Online Learning and Sequential Anomaly Detection in Trajectories

Rikard Laxhammar and Göran Falkman, *Member, IEEE,*

Abstract—Detection of anomalous trajectories is an important problem in the surveillance domain. Various algorithms based on learning of normal trajectory patterns have been proposed for this problem. Yet, these algorithms typically suffer from one or more limitations: They are not designed for sequential analysis of incomplete trajectories or online learning based on an incrementally updated training set. Moreover, they typically involve tuning of many parameters, including ad-hoc anomaly thresholds, and may therefore suffer from overfitting and poorly-calibrated alarm rates. In this article, we propose and investigate the Sequential Hausdorff Nearest-Neighbor Conformal Anomaly Detector (SHNN-CAD) for online learning and sequential anomaly detection in trajectories. This is a parameter-light algorithm that offers a well-founded approach to the calibration of the anomaly threshold. The discords algorithm, originally proposed by Keogh *et al.*, is another parameter-light anomaly detection algorithm that has previously been shown to have good classification performance on a wide range of time-series datasets, including trajectory data. We implement and investigate the performance of SHNN-CAD and the discords algorithm on four different labeled trajectory datasets. The results show that SHNN-CAD achieves competitive classification performance with minimum parameter tuning during unsupervised online learning and sequential anomaly detection in trajectories.

Index Terms—Anomaly detection, trajectory data, online learning, conformal prediction

1 Introduction

Anomalous behaviour may indicate important objects and events in a wide variety of domains. One such domain is surveillance where there is a clear trend towards more and more advanced sensor systems producing huge amounts of geo-spatial *trajectory data* from moving objects, such as people, vehicles, vessels and animals. In video surveillance of ground activities, for example, anomalous trajectories may be indicative of illegal and adverse activity related to personal assault, robbery, burglary, infrastructural sabotage etc. Timely detection of these relatively infrequent events, which is critical for enabling pro-active measures, requires careful analysis of all moving objects at all times; this is typically a great challenge to human analysts due to information overload, fatigue and inattention. Thus, there is a need for automated trajectory analysis.

This article is concerned with algorithms for automated detection of anomalous trajectories. Various algorithms based on learning of normal trajectory patterns from historical data have previously been proposed for this problem [1]. However, these algorithms typically suffer from one or more limitations: They are primarily designed for offline anomaly detection in databases and do not support *sequential anomaly detection* in incomplete trajectories or *online learning* based on an incrementally updated training set. Moreover, they typically involve tuning of many parameters, including ad-hoc anomaly thresholds, and may therefore suffer from overfitting and poorly-calibrated alarm rates.

In our previous work [2], we introduced the *Conformal Anomaly Detector* (CAD), which is a general algorithm for anomaly detection that is based on the framework of Conformal prediction [3]. The main idea of CAD is to estimate the *p*-value for new data based on a specified *Non-Conformity Measure* (NCM) [3]. Intuitively, the *p*-value corresponds to the probability of observing data that appears to be at least as extreme as the data that was actually observed. If the *p*-value is below a predefined anomaly threshold, the new data is considered very unlikely and, therefore, classified as anomalous. A key property of CAD is that it offers a well-founded approach to the tuning of the anomaly threshold that guarantees that the alarm rate will be well-calibrated under relatively weak statistical assumptions. Apart from the anomaly threshold, the only design parameter of CAD is the specified NCM. We previously proposed a novel NCM based on the *Directed Hausdorff Distance* (DHD) [4] for sequential conformal anomaly detection in trajectories [2]. The intuition of DHD is that it measures the degree to which an object resembles some *part* of another object. This key property makes DHD well-suited for measuring the distance from an incomplete trajectory to other trajectories.

1.1 Main Contributions

This article deepens and expands on our previous results [2]. In particular, we:

- *R. Laxhammar is with the Saab AB, Järfälla 175 41, Sweden. E-mail: rikard.laxhammar@saabgroup.com.*
- *G. Falkman is with the University of Skövde, Skövde 541 28, Sweden. E-mail: goran.falkman@his.se.*

Manuscript received 14 Nov. 2012; revised 15 May 2013; accepted 22 Aug. 2013. Date of publication 12 Sep. 2013; date of current version 12 May 2014. Recommended for acceptance by F. Fleuret.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier 10.1109/TPAMI.2013.172

- Present an extensive discussion of limitations of previous algorithms for anomaly detection in general and the detection of anomalous trajectories in particular.
- Formalise CAD and the concepts of offline vs. online and unsupervised vs. semi-supervised conformal anomaly detection. This includes the statement and proof of a theorem regarding the property of well-calibrated alarm rate of CAD.
- Formalise the *Directed Hausdorff k-Nearest Neighbour Non-Conformity Measure* (DH-kNN NCM). This includes the statement and proof of a theorem regarding the monotonicity of the p-values during sequential update of an incomplete trajectory, which further implies that the alarm rate of CAD is well-calibrated during sequential anomaly detection in incomplete trajectories.
- Re-introduce our algorithm for sequential anomaly in trajectories [2] as the *Sequential Hausdorff Nearest Neighbour Conformal Anomaly Detector* (SHNN-CAD). This includes a more detailed description of the algorithm, a discussion of its properties and implementation and an analysis of its time complexity.
- Extend the empirical investigation of DH-kNN NCM and SHNN-CAD by including new experiments on new trajectory datasets and comparing it with the discords approach [5].

1.2 Outline

The outline of the article is as follows: First we introduce the problem of anomaly detection and present an overview of previous work related to detection of anomalous trajectories (Section 2). Next, we identify and discuss some general limitations of previously proposed algorithms (Section 3). In Section 4, we first present some key results of conformal prediction [3] that underpin CAD. This is followed by the formalisation of CAD and discussions of its properties and operating modes. In Section 5, we are concerned with the problem of sequential anomaly detection in trajectories: We first introduce DHD [4] before formalising DH-kNN NCM. In the remainder of the section, we present SHNN-CAD, discuss its properties and implementation and analyse its time complexity. In Section 6, we implement DH-kNN NCM and SHNN-CAD and evaluate their performance on four trajectory datasets. For comparative purposes, we also implement and evaluate performance of the discords algorithm [5], which has previously been proposed for detecting anomalous trajectories [6]. Future work and generalisations are discussed in Section 7. Finally, we conclude the article in Section 8.

2 Background

2.1 Anomaly Detection

According to Chandola *et al*. [7], "anomalies are patterns in data that do not conform to a well-defined notion of normal behaviour". These patterns typically correspond to new, rare, unknown or otherwise extreme behaviour and may therefore be of particular interest. From an application owners point of view, *anomaly detection* may be considered as a binary classification problem where each *example*, also known as data point or object, either belongs to the *normal class* or the *abnormal class*. However, unlike traditional supervised classification, available training data is typically unlabelled or assumed to only include examples labelled normal [7]. That is, there are no labelled examples of the abnormal class available for training. When training data is labelled normal, anomaly detection algorithms typically focus on learning one-class classifiers [8], probability density functions [9] or other types of models, which are used for determining whether unlabelled test examples belong to the normal class or not; such approaches are referred to as *semi-supervised anomaly detection* [7]. When training data is unlabelled, there is usually no distinction between training and test data; the training set is assumed to include a small minority of abnormal examples, which are discriminated from the normal examples by using unsupervised learning techniques, such as clustering. Such approaches may be refereed to as *unsupervised anomaly detection* [7].

2.2 Detection of Anomalous Trajectories

A trajectory is modelled as a finite sequence of data points:

$$(\chi_1, t_1), \ldots, (\chi_L, t_L) : t_{i-1} < t_i, i = 2, \ldots, L, \tag{1}$$

where each $\chi_i \in \mathbb{R}^d$ is a multi-dimensional feature vector at time point t_i and where the total length in time of the trajectory is $t_L - t_1$ [1]. In the simple case, $\chi_i \in \mathbb{R}^2$ represents an object's (estimated) location in the two-dimensional spatial plane at time point t_i. Depending on the application, the feature space may be extended by, e.g., a third spatial dimension or the velocity.

2.2.1 Clustering-Based Approaches

A large amount of work related to anomaly detection in trajectories has been published in the video surveillance domain [1], [10], [11]. Most of these methods involve *trajectory clustering*, where cluster models corresponding to normal paths are learnt from historical trajectories. This process typically consists of three steps that are sometimes mixed: preprocessing of raw trajectories, clustering of preprocessed trajectories and modelling of trajectory clusters [1]. New trajectories are then typically classified as normal or abnormal based on the distance to the closest cluster, or the likelihood of the most probable cluster.

Two main techniques are used for preprocessing trajectories: *normalisation* and *dimensionality reduction* [1]. Normalisation techniques ensure that trajectories are of equal length, for example by extending short trajectories using zero padding [10] or by re-sampling trajectories. Dimensionality reduction techniques, such as vector quantisation, discrete Fourier transforms [12], parametrised spline models [13], wavelets, Hidden Markov Models (HMM) [14], principal component analysis and spectral methods [15], all map trajectories into a lower dimensional feature space that is computationally more manageable [1].

All clustering algorithms require that an appropriate *similarity measure*, also known as a distance measure, is defined. Euclidian distance (ED) (e.g., [16]) is perhaps the simplest and most intuitive similarity measure. Other similarity measures and modifications of ED have been proposed for relaxing alignment and length constraints,

such as Dynamic Time Warping (DTW) [1] and Longest Common Sub-Sequence (LCSS) [17]. Different algorithms have been proposed for creating and updating trajectory clusters based on a specified similarity measure [1]. Once trajectories have been clustered, appropriate cluster models, sometimes referred to as *path models* [1] or *motion patterns* [10], are defined. Two main types of path models have been adopted: The first considers a complete path, from the starting point to the endpoint. The second decomposes a path into smaller atomic parts called *subpaths* [1]. Complete path models are typically based on a *centroid* representation of the cluster, which corresponds to the "average" trajectory [1]. The centroid is sometimes complemented by an *envelope*, which captures the extension and variance of the trajectories [1]. Hu *et al*. [10] model each path as a chain of Gaussian distributions. Morris and Trivedi [18] proposed the modelling of each path by a HMM, where each hidden state is modelled by a Gaussian mixture model. Subpath models can be further defined in terms of probabilistic connections [1]. An example of a subpath structure was proposed by Piciarelli [19], where each subpath is modelled as a node in a tree-like structure augmented with probabilities for node transitions.

For complete trajectories, anomaly detection is typically carried out by first determining the path model that best explains the new trajectory, i.e., having the maximum likelihood or minimal distance to the new trajectory. The distance or likelihood is then compared to an anomaly threshold. Some algorithms, but far from all, support *sequential,* also known as *online* [1] or *incremental* [10], anomaly detection in *incomplete* trajectories. Morris and Trivedi [18] proposed an algorithm that monitors the likelihood for the part of the trajectory that is within a sliding window of a fixed size. If this likelihood drops below a specified threshold, the trajectory is classified as anomalous. Hu *et al*. [10] proposed an algorithm for sequential anomaly detection in incomplete trajectories. For each new data point, the algorithm first updates the most probable path model (motion pattern) given the part of the trajectory observed so far. The likelihood for the new point is then calculated relative the most probable path model and the point is classified as anomalous if its likelihood is below a specified threshold. If a series of points are classified as anomalous, the trajectory is classified as anomalous. Fu *et al*. [20] proposed an algorithm that incrementally determines the Maximum A Posterior (MAP) path model (trajectory cluster) as more data points are observed from an incomplete trajectory. The incomplete trajectory is classified as anomalous if it leaves the envelope of the current MAP path model, or if the velocity of the trajectory deviates from the velocity of the MAP path model.

