupervised learning trajectory anomaly detection algorithm based on deep representation," *International Journal of Distributed Sensor Networks*, vol. 16, no. 12, p. 1550147720971504, 2020.

[47] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[48] M. M. Van Hulle, "Self-organizing maps." *Handbook of Natural Computing*, vol. 1, pp. 585–622, 2012.

[49] S. Kaski and K. Lagus, "Comparing self-organizing maps," in *International conference on artificial neural networks.* Springer, 1996, pp. 809–814.

[50] T. Kohonen, "Essentials of the self-organizing map," *Neural networks*, vol. 37, pp. 52–65, 2013.

[51] D. Miljković, "Brief review of self-organizing maps," in *2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO).* IEEE, 2017, pp. 1061–1066.

[52] D. O. Olguín, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland, "Sensible organizations: Technology and methodology for automatically measuring organizational behavior," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 43–55, 2008.

[53] Y. Tao, A. Both, R. I. Silveira, K. Buchin, S. Sijben, R. S. Purves, P. Laube, D. Peng, K. Toohey, and M. Duckham, "A comparative analysis of trajectory similarity measures," *GIScience & Remote Sensing*, vol. 58, no. 5, pp. 643–669, 2021.

[54] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd international conference on knowledge discovery and data mining*, 1994, pp. 359–370.

[55] J. Zhu, D. Cheng, W. Zhang, C. Song, J. Chen, and T. Pei, "A new approach to measuring the similarity of indoor semantic trajectories," *ISPRS International Journal of Geo-Information*, vol. 10, no. 2, p. 90, 2021.

[56] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Time-dependent popular routes based trajectory outlier detection," in *International Conference on Web Information Systems Engineering.* Springer, 2015, pp. 16–30.

[57] M. A. Saleem, W. Nawaz, Y.-K. Lee, and S. Lee, "Road segment partitioning towards anomalous trajectory detection for surveillance applications," in *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI).* IEEE, 2013, pp. 610–617.

[58] S. Saitta, B. Raphael, and I. F. Smith, "A comprehensive validity index for clustering," *Intelligent Data Analysis*, vol. 12, no. 6, pp. 529–548, 2008.

[59] Q. Zhao and P. Fränti, “Wb-index: A sum-of-squares based index for cluster validity,” *Data & Knowledge Engineering*, vol. 92, pp. 77–89, 2014.

[60] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[61] N. Rahmah and I. S. Sitanggang, “Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra,” in *IOP conference series: earth and environmental science*, vol. 31, no. 1. IOP Publishing, 2016, p. 012012.

[62] K. Giri and T. K. Biswas, “Determining optimal epsilon (eps) on dbscan using empty circles,” in *International Conference on Artificial Intelligence and Sustainable Engineering: Select Proceedings of AISE 2020, Volume 1.* Springer Nature, 2020, p. 265.

[63] N. Di Mauro and S. Ferilli, “Unsupervised lstms-based learning for anomaly detection in highway traffic data,” in *Foundations of Intelligent Systems: 24th International Symposium, ISMIS 2018, Limassol, Cyprus, October 29–31, 2018, Proceedings 24.* Springer, 2018, pp. 281–290.

[64] A. Alamr and A. Artoli, “Unsupervised transformer-based anomaly detection in ecg signals,” *Algorithms*, vol. 16, no. 3, p. 152, 2023.

[65] D. S. Hirschberg, “Algorithms for the longest common subsequence problem,” *Journal of the ACM (JACM)*, vol. 24, no. 4, pp. 664–675, 1977.

[66] L. Bergroth, H. Hakonen, and T. Raita, “A survey of longest common subsequence algorithms,” in *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000.* IEEE, 2000, pp. 39–48.

[67] V. Chvatal and D. Sankoff, “Longest common subsequences of two random sequences,” *Journal of Applied Probability*, vol. 12, no. 2, pp. 306–315, 1975.

[68] M. Paterson and V. Dančík, “Longest common subsequences,” in *International symposium on mathematical foundations of computer science.* Springer, 1994, pp. 127–142.

[69] M. Hahsler, M. Piekenbrock, and D. Doran, “dbscan: Fast density-based clustering with r,” *Journal of Statistical Software*, vol. 91, pp. 1–30, 2019.

[70] T. Ali, S. Asghar, and N. A. Sajid, “Critical analysis of dbscan variations,” in *2010 international conference on information and emerging technologies.* IEEE, 2010, pp. 1–6.

[71] B. Borah and D. K. Bhattacharyya, “An improved sampling-based dbscan for large spatial databases,” in *International conference on intelligent sensing and information processing, 2004. proceedings of.* IEEE, 2004, pp. 92–96.

[72] Z. Shi and L. S. Pun-Cheng, “Spatiotemporal data clustering: a survey of methods,” *ISPRS international journal of geo-information*, vol. 8, no. 3, p. 112, 2019.