2.2.2 Non-Clustering-Based Approaches

Other approaches to detecting anomalous trajectories have been proposed that do not involve clustering of trajectories as presented above (e.g., [21]). Piciarelli *et al*. [16] proposed a trajectory learning and anomaly detection algorithm based on a one-class Support Vector Machine (SVM), where each trajectory is represented by a fixed-dimensional feature vector of evenly sampled points from the raw trajectory. One of the main novelties of this algorithm is its ability to automatically detect and remove anomalies in the training data. Yankov *et al*. [6] proposed the application of *time-series discords* [5] for detecting anomalous trajectories in a database. Assuming preprocessed trajectories of equal length, the *k*th discord corresponds to the trajectory with the *k*th largest ED to its nearest neighbour trajectory in the database. Owens and Hunter [22] proposed an algorithm based on a Self-Organising Map (SOM) that is appropriate for learning and sequential anomaly detection in trajectories. Each data point from a trajectory is represented by a fixed-length feature vector encompassing the current location, velocity and acceleration together with information on the recent position. Lee *et al*. [23] proposed a *partition-and-detect* framework for detection of anomalous sub-trajectories in a trajectory database. A two-step anomaly (outlier) detection algorithm is proposed, which first partitions each trajectory into a number of line segments. Next, anomalous trajectory partitions, i.e., line segments, are detected according to a combination of distance and density-based analysis.

3 Limitations of Previous Algorithms

This section presents in more detail some of the main limitations of previous algorithms for anomalous trajectory detection, which motives the proposal of SHNN-CAD. In the first subsection, we discuss issues that are related to anomaly detection in general. The second subsection is more focused towards problems that are specific for the trajectory domain.

3.1 Anomaly Detection in General

3.1.1 Invalid Statistical Assumptions

Parameterised statistical models, such as the Gaussian distribution, are attractive because parameter estimation is rather straightforward and because there exist statistical tests that provide well-founded confidence for anomaly detection [7]. However, accuracy of estimated models, and robustness of anomaly detection, depend on whether the structural assumptions are valid. For example, fitting a Gaussian model to data generated from an approximately uniform distribution will result in overestimation and underestimation of the probability density at the centre (mean) and at the tails of the distribution, respectively [24]. In case of underestimation, examples of the normal class will be assigned a lower likelihood and, hence, are more likely to be classified as abnormal. In case of overestimation, the sensitivity to abnormal examples will decrease as there is an increased risk that subtle anomalies will be erroneously classified as normal. In many real world applications, such as traffic surveillance, where behaviour is governed by social systems and structures rather than by physical laws, the Gaussian assumption may indeed be questioned (see [25]).

3.1.2 Parameter-Laden Algorithms

Most anomaly detection algorithms require careful preprocessing and setting of multiple application specific parameters and detection thresholds in order to achieve (near) optimal performance; trajectory anomaly detection is no exception to this. In fact, Keogh *et al*. [26] argue that most data

mining algorithms are more or less *parameter-laden*, which is undesirable for several reasons. In an extensive empirical study, they showed that "in case of anomaly detection, parameter-laden algorithms are particularly vulnerable to overfitting" [26]. The risks of overfitting parameter-laden models in real world applications were also discussed by Hand [27], who showed empirically that "the marginal gain from complicated models is typically small compared to the predictive power of the simpler models". According to Markou and Sing [28], "an [anomaly] detection method should aim to minimise the number of parameters that are user set". Indeed, one may argue that use of few parameters suppresses bias towards particular types of anomalies and makes the method easier to implement for different applications.

3.1.3 Ad-Hoc Anomaly Thresholds

A key parameter in most, if not all, anomaly detection algorithms is the *anomaly threshold.* It regulates the balance between the *sensitivity* [29], i.e., the fraction of the abnormal examples that are successfully detected, and the *false alarm rate* [29], i.e., the fraction of normal examples that are erroneously detected as anomalous. Most algorithms define the threshold in terms of a distance, density or data likelihood [7]. These measures are typically not normalised and the procedures for setting the thresholds seem to be more or less ad-hoc. In particular, their interpretation are not very intuitive to, e.g., an operator of a surveillance system and it may be difficult to predict the actual anomaly detection rate on future data. In case of modern classification-based methods, such as one-class SVM, it has been argued that "while these approaches provide impressive computationally efficient solutions on real data, it is generally difficult to precisely relate tuning parameter choices to desired false alarm probability" [30]. The difficulty of tuning the anomaly threshold may result in high false alarm rates and low precision, which is critical to the usability of the system [31]. More specifically, a low precision will increase the workload of an operator of a surveillance system. This increases the risk that the anomaly detector is considered a nuisance and, hence, is ignored or turned off [32].

3.2 Anomalous Trajectory Detection

3.2.1 Offline Learning

With a few exceptions (notably [19]), previous algorithms (e.g., [6], [23]) are designed for *offline learning* in the sense that all training trajectories are assumed to be available from the outset; fixed model parameters and thresholds are estimated or tuned once based on a batch of training data and are then repeatedly used for anomaly detection. In contrast, algorithms designed for *online learning* incrementally updates the model parameters as each new training trajectory is observed. The need for online learning was highlighted by Piciarelli *et al*., who proposed an efficient online trajectory clustering algorithm [19]. It may be counter-argued that taking the system offline is not strictly necessary in case of an offline learning process that runs in parallel with the anomaly detector, where updated model parameters are introduced at specific time points. For example, some authors discuss regular batch-learning repetitions using an updated training set [18]. However, it is not clear how to determine the appropriate time point for such an update. On the one hand, minimising the learning delay, i.e., the time between successive model updates, may be desirable in order to maintain a more timely and accurate model. But on the other hand, computational complexity of offline learning algorithms, which is typically higher than for online algorithms, may impose restrictions on how small delay that can be achieved in a practical application.

3.2.2 Offline Anomaly Detection

Previously proposed algorithms are mainly designed for anomaly detection in databases and, hence, it is often explicitly or implicitly assumed that the *complete* trajectory (from start to end points) is available prior to classification. This is true for, e.g., algorithms based on dimensionality reduction and other normalisation techniques [12]–[15]. Moreover, most of the proposed trajectory similarity measures, including ED, DTW and LCSS (Section 2.2), are essentially designed for complete trajectories, even though some of them (e.g., LCSS) are more robust to missing data points than others. In surveillance applications where trajectories are sequentially updated in real-time, anomaly detection is delayed until the trajectory has terminated and, thus, the ability to react to impending events is limited.

In contrast, algorithms that are designed for *sequential anomaly detection* in *incomplete* trajectories (also known as trajectory streams [33]) enable timely detection of anomalies as they evolve. Examples of such algorithms include various point-based anomaly detectors where a low-dimensional feature model of the current and previous trajectory states is considered (e.g., [21], [22]). Yet, an obvious limitation of such algorithms is that they do not capture more long-term trajectory behaviour. A compromise between a point-based approach and a traditional model of complete trajectories is to consider a sub-trajectory model. Hu *et al*. [10] proposed an algorithm where each trajectory is divided into a number of fixed length and non-overlapping subsequences of data points. Each such subsequence is then represented as a high-dimensional feature vector. However, such methods typically involves more preprocessing, such as alignment and interpolation, and more parameters, such as size of the subsequence; it may not be obvious how the size of the subsequence should be chosen. Moreover, there will still be a delay in anomaly detection that is bounded by the length of the subsequence. Bu *et al*. [33] developed an algorithm for online learning and sequential anomaly detection in a single continuous trajectory stream. However, this algorithm has at least four free parameters and it is not clear how different values affect the expected anomaly detection rate.

4 Conformal Anomaly Detection

In the previous section, we discussed a number of general limitations of previous anomaly detection algorithms that need to be addressed. In this section, we introduce and discuss conformal anomaly detection, which is a novel approach to anomaly detection that addresses these issues.

Conformal anomaly detection is an extension of the framework of conformal prediction. Therefore, we will first give a brief overview of conformal prediction and its main results.

4.1 Preliminaries: Conformal Prediction

Conformal prediction is a technique for providing valid measures of *confidence* for individual predictions made by machine learning algorithms [34]. Assume a training set $z_1, \ldots, z_l$ where each example $z_i = (x_i, y_i) : i = 1, \ldots, l$ consist of features $x_i \in \mathbf{X}$ and label $y_i \in \mathbf{Y}$. Given a new example with observed features x_{l+1} and predefined *significance level* $\epsilon \in (0, 1)$, a conformal predictor outputs a *prediction set* $\Gamma^{\epsilon}_{l+1} \subseteq \mathbf{Y}$ for the unknown label y_{l+1}. The prediction set is *valid* at the specified significance level in the sense that the probability of an *error*, i.e., the event that the prediction set does *not* include the true label, is guaranteed to be less or equal to ϵ:

$$Pr\left(y_{l+1} \notin \Gamma^{\epsilon}_{l+1}\right) \leq \epsilon, \tag{2}$$

under the relatively weak statistical assumption that $z_1, \ldots, z_{l+1}$ are *Independent and Identically Distributed* (IID) [3]. For example, setting $\epsilon = 0.05$, we know that the probability that a prediction set includes the true label is at least 95%, regardless of the underlying probability distribution, which may be unknown. That is, we have 95% confidence in the prediction set.

The basic idea of conformal prediction is to estimate the *p-value* p_y for each possible label $y \in \mathbf{Y}$ for the new example and exclude from the prediction set those labels where $p_y < \epsilon$ [34]. In order to estimate *p*-values, the concept of a *Non-Conformity Measure* (NCM) is introduced, which measures how "different" an example is relative a set of examples. Formally, an NCM is a real-valued function $A(B, z)$ that returns a *nonconformity score* α measuring how different an example z is from the examples in the *bag* (also known as multi-set) B [35]. By calculating the nonconformity score for each example $z_i : i = 1, \ldots, n$ relative the rest of the examples $B_i = \{z_j : j = 1, \ldots, n, j \neq i\}$, the *p*-value p_i for z_i can be estimated as the ratio of the nonconformity scores $\alpha_1, \ldots, \alpha_n$ that are at least as large as α_i:

$$p_i = \frac{\left|\{j = 1, \ldots, n : \alpha_j \geq \alpha_i\}\right|}{n}. \tag{3}$$

Given a new example with observed features x_{l+1} and hypothetical label Y, it is expected that any $Y \neq y_{l+1}$, i.e., any label other than the true label, would result in the corresponding α_{l+1} being relatively large compared to $\alpha_1, \ldots, \alpha_l$. This would imply that p_Y is small and that the corresponding example constitutes an outlier. Hence, the prediction set is formed by estimating p_Y for each $Y \in \mathbf{Y}$ and including in the prediction set those Y for which $p_Y \geq \epsilon$:

$$\Gamma^{\epsilon}_{l+1} = \{Y : Y \in \mathbf{Y}, p_Y \geq \epsilon\}. \tag{4}$$

The validity property of conformal predictors (2) can be further strengthened by slightly modifying (3):

$$p_i = \frac{\left|\{j : \alpha_j > \alpha_i\}\right| + \tau_i \left|\{j : \alpha_j = \alpha_i\}\right|}{n}, \tag{5}$$

where $\tau_i \in [0, 1]$ is an independent and random sample from the uniform distribution [3]. The result of estimating *smoothed p*-values according to (5) is that the probability of error is now *exactly* equal to ϵ [3]:

$$Pr\left(y_{l+1} \notin \Gamma^{\epsilon}_{l+1}\right) = \epsilon. \tag{6}$$

One of the most important properties of conformal predictors based on (5) is that successive prediction errors are *independent* during online learning and prediction [34], i.e., when the training set is updated with the true label for the current test example before predicting the label of the next example: Assume a sequence of test examples $z_{l+1}, \ldots, z_N$. For each $n = l + 1, \ldots, N$, the online conformal predictor outputs a prediction set for y_n based on the observed features x_n and the *updated* training set $z_1, \ldots, z_{n-1}$. Let $\mathrm{err}^{\epsilon}_n \in \{0, 1\}$ be an indicator variable for the event that $y_n \notin \Gamma^{\epsilon}_n$. Then, the sequence of random variables $\mathrm{err}^{\epsilon}_{l+1}, \ldots, \mathrm{err}^{\epsilon}_N$ are independent and will each take value 1 with probability ϵ [34]. This implies that the empirical error rate of the online conformal predictor is guaranteed to be *well-calibrated*:

$$\frac{\left|\{i = l + 1, \ldots, N : \mathrm{err}^{\epsilon}_i = 1\}\right|}{N - l} \approx \epsilon. \tag{7}$$

A conformal predictor will produce valid prediction sets that satisfy (2) using any real-valued function $A(B, z)$ as the NCM. However, the prediction sets will only be small and, thus, informative if an appropriate NCM is chosen. If available, domain knowledge regarding, e.g., the structure of the data generating process should be exploited when defining nonconformity measures. Nevertheless, several general NCMs based on standard machine learning algorithms have been proposed that do not require any particular domain knowledge, e.g., the k-nearest neighbours algorithm, (kernel) ridge regression, SVM and neural networks [3].

4.2 The Conformal Anomaly Detector

The framework of conformal prediction was originally developed for supervised learning and prediction applications [3]. However, theoretical results from conformal prediction are also highly relevant for the problem of anomaly detection. The key observation is that the estimated *p*-value (3) is a general and useful measure of anomaly. Based on the idea of estimating *p*-values using a NCM, we define the Conformal Anomaly Detector (CAD) as follows:

Given a training set $(z_1, \ldots, z_l)$, a NCM A and predefined anomaly threshold ϵ, CAD (Algorithm 1) classifies an unlabelled test example z_{l+1} based on its *p*-value p_{l+1}. That is, the nonconformity score is calculated for each example relative the rest using using A (lines 1–3 of Algorithm 1) and p_{l+1} estimated for z_{l+1} according to (3) (line 4). If $p_{l+1} < \epsilon$, then z_{l+1} is classified as a *conformal anomaly*: $\mathrm{Anom}^{\epsilon}_{l+1} = 1$. Otherwise, z_{l+1} is classified as *normal*: $\mathrm{Anom}^{\epsilon}_{l+1} = 0$.

The definition of a conformal anomaly is consistent with the statistical definition of an *outlier* given by Hawkins [36]. That is, a conformal anomaly corresponds to a test example z_{l+1} that deviates so much from $(z_1, \ldots, z_l)$ as to arouse suspicion that it was *not* generated according to the same mechanism as $(z_1, \ldots, z_l)$. Analogously to statistical hypothesis testing, ϵ corresponds to an upper bound of the probability of erroneously rejecting the null hypothesis that z_{l+1} and $(z_1, \ldots, z_l)$ are independent and random samples

Algorithm 1: Conformal Anomaly Detector (CAD)

Input: NCM A, anomaly threshold ϵ, training set $(z_1, \ldots, z_l)$ and test example z_{l+1}.
Output: Indicator variable $\text{Anom}_{l+1}^{\epsilon} \in \{0, 1\}$.

```
for i ← 1 to l + 1 do
    α_i ← A({z_1, ..., z_{l+1}} \ z_i, z_i)
end for
p_{l+1} ← |{i=1,...,l+1 : α_i ≥ α_{l+1}}| / (l+1)
if p_{l+1} < ε then
    Anom^ε_{l+1} ← 1
else
    Anom^ε_{l+1} ← 0
end if
```

from the same probability distribution [35]. Hence, the following proposition regarding the probability of a conformal anomaly can be stated:

Proposition 1. *Assume that $z_1, \ldots, z_{l+1}$ are IID. For any choice of NCM A, the specified anomaly threshold ϵ corresponds to an upper bound of the probability of the event that z_{l+1} is classified as a conformal anomaly by CAD:*

$$Pr\left(\text{Anom}_{l+1}^{\epsilon} = 1\right) \leq \epsilon. \tag{8}$$

The larger the value of ϵ, the higher the expected rate of detected conformal anomalies.

From an application perspective, there are at least three different explanations for a conformal anomaly. Firstly, it may correspond to a relatively rare or previously unseen example generated from the same probability distribution as $(z_1, \ldots, z_l)$. Secondly and thirdly, it may be a "true" anomaly in the sense that it was not generated according to the same probability distribution as $(z_1, \ldots, z_l)$; either z_{l+1} is a true novelty, or $(z_1, \ldots, z_l)$ itself is in fact not IID. A non-IID training set may be explained by incomplete or biased data collection. In a public video surveillance application, e.g., observed behaviour during early morning or late afternoon may appear anomalous if the training set is based on data recorded during a limited time of the day, e.g., 10 a.m. to 2 p.m. Another possible reason for a non-IID training set is that the underlying probability distribution has actually changed. In maritime surveillance, for example, new vessel trajectories may arise as a result of new traffic regulations. If the rate of detected conformal anomalies suddenly starts to deteriorate from ϵ in an online application, there may be reasons to suspect that the underlying probability distribution has changed recently.

4.3 Offline vs. Online Conformal Anomaly Detection

Assume an initial training set $(z_1, \ldots, z_l)$ and a sequence of new test examples $z_{l+1}, \ldots, z_N$. The *offline* CAD classifies each $z_n{:}n = l+1, \ldots, N$ based on the initial training set $(z_1, \ldots, z_l)$. In contrast, the *online* CAD classifies each z_n based on the *updated* training set $(z_1, \ldots, z_{n-1})$. That is, z_{n-1} is appended to the training set $(z_1, \ldots, z_{n-2})$ before the classification of z_n. If $z_1, \ldots, z_n$ are IID, we know from Proposition 1 that the *unconditional* probability that z_n is classified as a conformal anomaly is bounded by ϵ. Nevertheless, the actual rate of detected anomalies for the offline CAD may be significantly higher than ϵ since the sequence of random variables $\text{Anom}_{l+1}^{\epsilon}, \ldots, \text{Anom}_{N}^{\epsilon}$ are *not* independent. In the online framework, though, the rate of detected anomalies will, with high probability, be less or approximately equal to ϵ. This non-asymptotic property, which we refer to as *well-calibrated alarm rate*, is formalised by Theorem 2:

Theorem 2. *Assume that the sequence of examples $z_1, \ldots, z_N$ are IID and that $N-l$ is large. For any choice of NCM A and anomaly threshold ϵ, the rate of detected conformal anomalies will, with very high probability, be less or approximately equal to ϵ for the online CAD:*

$$\frac{\left|\text{Anom}_n^{\epsilon} = 1{:}n = l+1, \ldots, N\right|}{N-l} \lessapprox \epsilon.$$

Proof. Let us temporarily assume that the deterministic p-value (3) for each test example $z_n{:}n = l+1, \ldots, N$ is replaced by the corresponding smoothed p-value (5). This would imply that $Pr\left(\text{Anom}_n^{\epsilon} = 1\right) = \epsilon$, i.e., that the probability of the event that z_n is classified as a conformal anomaly is exactly ϵ. Since the smoothed p-values $p_{l+1}, \ldots, p_N$ are independent in the online framework [3, Theorem 8.2], the random variables $\text{Anom}_{l+1}^{\epsilon}, \ldots, \text{Anom}_{N}^{\epsilon}$ are also independent. Therefore, if $N-l$ is large, it follows from the law of large numbers that the rate of detected conformal anomalies will, with very high probability, be approximately equal to ϵ. Now, since the probability of a conformal anomaly is less or equal to ϵ in the case of deterministic p-values (Proposition 1), the rate of detected conformal anomalies for the online CAD will, with very high probability, be less or approximately equal to ϵ. □

4.4 Unsupervised and Semi-Supervised Conformal Anomaly Detection

The conformal anomaly detection framework is essentially based on an unsupervised learning paradigm where there is no explicit notion of normal and abnormal classes. Nevertheless, in anomaly detection applications, each example is often considered to belong to either a normal or an abnormal class (Section 2.1). Similar to Eskin [37], let us therefore assume that examples of the normal and abnormal classes are generated according to the unknown probability distributions P_{Normal} and $P_{Abnormal}$, respectively, and that the unknown prior probability of the abnormal class is equal to λ. The generative distribution for *unlabelled* examples can then be modelled as a *mixture model*:

$$P_{Data} = (1-\lambda)\, P_{Normal} + \lambda P_{Abnormal}. \tag{9}$$

Depending on whether the training set is assumed to be unlabelled or labelled normal, CAD can be considered to operate in an unsupervised or semi-supervised anomaly detection mode, respectively (Section 2.1).

The unsupervised online CAD is perhaps the most interesting from both a theoretical and practical point of view; it does not require any label feedback and the overall rate of detected conformal anomalies is well-calibrated, regardless of P_{Normal}, $P_{Abnormal}$ and λ. The only assumption is that $(z_1, \ldots, z_N)$ constitute an IID sample from P_{Data}, which does not seem to be an impractical or otherwise unrealistic assumption. The semi-supervised online CAD also

has a similar notion of well-calibrated alarm rate: Assume that $\left(z'_{l+1}, \ldots, z'_Q : Q \leq N\right)$ corresponds to the subsequence of normal test examples of the full sequence $(z_{l+1}, \ldots, z_N)$ and that each $z'_n : n = l+2, \ldots, Q$ is classified based on the updated training set $\left(z_1, \ldots, z_l, z'_{l+1}, \ldots, z'_{n-1}\right)$. Further, assume that $Q-l$ is large and that $\left(z_1, \ldots, z_l, z'_{l+1}, \ldots, z'_Q\right)$ constitute an IID sample from P_{Normal}. Then, the rate of the normal test examples $\left(z'_{l+1}, \ldots, z'_Q\right)$ that are erroneously classified as anomalous, i.e., the *false alarm rate*, will be well-calibrated. The semi-supervised approach is, however, less practical in the online mode since it requires label feedback for each $z_n : n = l+1, \ldots, N$ in order to know whether z_n is part of $\left(z'_{l+1}, \ldots, z'_Q\right)$ and, hence, should be added to the training set before the classification of z_{n+1}. The semi-supervised offline CAD mitigates this practical problem at the cost of losing the theoretical guarantee of a well-calibrated false alarm rate.