[73] L. Yang and M. Worboys, “Generation of navigation graphs for indoor space,” *International Journal of Geographical Information Science*, vol. 29, no. 10, pp. 1737–1756, 2015.

[74] P. Flikweert, R. Peters, L. Díaz-Vilariño, R. Voûte, and B. Staats, “Automatic extraction of a navigation graph intended for indoorgml from an indoor point cloud,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 271–278, 2019.

[75] P. Flikweert, “Automatic extraction of an indoorgml navigation graph from an indoor point cloud,” 2019.

[76] L. Niu and Y. Song, “A schema for extraction of indoor pedestrian navigation grid network from floor plans,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 41, pp. 325–330, 2016.

[77] C. S. Jensen, H. Lu, and B. Yang, “Graph model based indoor tracking,” in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware.* IEEE, 2009, pp. 122–131.

[78] E. W. Dijkstra *et al.*, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[79] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee, “Top-eye: Top-k evolving trajectory outlier detection,” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 1733–1736.

[80] K. Usman and M. Ramdhani, “Comparison of classical interpolation methods and compressive sensing for missing data reconstruction,” in *2019 IEEE International Conference on Signals and Systems (ICSigSys).* IEEE, 2019, pp. 29–33.

[81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[82] T. Kohonen, *Self-organizing maps.* Springer Science & Business Media, 2012, vol. 30.

[83] S. D. Fabiyi, “A review of unsupervised artificial neural networks with applications,” *International Journal of Computer Applications*, vol. 181, no. 40, pp. 22–26, 2019.

[84] S. Patro and K. K. Sahu, “Normalization: A preprocessing stage,” *arXiv preprint arXiv:1503.06462*, 2015.

[85] E. A. Uriarte and F. D. Martín, “Topology preservation in som,” *International journal of applied mathematics and computer sciences*, vol. 1, no. 1, pp. 19–22, 2005.

[86] D. T. Lan and S. Yoon, “Trajectory clustering-based anomaly detection in indoor human movement,” *Sensors*, vol. 23, no. 6, p. 3318, 2023.

Author's Publications

International Journals

1. **Doi Thi Lan**, Seokhoon Yoon, "Trajectory Clustering-Based Anomaly Detection in Indoor Human Movement," *Sensors*, vol. 23, no. 6, p.3318, Mar. 2023.

2. **Doi Thi Lan**, Seokhoon Yoon, "Anomalous Indoor Human Trajectory Detection based on the Transformer Encoder and Self-Organizing Map," *IEEE Access*, vol. 11, pp. 131848-131865, Nov. 2023.

3. Quan T. Ngo, **Doi Thi Lan**, Seokhoon Yoon, Woo−Sung Jung, Taehyun Yoon, and Dae−Young Kim "Companion Mobility to Assist in Future Human Location Prediction," *IEEE Access*, vol. 10, pp. 68111-68125, Jun. 2022.

Domestic Journals

1. **Doi Thi Lan**, Seokhoon Yoon, "Detecting Anomalous Trajectories of Workers using Density Method," *International Journal of Internet, Broadcasting and Communication*, vol. 14, no. 2, pp.109-118, May. 2022.

2. **Doi Thi Lan**, Seokhoon Yoon, "Detecting Abnormal Human Movements Based on Variational Autoencoder," *International Journal of Internet, Broadcasting and Communication*, vol. 15, no. 3, pp.94-102, Aug. 2023.

3. Dat Van Anh Duong, **Doi Thi Lan**, Seokhoon Yoon, "An Abnormal Worker Movement Detection System Based on Data Stream Processing and Hierarchical Clustering," *International Journal of Internet, Broadcasting and Communication*, vol. 14, no. 4, pp.88-95, Nov. 2022.

International Conferences

1. **Doi Thi Lan**, Seokhoon Yoon, "Anomalous Human Trajectory Detection Using Clustering Methods," in *Proc. The 14th International Conference on ICT Convergence (ICTC)*, pp.365-367, Oct. 2023, Jeju, Korea. Online Proceeding on Conference website (https://ictc.org/program_proceeding).

Domestic Conferences

1. **Doi Thi Lan**, Seokhoon Yoon, "Human Trajectory Anomaly Detection in Working Sites," in *Proc. The Korean Institute of Communications and Information Sciences Summer Conference 2022*, pp.1262-1263, Jun. 2022, Jeju, Korea.

2. **Doi Thi Lan**, Seokhoon Yoon, "Detecting Anomalous Movement of Human using Clustering Method," in *Proc. The Korean Institute of Communications and Information Sciences Summer Conference 2023*, pp.1411-1412, Jun. 2023, Jeju, Korea.

3. Quan T. Ngo, Dat Van Anh Duong, **Doi Thi Lan**, Seokhoon Yoon, "Estimating Workers' Locations in Industrial Sites," in *Proc. The Korean Institute of Communications and Information Sciences Fall Conference 2021*, pp.239-240, Nov. 2021, Yeosu, Korea.