Regardless of whether CAD operates in the unsupervised or semi-supervised mode, classification performance is obviously dependent on the chosen NCM and how well it discriminates between examples from $P_{Abnormal}$ and P_{Normal}. In the unsupervised mode, classification performance is further dependent on the character of $P_{Abnormal}$; intuitively, if the abnormal examples vary greatly, i.e., $P_{Abnormal}$ has high entropy, then performance in the unsupervised mode should not be worse than in the semi-supervised mode. However, if abnormal examples vary relatively little from each other, i.e., $P_{Abnormal}$ has very small entropy, then performance in the unsupervised mode may suffer as new abnormal examples might not differ significantly from the abnormal examples in the training set.

4.5 Tuning of the Anomaly Threshold

In the case of the unsupervised CAD, ϵ should generally be close to λ in order to achieve a good balance between the sensitivity and precision. Indeed, assuming an ideal NCM such that $\alpha_i > \alpha_j$ for any pair of examples z_i and z_j belonging to the abnormal and normal classes, respectively, it is intuitively clear that $\epsilon = \lambda$ would result in sensitivity and precision both being close to 100%. Considering the semi-supervised CAD, ϵ corresponds the expected false alarm rate (Section 4.4) and should therefore be set as low as possible while still maintaining reasonable sensitivity. Nevertheless, setting $\epsilon < \frac{1}{l}$ should always be avoided for both the unsupervised and semi-supervised CAD and regardless of the NCM, since the sensitivity to abnormal examples will then be zero. To see this, assume we observe an abnormal example z_{l+1} such that $\alpha_{l+1} \gg \alpha_i : i = 1, \ldots, l$. From line 4 of Algorithm 1, it is clear that $p_{l+1} = \frac{1}{l+1}$. Hence, if $\epsilon < \frac{1}{l+1}$, z_{l+1} will not be classified as anomalous, even though it is very extreme.

5 Sequential Conformal Anomaly Detection in Trajectories

The only design parameter of CAD is the NCM. In principle, any real-valued function $A(B, z)$ that accurately discriminates between examples from the normal and abnormal classes would be appropriate. Assuming that each example represents a complete trajectory, most, if not all, of the previously proposed trajectory similarity measures (Section 2.2) would be applicable as NCMs for offline conformal anomaly detection in a trajectory database. However, the application of CAD for sequential anomaly detection in trajectories imposes some further requirements on the NCM. That is, it should enable the computation of a *preliminary* nonconformity score for a partially observed trajectory; otherwise, anomaly detection is delayed until the complete trajectory has been observed. Further, the p-value for an incomplete trajectory should *monotonically decrease* as the trajectory is sequentially updated; if this is not true, then the property well-calibrated alarm rate (Theorem 2) will no longer be valid. These properties are not fulfilled by previous NCMs (e.g., [3]) or other trajectory similarity measures proposed for anomaly detection.

In Section 5.2, we propose the Directed Hausdorff k-nearest Neighbour (DH-kNN) NCM for sequential conformal anomaly detection in trajectories. It is based on a nearest neighbour NCM (Section 5.1.1) in combination with the directed Hausdorff distance (DHD) (Section 5.1.2). One of the key properties of DHD is that examples, in our case trajectories, are represented as *point sets* of arbitrary size, which may be incrementally updated. That is, trajectories are not required to be complete or normalised to fixed-dimensional feature vectors. Moreover, DHD has some properties that can be used for proving the monotonicity of the p-values during sequential update of an incomplete trajectory using DH-kNN NCM (Theorem 3).

5.1 Preliminaries

5.1.1 Nearest Neighbour Nonconformity Measures

NCMs based on the nearest neighbour principle have previously been proposed for supervised classification [3] and anomaly detection [38]. The intuition is that examples of the same class are close to each other in feature space, while examples of different classes are further away from each other. Nearest neighbour methods are relatively easy to implement, require little in the way of tuning and often perform quite well [39]. In particular, nearest neighbour methods are appropriate for online learning since they do not require extensive update of model parameters when new training data is added (compared to, e.g., neural networks and SVMs). Moreover, nearest neighbour algorithms have some strong consistency results: The asymptotic error rate is less than twice the Bayes error rate, which corresponds to the minimum achievable error rate given the distribution of the data [40].

5.1.2 Hausdorff Distance

Generally speaking, the Hausdorff distance is a dissimilarity measure for two sets of points in a metric space. It is a well known distance measure in the field of computational geometry and image processing, where it has been applied for shape matching and shape recognition [4]. Given two sets of points $A, B \subseteq \mathbb{R}^d$, DHD, $\overrightarrow{\delta}_H$, from A to B is defined as:

$$\overrightarrow{\delta}_H(A, B) = \max_{a \in A} \left\{ \min_{b \in B} \{dist(a, b)\} \right\}, \tag{10}$$

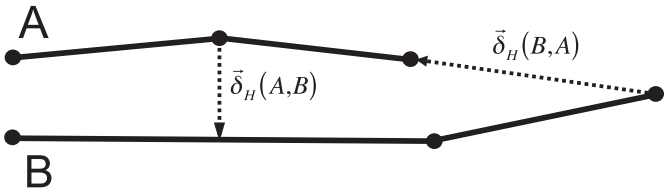

Fig. 1. Illustration of the directed Hausdorff distances between two polygonal curves *A* and *B*.

where distance between points is measures by some metric $dist\,(a, b)$, typically ED. That is, $\overrightarrow{\delta_H}(A, B)$ corresponds to the maximum distance from a point in A to the closest point in B. Assuming that the point sets represent different shapes, the DHD captures the degree to which shape A resembles some *part* of shape B. Hence, it is a natural distance measure when A is incomplete, since it does not require that every part of B is matched with some part of A. The DHD between two polygonal curves are illustrated in Fig. 1. It should be noted that the DHD is not a metric since it is not symmetric; the reverse $\overrightarrow{\delta_H}(B, A)$ is in general different from $\overrightarrow{\delta_H}(A, B)$.

5.2 The Directed Hausdorff k-Nearest Neighbours Nonconformity Measure

We propose combining the previously proposed nearest neighbour NCM for anomaly detection [38] with the DHD as follows: Assume a bag of unlabelled examples $\{z_1, \ldots, z_n\}$ where each $z_i \subseteq \mathbb{R}^d : i = 1, \ldots, n$ is represented by a non-empty set of points in a d-dimensional space. DH-kNN NCM for z_i relative $\{z_1, \ldots, z_n\} \setminus z_i$ is defined as:

$$\alpha_i := \sum_{j=1}^{k} \overrightarrow{\delta_H}\left(z_i, NN\left(z_i, \{z_1, \ldots, z_n\} \setminus z_i, j\right)\right), \tag{11}$$

where $NN\left(z_i, \{z_1, \ldots, z_n\} \setminus z_i, j\right) \in \{z_1, \ldots, z_n\} \setminus z_i$ corresponds to the jth nearest neighbour to z_i according to (10). One of the key properties of (11) is that the preliminary p-value $\hat{p}_{l+1}$ for an incomplete test example $z^*_{l+1} \subset z_{l+1}$ monotonically decreases as more points are included from z_{l+1}; this property is formalised by Theorem 3, which is proven below:

Theorem 3. *Assume that z_i^* and z_i^{**} are non-empty subsets of z_i such that $z_i^* \subset z_i^{**} \subset z_i$. Then, the corresponding preliminary p-values $\hat{p}_i$ and $\hat{p}'_i$ and the final p-value p_i for the ith example relative $\{z_1, \ldots, z_n\} \setminus z_i$, estimated according to (11), must satisfy $\hat{p}_i \geq \hat{p}'_i \geq p_i$. That is, the p-value for a subset of z_i monotonically decreases as more points are included in the subset.*

Proof. From (10) it is clear that $\overrightarrow{\delta_H}\left(z_i^*, z\right) \leq \overrightarrow{\delta_H}\left(z_i^{**}, z\right) \leq \overrightarrow{\delta_H}(z_i, z)$ for any $z \in \{z_1, \ldots, z_n\}$; the maximum distance from any point $a \in A$ to the closest point in B monotonically increases as more points are added to A. Thus, the sum in (11) also monotonically increases as more points are considered from z_i. The preliminary nonconformity scores $\hat{\alpha}_i$ and $\hat{\alpha}'_i$ and the final nonconformity score α_i for the ith example relative $\{z_1, \ldots, z_n\} \setminus z_i$, calculated according to (11), must therefore satisfy:

$$\hat{\alpha}_i \leq \hat{\alpha}'_i \leq \alpha_i. \tag{12}$$

Further, considering (10) we see that the distance from each point a from a fixed set A to the closest point in B monotonically decreases as more points are added to B. Hence, it follows that $\overrightarrow{\delta_H}\left(z_j, z_i^*\right) \geq \overrightarrow{\delta_H}\left(z_j, z_i^{**}\right) \geq \overrightarrow{\delta_H}\left(z_j, z_i\right)$ for any $z_j \in \{z_1, \ldots, z_n\}$. Consequently, the sum in (11) also monotonically decreases as more points are considered from z_i. This implies that preliminary nonconformity scores $\hat{\alpha}_j$ and $\hat{\alpha}'_j$ and the final nonconformity score α_j for any other example $z_j \in \{z_1, \ldots, z_n\} \setminus z_i$ relative $\{z_1, \ldots, z_n\} \setminus z_j$, calculated according to (11), must satisfy:

$$\hat{\alpha}_j \geq \hat{\alpha}'_j \geq \alpha_j. \tag{13}$$

From (12) and (13), it follows that:

$$\left|\left\{\hat{\alpha}_j \geq \hat{\alpha}_i\right\}\right| \geq \left|\left\{\hat{\alpha}'_j \geq \hat{\alpha}'_i\right\}\right| \geq \left|\left\{\alpha_j \geq \alpha_i\right\}\right| : j = 1, \ldots, n.$$

Hence, the corresponding p-values must satisfy $\hat{p}_i \geq \hat{p}'_i \geq p_i$. □

As a consequence of Theorem 3, well-calibrated alarm rate is maintained for any conformal anomaly detector based on (11) when the new example is sequentially updated with new points. Note that this would not be the case if we used another NCM based on, for example, the *average* (instead of maximum) distance to the closest point of the other set.

5.3 The Sequential Hausdorff Nearest Neighbours Conformal Anomaly Detector

Based on (11), we propose SHNN-CAD (Algorithm 2) for sequential anomaly detection applications. Given:

- the anomaly threshold ϵ,
- the number of nearest neighbours k,
- the training set $(z_1, \ldots, z_l)$,
- the distance matrix H, where each element $H_{i,j} : i = 1, \ldots, l, j = 1, \ldots, k$ corresponds to the DHD from z_i to its jth-nearest neighbour among $\left(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_l\right)$,
- the empty priority queue Q [41],
- the test example $z_{l+1} = \{x_1 \cup x_2 \cup \cdots \cup x_L\}$ observed as a sequence of disjunct subsets $x_1, \ldots, x_L$ such that $x_i \cap x_j = \emptyset : i, j = 1, \ldots, L \;\&\; j \neq i$,

SHNN-CAD sequentially updates the classification of z_{l+1} and outputs:

- the sequence of indicator variables $\text{Anom}^{\epsilon}_{l+1,1}, \ldots, \text{Anom}^{\epsilon}_{l+1,L}$, where each $\text{Anom}^{\epsilon}_{l+1,i} : i = 1, \ldots, L-1$ corresponds the preliminary classification of z_{l+1} based on the subset $\{x_1 \cup \cdots \cup x_i\} \subset z_{l+1}$, and $\text{Anom}^{\epsilon}_{l+1,L}$ corresponds to the final classification of z_{l+1},
- the vector $(h_1, \ldots, h_l)$ where $h_i : i = 1, \ldots, l$ corresponds to the DHD from z_{l+1} to z_i,
- the vector $\left(h'_1, \ldots, h'_l\right)$ where $h'_i : i = 1, \ldots, l$ corresponds to the DHD from z_i to z_{l+1}.

The algorithm works as follows: For each $z_i : i = 1, \ldots, l$, the DHD from z_{l+1} to z_i, h_i, is initialised to zero and the sum of the distances to the $(k-1)$-nearest neighbours of z_i is precomputed (lines 1–4 of Algorithm 2). The following nested loops (lines 5–31) are then repeated for each

x_j:$j = 1, \ldots, L$ and z_i:$i = 1, \ldots, l$: First, h_i is updated as the DHD from $\{x_1 \cup \cdots \cup x_j\}$ to z_i, which is calculated in an incremental manner based on the distance from $\{x_1 \cup \cdots \cup x_{j-1}\}$ to z_i calculated in the previous iteration (line 7). If Q, which stores the current k-nearest neighbour distances to $\{x_1 \cup \cdots \cup x_j\}$, contains less than k distance values, h_i is simply inserted into Q (line 9). If Q contains k distance values and h_i is less than the current kth nearest neighbour distance, the maximum distance value in Q is removed and h_i is inserted into Q (lines 12–13). The DHD from z_i to z_{l+1}, h'_i, is updated as the distance from z_i to $\{x_1 \cup \cdots \cup x_j\}$. The preliminary nonconformity score $\hat{\alpha}_i$ for z_i is then updated depending on whether h'_i is smaller than the DHD from z_i to its kth-nearest neighbour in the training set (lines 16–21). Next, the k distance values are extracted from Q and $\hat{\alpha}_{l+1}$ is updated as the sum of these distances (lines 23–24). Finally, the preliminary p-value and the classification of the test example are updated (lines 25–30).

In the case of the unsupervised online SHNN-CAD, z_{l+1} is added to the training set when $x_1, \ldots, x_L$ have been observed. This is also true for the semi-supervised online SHNN-CAD if z_{l+1} is assumed to belong to the normal class. If z_{l+1} is added to the training set, then:

1) the sorted distances to the k-nearest neighbours of each $z_1, \ldots, z_l$, i.e., rows 1–l of H, are updated based on h',
2) the sorted distances to the k-nearest neighbours of z_{l+1} are extracted from h and appended as the last row of H.

The updated training set $(z_1, \ldots, z_{l+1})$ and the updated distance matrix H are then provided as input to SHNN-CAD when classifying the next test example z_{l+2}.

5.3.1 Implementation and Complexity Analysis

A natural approach to implementing SHNN-CAD for sequential anomaly detection in trajectories is to adopt a polyline representation; each trajectory (1) is represented by an example $z_i = \{x_1 \cup x_2 \cup \cdots \cup x_{L_i}\}$ where $x_1 = \chi_1$ and $x_j = \{\chi_{j-1} + s \cdot (\chi_j - \chi_{j-1}) : s \in (0, 1]\}$ for $j = 2, \ldots, L_i$. That is, the first subset x_1 represents the first trajectory point and each remaining subset x_j represent the set of all points along the line segment that connects χ_{j-1} and χ_j. This approach allows for efficient compression of trajectories and exact calculation of (10) with time complexity $O((L_i + L_j)\, log\, (L_i + L_j))$ based on algorithms from computational geometry [4], where L_i and L_j are the number of line segments of the corresponding trajectories. Moreover, assuming that Q is implemented as a heap based on a binary tree, the time complexity of insertElement(), removeMaxElement(), maxElement() and removeAllElements() in Algorithm 2 are $O(\log(k))$, $O(\log(k))$, $O(1)$, and $O(k)$, respectively, where k is the number of elements of the queue [41].

Based on the assumptions above, the time complexity of Algorithm 2 can be analysed as follows: The sections corresponding to lines 7, 8–15, 16 and 17–21 have complexity $O(L_i \log(L_i))$, $O(\log(k))$, $O((L_i + j)\log(L_i + j))$ and $O(1)$, respectively. Thus, the overall complexity of the inner loop (lines 6–22) is:

Algorithm 2: The Sequential Hausdorff Nearest Neighbours Conformal Anomaly Detector (SHNN-CAD)

Input: Anomaly threshold ϵ, number of nearest neighbours k, training set $(z_1, \ldots, z_l)$, distance matrix H, empty priority queue Q, sequence of updates $x_1, \ldots, x_L$ of test example z_{l+1}
Output: Seq. of ind. variables $\text{Anom}^{\epsilon}_{l+1,1}, \ldots, \text{Anom}^{\epsilon}_{l+1,L}$, distance vectors $(h_1, \ldots, h_l)$ and $(h'_1, \ldots, h'_l)$.

1: **for** $i \leftarrow 1$ to l **do**
2: $v_i \leftarrow \text{sum}\{H_{i,1}, \ldots, H_{i,k-1}\}$
3: $h_i \leftarrow 0$
4: **end for**
5: **for** $j \leftarrow 1$ to L **do**
6: **for** $i \leftarrow 1$ to l **do**
7: $h_i \leftarrow \max\left\{\overrightarrow{\delta_H}(x_j, z_i), h_i\right\}$
8: **if** $i \leq k$ **then**
9: Q.insertElement(h_i)
10: **else**
11: **if** Q.maxElement() $> h_i$ **then**
12: Q.removeMaxElement()
13: Q.insertElement(h_i)
14: **end if**
15: **end if**
16: $h'_i \leftarrow \overrightarrow{\delta_H}(z_i, \{x_1 \cup \cdots \cup x_j\})$
17: **if** $h'_i < H_{i,k}$ **then**
18: $\hat{\alpha}_i \leftarrow v_i + h'_i$
19: **else**
20: $\hat{\alpha}_i \leftarrow v_i + H_{i,k}$
21: **end if**
22: **end for**
23: $(h^*_1, \ldots, h^*_k) \leftarrow Q$.removeAllElements()
24: $\hat{\alpha}_{l+1} \leftarrow \text{sum}\{h^*_1, \ldots, h^*_k\}$
25: $\hat{p}_{l+1} \leftarrow \frac{|\{i=1,\ldots,l+1 : \hat{\alpha}_i \geq \hat{\alpha}_{l+1}\}|}{l+1}$
26: **if** $\hat{p}_{l+1} < \epsilon$ **then**
27: $\text{Anom}^{\epsilon}_{l+1,j} \leftarrow 1$
28: **else**
29: $\text{Anom}^{\epsilon}_{l+1,j} \leftarrow 0$
30: **end if**
31: **end for**

$$\begin{aligned} &O\left(l \log(k) + (L_1 + j)\log(L_1 + j) + \ldots\right. \\ &\qquad \left. \ldots + (L_l + j)\log(L_l + j)\right) = \\ &= O\left(l \log(k) + l \cdot L_{max} \log(L_{max})\right), \end{aligned} \tag{14}$$

where L_{max} is the maximum number of line segments among all of the trajectories. The sections corresponding to lines 23, 24 and 25 have complexity $O(k)$, $O(k)$ and $O(l)$, respectively. Since $l > k$, the complexity of each iteration of the outer loop (lines 5–31), i.e., the update of the classification of z_n based on x_j, is equivalent to (14). Further, the pre-computations (lines 1–4) have complexity $O(l \cdot k)$. Thus, the overall time complexity of Algorithm 2 is:

$$O\left(l \cdot L_{max} \log(k) + l \cdot L^2_{max} \log(L_{max}) + l \cdot k\right). \tag{15}$$

Assuming that $L_{max} > k$, which is typically the case, (15) is reduced to $O(l \cdot L^2_{max} \log(L_{max}))$.

An alternative approach to the continuous polyline representation is to only consider the end points of the line segments, i.e., assume that $x_j = \{\chi_j\} : j = 1, \ldots, L$. This representation corresponds to a less detailed model of the actual trajectory, but allows for a more simple implementation of the DHD.

5.4 Discussion

Compared to previously proposed algorithms (Section 3), SHNN-CAD and the framework of conformal anomaly detection have some principal advantages:

- there are no assumptions regarding the structure of the probability distribution,
- it is parameter-light, i.e., there is only one free parameter k apart from the anomaly threshold,
- it requires no particular preprocessing or normalisation of trajectories,
- it supports sequential anomaly detection in incomplete trajectories,
- it supports online learning,
- it offers a well-founded approach to the tuning of the anomaly threshold ϵ (Section 4.5) with well-calibrated alarm rate in the online mode (Theorem 2 and 3).

The property that trajectories need not be complete has in fact more general implications. It may be argued that the DHD is not only robust to future data points that have not yet been observed; it also robust to previous data points that are missing due to delayed track initialisation, re-initialisation of a previous track that was lost, etc.

The DHD is by definition insensitive to the ordering of the data points. Thus, if only position is included in the point feature model, this may result in contra-intuitive matching. An example is two objects that follow the same path but travel in opposite direction. This could be addressed by simply extending the point feature model to also include the current course or velocity vector, which would also capture the speed of the object.

6 Empirical Investigations

In this section, we implement DH-kNN NCM (Section 5.2) and investigate its accuracy on three different datasets of labelled trajectories (Section 6.1). Moreover, we implement the unsupervised online SHNN-CAD (Section 4.3–4.4 and 5.3) and investigate its classification performance (Section 6.2). For simplicity, we implement an algorithm for calculating (10) that only considers the finite set of points of the trajectories. That is, we do not implemented an algorithm for calculating the exact distance between two trajectories represented as polylines, where all the intermediate points along the line segments are considered (Section 5.4).

For comparative purposes, we also implement and investigate the performance of the time-series discords algorithm [5], which was originally proposed by Keogh *et al*. for detecting anomalies in time-series data. The discords algorithm is parameter-light and has been demonstrated to have competitive classification performance on a wide range of time-series databases (e.g., [5]), including trajectory datasets [6], [16]. Hence, we use the discords algorithm as a benchmark for evaluating the relative performance of DH-kNN NCM and SHNN-CAD in all of the following experiments.

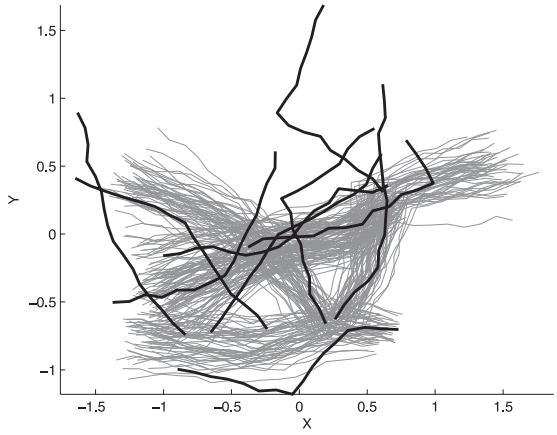

Fig. 2. Plot of trajectories from one of the 1000 subsets of synthetic trajectories used for evaluating the accuracy of different trajectory outlier measures (Section 6.1.1). Grey trajectories (250) are labelled normal and black trajectories (10) are labelled abnormal.

All algorithms are implemented in Java and executed on a Macbook Pro 2.66 GHz Intel Core 2 Duo processor with 8 GB of RAM.

6.1 Accuracy of Trajectory Outlier Measures

We investigate the accuracy of DH-kNN NCM and the discords algorithm by reproducing three previously published experiments on three different datasets of labelled trajectories (Section 6.1.2–6.1.3). The main objective is to investigate how accurately DH-kNN NCM discriminates complete abnormal trajectories from complete normal trajectories, compared to previously proposed trajectory outlier measures.

6.1.1 Synthetic Trajectories

The first experiment was originally published by Piciarelli *et al*. who investigated the accuracy of an outlier measure based on one-class SVM on a synthetic dataset [16]. The dataset[1] was created by the authors and consists of 1000 randomly generated trajectory subsets. Each subset contains 260 two-dimensional trajectories of length 16 without any time information. Of the 260 trajectories, 250 belong to five different clusters and are labelled normal. The remaining 10 are stray trajectories that do not belong to any cluster and are labelled abnormal (see Fig. 2 for a plot of one of the subsets). For each of the 1000 subsets, the authors calculated the outlier score for each of the 260 trajectories relative the remaining 259. The *error rate* was calculated by averaging over the number of normal trajectories among the top-10 trajectories with highest outlier scores. We repeat this procedure for DH-kNN NCM by calculating the outlier score for each trajectory according to (11). Moreover, we calculate the top-ten discords [5], where each trajectory corresponds to a subsequence. The average accuracy, i.e. 1 minus the

1. http://avires.dimi.uniud.it/papers/trclust/

TABLE 1
Average Accuracy on the Synthetic Trajectories (Section 6.1.1)

	# of most similar neighbours considered				
Outlier Measure	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
DH-kNN NCM	96.51%	97.09%	97.05%	96.95%	96.77%
SVM [16]	96.30% [16]				
Discords [5]	97.06%				

Note that accuracy results for SVM are 1 minus the corresponding error rate reported by Piciarelli et al. [16].

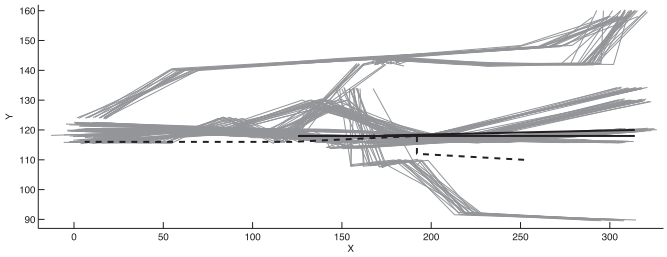

Fig. 3. Plot of the 239 trajectories from the first set of recorded video trajectories (Section 6.1.2). Grey corresponds to normal trajectories and solid and dashed black correspond to the two abnormal trajectories.

corresponding average error rate, for all three algorithms are summarised in Table 1.

6.1.2 First Set of Recorded Video Trajectories

In the second experiment, we investigate accuracy on a public dataset of 239 recorded video trajectories[2] that were extracted from IR surveillance videos using a motion detection and tracking algorithm [42]. The trajectories are three-dimensional including time stamps and each have length five. Of the 239 trajectories, 237 are labelled normal and two are labelled abnormal (see Fig. 3). We calculate the nonconformity score for each of the 239 trajectories relative the rest according to (11) for $k = 1, \ldots, 5$. However, in contrast to Pokrajac [42], we only consider the two-dimensional spatial location of each trajectory point, i.e., we do not use the time information. Sorting the resulting nonconformity scores, we observe that the two trajectories labelled abnormal have the top-two largest nonconformity scores, regardless of k. Moreover, we observe that the top-two discords (for the two-dimensional representation) also correspond to the abnormal trajectories. Hence, similar to previous algorithms [42], DH-kNN NCM and discords achieve perfect accuracy on this dataset.

6.1.3 Second Set of Recorded Video Trajectories

The second set, which is known at the LAB-dataset [12], consists of 152 trajectories labelled normal and 8 trajectories labelled abnormal. The trajectories contain no time stamps and are of varying length, where the maximum and average length is 425 and 192 points, respectively (Fig. 4). The trajectories were extracted from a video recording where people were tracked in a controlled laboratory environment. Movements were planned so that normal trajectories can be grouped into four clusters and the abnormal trajectories varied deliberately from the normal trajectories. Analogously to the experimental setup described by Khalid [12], we evaluate accuracy of DH-kNN and the discords algorithm as follows: We first construct a training set by randomly sampling without replacement half of the 152 normal trajectories. The remaining 76 normal trajectories and the eight abnormal trajectories are used as the test set. For each test trajectory, the nonconformity score relative the training set is estimated according to (11). Moreover, the top-eight discords among the test trajectories are calculated relative the training set, where each trajectory in the training and test sets is normalised to length 50 based on linear interpolation; note that the discords algorithm require that the trajectories have the same length. Analogously to the previous experiments (Section 6.1.1–6.1.3), accuracy is calculated as the fraction of the top-eight nonconforming test trajectories and the top-eight discords that are labelled abnormal. This procedure is repeated 100 times. Average accuracy for DH-kNN NCM with $k = 1, \ldots, 5$ and the discords algorithm are summarised in Table 2. Note that the results reported by Khalid [12] are based on a single run, i.e., not the average of 100 experiments.

2. www.cs.umn.edu/~aleks/inclof

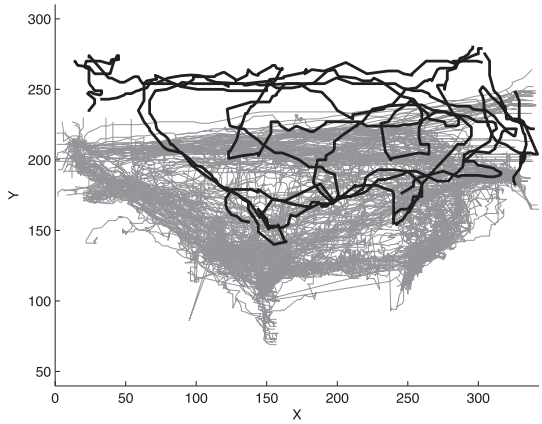

Fig. 4. Plot of the recorded video trajectories from the LAB-dataset [12] (Section 6.1.3). Grey trajectories (152) are labelled normal and black trajectories (8) are labelled abnormal.

6.2 Unsupervised Online Learning and Sequential Anomaly Detection

In this experiment, we investigate the sequential classification performance of the unsupervised online SHNN-CAD and the discords algorithm on a relatively large synthetic dataset. The objectives of the experiment are to:

1) Investigate how the sensitivity and precision depends on the size of training set and the value of ϵ.

TABLE 2
Avg. Accuracy on 100 Test Sets of Labelled Video Trajectories (Section 6.1.3)

	# of nearest neighbours considered				
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
DH-kNN NCM	87.9%	88.3%	87.3%	85.9%	84.1%
Discords	98.6%				
m-Mediods [12]	100%[12]				

Note that results for m*-Mediods were reported by Khalid [12] and are based on a single test set.*

2) Investigate how the *detection delay*, i.e., the number of observed trajectory points prior to the detection of an abnormal trajectory, depends on the size of the training set and the value of ϵ. Recall that a low detection delay is advantageous since it enables earlier response in a realtime surveillance application (Section 3.2.2).
3) Investigate the relative performance of SHHN-CAD vs. discords.
4) Validate that the rate of detected anomalies (both normal and abnormal trajectories) is indeed well-calibrated for SHNN-CAD.

6.2.1 Dataset

For this experiment, a dataset of synthetic trajectories[3] was generated by ourselves using the publicly available trajectory generator software[4] written by Piciarelli. The dataset, which is previously unpublished, contains 100 random trajectory sequences, where each sequence consists of 2000 two-dimensional trajectories of length 16 that are labelled normal or abnormal. Each sequence, which is independent of the other 99 sequences, was created as follows: First, a set of 2000 normal trajectories from 10 different trajectory clusters and another set of 1000 abnormal trajectories from 1000 different "clusters" were created using the trajectory generator with the randomness parameter set to the default value 0.7. Next, a sequence of 2000 trajectories was created by random sampling without replacement from the set of normal trajectories. Finally, each normal trajectory in the sequence was independently and with probability 1% replaced by an abnormal trajectory, which was randomly sampled without replacement from the set of abnormal trajectories. Hence, the trajectory generation process is consistent with (9) with $\lambda = 0.01$. A subset of the normal trajectories and all abnormal trajectories from the first sequence are shown in Fig. 5.

6.2.2 Design

For SHNN-CAD, we fix $k = 2$ since it was shown to give the best accuracy results for DH-kNN NCM on all of the previous datasets (Tables 1 and 2). For each of the 100 sequences of 2000 trajectories, we allocate the first three trajectories as initial training set and calculate the corresponding distance matrix M of size 3×3 (Section 5.3); note that three is the minimum size of the training set for which (11) is defined when $k = 2$. The remaining 1997 trajectories are sequentially classified by the unsupervised online SHNN-CAD (Algorithm 2). The whole process is repeated for three different anomaly thresholds: $\epsilon = 0.005, 0.01$ and 0.02. The choice of $\epsilon = 0.01$ is motivated by the discussion in Section 4.5, i.e., that the anomaly threshold should be set close to λ. Nevertheless, the exact value of λ may be unknown and, hence, we also investigate the performance for $\epsilon = 0.005$ and 0.02. Moreover, in order to avoid the problem of zero sensitivity when the size of training set is relatively small (Section 4.5), the anomaly threshold is dynamically re-tuned to $(l+1)^{-1}$ as long as $l < \epsilon^{-1}$.

3. https://www.researchgate.net/profile/Rikard_Laxhammar/
4. http://avires.dimi.uniud.it/papers/trclust/create_ts2.m

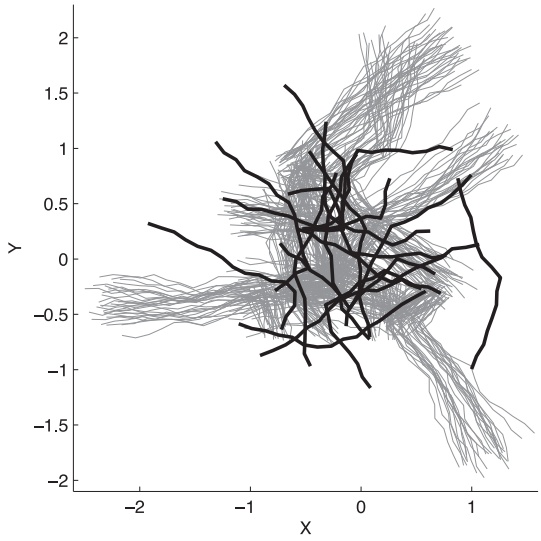

Fig. 5. Plot of trajectories from the first sequence of synthetic trajectories used in the online learning and sequential anomaly detection experiments (Section 6.2). Grey trajectories (300) are labelled normal and black trajectories (17) are labelled abnormal. Note that for clarity, only 300 of the total 1983 normal trajectories are plotted.

The experiment is also reproduced for a brute-force implementation of an online version of the top-k discords algorithm. According to the general definition [5], the top-k discords of a time-series correspond to the top-k non-overlapping subsequences of predefined length m that have the largest distances to their nearest non-overlapping match. In previous work [6], [16] and in the previous experiments of this paper, each subsequence simply corresponds to a complete and normalised trajectory of fixed length m. However, since we are now concerned with sequential anomaly detection in incomplete trajectories, we need to consider a subsequence length that is less than the full length of the trajectories. Setting m to a low value may decrease the detection delay but may also decrease the sensitivity to abnormal trajectories when the length of the abnormal part is greater than m. In these experiments, we consider $m = 4, 8$ and 12, which correspond to a quarter, a half and three quarters of the full trajectory, respectively. For each update $j = m, \ldots, 16$ of trajectory $i = 4, \ldots, 2000$, the subsequence $\chi^i_{j-m+1}, \ldots, \chi^i_j$ is extracted, added to the training set of previous subsequences and classified as abnormal or normal depending on whether it is among the top-k discords of the updated training set. Intuitively, k should be chosen depending on the expected number of discords in the training set. Assuming that approximately 1% of all trajectories are abnormal and that each abnormal trajectory corresponds to one discord, k is dynamically tuned based on the number of trajectories observed so far, i.e., $k = \lceil 0.01i \rceil$. Nevertheless, the prior probability of an abnormal trajectory may be unknown and abnormal trajectories may include more than one discord, in particular when m is relatively small. Hence, we also investigate performance when $k = \lceil 0.005i \rceil$ and $\lceil 0.02i \rceil$.

6.2.3 Results

In order to evaluate the relative performance of SHNN-CAD vs. discords (objective 3), we calculate the average

TABLE 3
Average F_1-Score and Mean Detection Delay for the Online Learning and Sequential Anomaly Detection Experiments (Section 6.2.3)

			F_1	Detection delay
SHNN-CAD	$\epsilon = 0.005$		0.53	12.7
	$\epsilon = 0.01$		0.75	10.3
	$\epsilon = 0.02$		0.62	8.0
Discords	$k = \lceil 0.005i \rceil$	$m = 4$	0.46	10.2
		$m = 8$	0.58	10.6
		$m = 12$	0.62	12.8
	$k = \lceil 0.01i \rceil$	$m = 4$	0.62	8.8
		$m = 8$	0.74	9.7
		$m = 12$	0.76	12.3
	$k = \lceil 0.02i \rceil$	$m = 4$	0.63	7.3
		$m = 8$	0.65	8.9
		$m = 12$	0.60	12.1

Note that the maximum possible detection delay is 16*, since this is the length of each trajectory.*

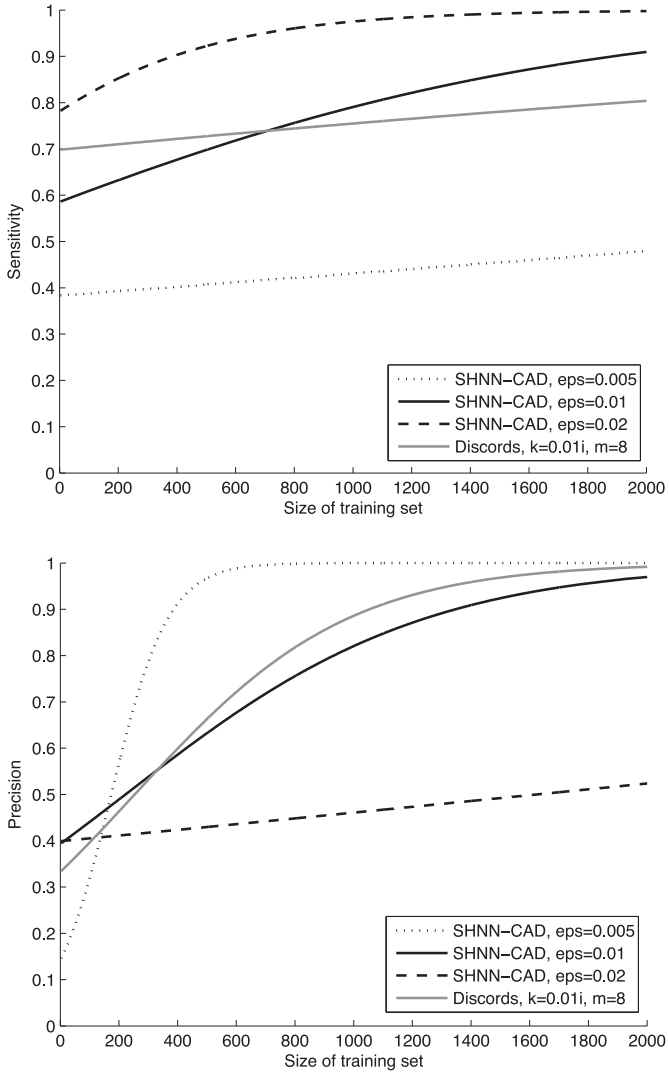

Fig. 6. Plots of the sensitivity (upper plot) and precision (lower plot) dependent on the size of the training set according to the logistic regression models, which are estimated based on the online learning and sequential anomaly detection results (Section 6.2).

F_1-score and the average detection delay for each combination of the detectors' parameters (Table 3). The F_1-score corresponds to an evenly weighted combination of precision and recall:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}, \tag{16}$$

and it is therefore useful for comparing the overall classification performance of multiple anomaly detectors. In order to address objective 1, we estimate logistic regression models for the sensitivity and precision, respectively, dependent on the size of the training set (Fig. 6). The coefficients of the logistic regression models for the sensitivity are estimated based on the final classification results for the subset of trajectories that are labelled abnormal. In case of the precision, the coefficients are estimated based on the true label for those trajectories that are detected as anomalous. Further, in order to address objective 2, we estimate a linear regression model for the detection delay dependent on the current size of the training set (Fig. 7). Finally, a histogram of the overall alarm rate for SHNN-CAD $\epsilon = 0.01$ is shown in Fig. 8 (objective 4). The average processing time on each sequence of 2000 trajectories was 19.1 seconds for SHNN-CAD with $\epsilon = 0.01$ and 45.0 seconds for discords with $m = 8$ and $k = \lceil 0.01i \rceil$.

6.3 Discussion

For the first two datasets (Section 6.1.1–6.1.2), it is clear that DH-kNN NCM is an accurate outlier measure, regardless of k (Table 1). On the third dataset, however, results for DH-kNN NCM were slightly worse compared to discords (Table 2), which indicate that the latter has some edge over the former in the case of complete trajectories. Nevertheless, the advantage of DH-kNN NCM becomes more clear when we consider the results for SHNN-CAD during online learning and sequential anomaly detection:

Considering Table 3, it is clear that the best classification performance for SHHN-CAD ($F_1 = 0.75$) is achieved when $\epsilon = 0.01$. This confirms our hypothesis that ϵ should be close to λ (Section 4.5). The best classification performance for SHNN-CAD is approximately the same as the best results for discords: $F_1 = 0.74$ and $F_1 = 0.76$ when $k = \lceil 0.01i \rceil$ and $m = 8, 12$, respectively. The detection delay for SHNN-CAD with $\epsilon = 0.01$ is also similar to that of discords with $m = 8$ (10.3 and 9.7 points, respectively). However, the detection delay for discords with $m = 12$ is considerably higher (12.3 points). Hence, we select $k = \lceil 0.01i \rceil$ and $m = 8$ when further investigating how sensitivity, precision and detection delay depend on the size of the training set[5] (Figs. 6 and 7).

Examining Fig. 6, it is clear that the sensitivity and precision improve as more unlabelled training data is accumulated. In particular, there is generally a large increase in precision. Moreover, the expected detection delay generally decreases as more the training is accumulated, with the exception of SHNN-CAD with $\epsilon = 0.005$. The explanation for this exception does not seem obvious. Nevertheless, a similar pattern was in fact observed for discords with $k = \lceil 0.005i \rceil$, which suggests that for low anomaly thresholds,

5. For clarity and brevity, we omit more detailed results for the remaining parameter values of discords in Figs. 6 and 7.

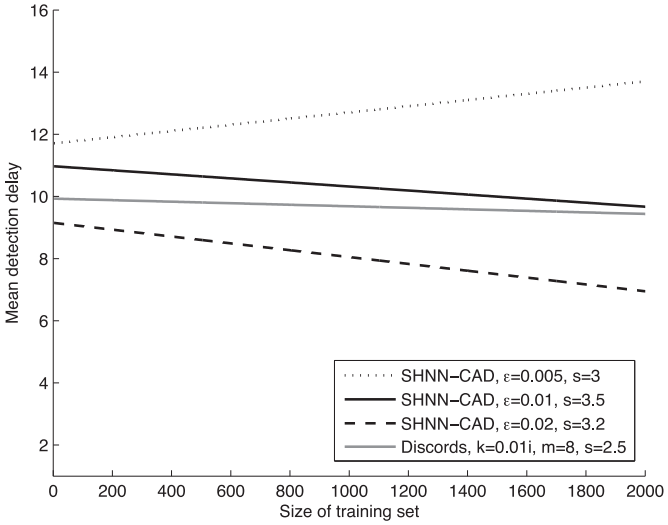

Fig. 7. Plot of the mean detection delay dependent on the size of the training set according to the linear regression models, which are estimated based on the online learning sequential anomaly detection results (Section 6.2). The variable *s* in the legend is an unbiased estimate of the standard deviation of the error term of the linear regression model.

the detection delay generally *increases* as the training set grows. Considering the end of the curves, i.e., at the point where the size of the training set approaches 2000, we see that the sensitivity and precision in most cases are still increasing, albeit at a slower rate. This indicates that the anomaly detectors may still benefit from more training data but are approaching the point where they may be considered fully trained.

Comparing the sensitivity for SHNN-CAD $\epsilon = 0.01$ and discords, we see that the latter initially has a slight advantage until the size of the training set is 700. From that point, the sensitivity of SHNN-CAD becomes superior and increases more rapidly than for discords. When the size of the training set has reached 2000, the sensitivity of SHNN-CAD and discords are approximately 0.9 and 0.8, respectively. In case of the precision, the difference is more subtle; SHNN-CAD has initial advantage but it is passed by discords after the size of the training set has reached 400. When the size of training set reaches 2000, the precision of SHNN-CAD and discords are both close to 1 with a small advantage to discords.

It should be noted that all the parameter values have a significant impact on performance for both SHNN-CAD and discords. However, the discords algorithm involves tuning of one more parameter than SHNN-CAD, i.e., the length of the subsequence m. Moreover, the choice of k for discords may not be as straightforward as the choice of ϵ for SHNN-CAD, since the optimal value of the former may depend on m. For example, the best classification performance for $m = 8, 12$ is achieved with $k = \lceil 0.01i \rceil$. But for $m = 4$, the classification performance and the detection delay are both superior with $k = \lceil 0.02i \rceil$.

Finally, considering Fig. 8, we see that the alarm rate on each of the 100 sequences is close to the specified anomaly threshold. Hence, the alarm rate of the online unsupervised SHNN-CAD is indeed well-calibrated in practice.

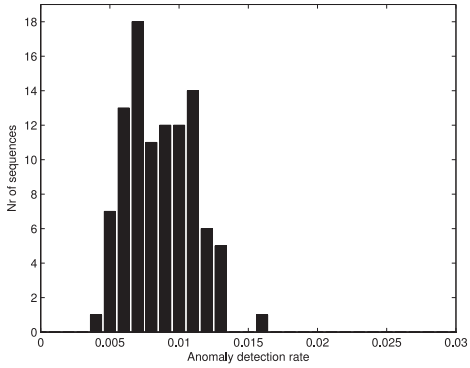

Fig. 8. Histogram of the overall alarm rate (normal and abnormal trajectories) for the unsupervised online SHNN-CAD with $\epsilon = 0.01$. Each data point corresponds to the average anomaly detection rate on one of the 100 random sequences of trajectories (Section 6.2).

7 Generalisation and Future Work

In the empirical investigations, we have only considered two-dimensional trajectories. Nevertheless, SHNN-CAD should be applicable to higher-dimensional trajectories including, e.g., velocity features, assuming that the different dimensions are properly normalised. Thus, future work includes investigating different normalisation techniques and performance on higher-dimensional trajectory datasets.

Assuming that an appropriate NCM is defined, CAD is applicable in virtually any unsupervised or semi-supervised, online or offline, anomaly detection problem. In fact, any existing anomaly or outlier detection algorithm, which has shown good classification performance in the current application domain, may potentially be adopted as a NCM in the conformal anomaly detection framework. The main benefit of such "wrapper" strategy is that it mitigates the problems with ad-hoc anomaly thresholds and offers theoretical guarantees regarding the calibration of the alarm rate. Hence, an interesting path for future work would be to investigate NCMs that are appropriate for other data types than trajectories.

An important aspect that has not yet been addressed in this work is how to deal with a continually increasing training set; if no pruning strategy is used, the size of the training set might eventually render online anomaly detection unfeasible. Moreover, there is the risk that the underlying distribution for normal data will change, known as *population drift* [27]. Hence, long-term learning strategies for pruning and determining when the training set is no longer valid should be investigated in future work.

8 Conclusion

Previous algorithms for detecting anomalous trajectories typically suffer from one or more limitations that have been identified and discussed in this article, i.e., they are primarily designed for offline anomaly detection and suffer from tuning of many parameters, including ad-hoc anomaly thresholds. In this article, we have proposed and investigated SHNN-CAD for online learning and sequential anomaly detection in trajectories. This is a parameter-light algorithm that offers a well-founded approach to tuning the anomaly threshold according to the desired alarm rate

or the expected frequency of anomalies. We have implemented and evaluated the classification performance of SHNN-CAD and, for comparative purposes, the discords algorithm on four different trajectory datasets. In particular, we have investigated the performance during unsupervised online learning and sequential anomaly detection on a relatively large set of synthetic trajectories. The results showed that SHNN-CAD achieved competitive classification performance with minimum parameter tuning. Future work includes investigating the application of SHNN-CAD on higher-dimensional trajectories with velocity features and long-term learning strategies for pruning the training set. The results regarding conformal anomaly detection generalises beyond the problem of detecting anomalous trajectories; assuming that an appropriate NCM is defined, CAD is applicable in virtually any unsupervised or semi-supervised, online or offline, anomaly detection problem.

Acknowledgments

This work has been supported in part by Saab AB and in part by the Swedish Knowledge Foundation in cooperation with University of Skövde. The authors would like to thank K. Wallenius and E. Sviestins for providing valuable feedback regarding the manuscript of this article and V. Vovk for valuable feedback regarding conformal anomaly detection. Moreover, the authors would like to acknowledge C. Piciarelli and A. Lazarević for making their datasets publicly available, and S. Khalid for kindly providing the LAB-dataset.

References

[1] B. Morris and M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1114–1127, Aug. 2008.

[2] R. Laxhammar and G. Falkman, "Sequential conformal anomaly detection in trajectories based on Hausdorff distance," in *Proc. 14th Int. Conf. Inform. Fusion*, Chicago, IL, USA, 2011.

[3] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. New York, NY, USA: Springer-Verlag, Inc., 2005.

[4] H. Alt, "The computational geometry of comparing shapes," in *Efficient Algorithms*. Berlin, Germany: Springer, 2009, pp. 235–248, LNCS 5760.

[5] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Efficiently finding the most unusual time series subsequence," in *Proc. 5th IEEE ICDM*, Washington, DC, USA, 2005.

[6] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk aware discord discovery: Finding unusual time series in terabyte sized datasets," *Knowl. Inf. Syst.*, vol. 17, no. 2, pp. 241–262, 2008.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM CSUR*, vol. 41, no. 3, pp. 1–58, 2009.

[8] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proc. NIPS*, vol. 12. 2000.

[9] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Proc 5th Int. Conf. MLDM*, Leipzig, Germany, 2007.

[10] W. Hu *et al.*, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, Sept. 2006.

[11] H. Dee and S. Velastin, "How close are we to solving the problem of automated visual surveillance? A review of real-world surveillance, scientific progress and evaluative mechanisms," *Mach. Vis. Appl.*, vol. 19, no. 5–6, pp. 329–343, 2008.

[12] S. Khalid, "Motion-based behaviour learning, profiling and classification in the presence of anomalies," *Pattern Recognit.*, vol. 43, no. 1, pp. 173–186, 2010.

[13] R. R. Sillito and R. B. Fisher, "Semi-supervised learning for anomalous trajectory detection," in *Proc. BMVC*, 2008, pp. 1034–1044.

[14] F. Porikli, "Trajectory distance metric using hidden Markov model based representation," in *Proc. 6th IEEE Int. Workshop PETS*, 2004.

[15] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 647–657, Sept. 2010.

[16] C. Piciarelli, C. Micheloni, and G. Foresti, "Trajectory-based anomalous event detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1544–1554, Nov. 2008.

[17] M. Vlachos, G. Kollios, and D. Gunopoulos, "Discovering similar multidimensional trajectories," in *Proc. 18th IEEE Int. Conf. Data Eng.*, San Jose, CA, USA, 2002.

[18] B. Morris and M. Trivedi, "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis," in *Proc. IEEE 5th Int. Conf. AVSS*, Santa Fe, NM, USA, 2008.

[19] C. Piciarelli and G. Foresti, "On-line trajectory clustering for anomalous events detection," *Pattern Recognit. Lett.*, vol. 27, no. 15, pp. 1835–1842, Nov. 2006.

[20] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *Proc. IEEE ICIP*, 2005.

[21] C. Brax, L. Niklasson, and M. Smedberg, "Finding behavioral anomalies in public areas using video surveillance data," in *Proc. 11th Int. Conf. Inform. Fusion*, Cologne, Germany, 2008.

[22] J. Owens and A. Hunter, "Application of the self-organising map to trajectory classification," in *Proc. 3rd IEEE Int. Workshop VS*, Dublin, Ireland, 2000.

[23] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proc. 24th ICDE*, Washington, DC, USA, 2008.

[24] R. Laxhammar, G. Falkman, and E. Sviestins, "Anomaly detection in sea traffic - A comparison of the Gaussian mixture model and the kernel density estimator," in *Proc. 12th Int. Conf. Inform. Fusion*, Washington, DC, USA, 2009.

[25] N. Taleb, *Fooled by Randomness: The Hidden Role of Chance in the Markets and in Life*. London, U.K.: Penguin Books, 2004.

[26] E. Keogh *et al.*, "Compression-based data mining of sequential data," *Data Min. Knowl. Discov.*, vol. 14, no. 1, pp. 99–129, 2007.

[27] D. Hand, "Classifier technology and the illusion of progress," *Statist. Sci.*, vol. 21, no. 1, pp. 1–14, 2006.

[28] M. Markou and S. Singh, "Novelty detection: A review - Part 1: Statistical approaches," *Sig. Process.*, vol. 83, no. 12, pp. 2481–2497, 2003.

[29] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[30] M. Zhao and V. Saligrama, "Anomaly detection with score functions based on nearest neighbor graphs," in *Proc. NIPS*, 2009, pp. 2250–2258.

[31] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM TISSEC*, vol. 3, no. 3, pp. 186–205, Aug. 2000.

[32] M. Riveiro, "Visual analytics for maritime anomaly detection," PhD thesis, Örebo Univ., Sweden, 2011.

[33] Y. Bu, L. Chen, and D. Wai-Chee Fu, A. Liu, "Efficient anomaly monitoring over moving object trajectory streams," in *Proc. 15th ACM SIGKDD*, New York, NY, USA, 2009.

[34] A. Gammerman and V. Vovk, "Hedging predictions in machine learning," *Comput. J.*, vol. 50, no. 2, pp. 151–163, 2007.

[35] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *J. Mach. Learn. Res.*, vol. 9, pp. 371–421, Mar. 2008.

[36] D. Hawkins, *Identification of Outliers*. New York, NY, USA: Chapman Hall, 1980.

[37] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proc. 17th ICML*, San Francisco, CA, USA, 2000.

[38] R. Laxhammar and G. Falkman, "Conformal prediction for distribution-independent anomaly detection in streaming vessel data," in *Proc. 1st Int. Workshop StreamKDD*, New York, NY, USA, 2010.

[39] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NJ, USA: Pearson Education, 2003.

[40] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York, NY, USA: Wiley, 1973.

[41] M. Goodrich and R. Tamassia, *Data Structures and Algorithms in Java*. New York, NY, USA: Wiley, 1998.

[42] D. Pokrajac, A. Lazarevic, and L. Latecki, "Incremental local outlier detection for data streams," in *Proc. IEEE Symp. CIDM*, Honolulu, HI, USA, 2007.

Rikard Laxhammar received the Ph.D. degree in computer science at University of Skövde, Skövde, Sweden, in 2014. He is employed by Saab AB, Järfälla, Sweden, since 2007, where he works as a software engineer. His main research interests are machine learning and anomaly detection. In his previous research, he has investigated different algorithms for anomaly detection in trajectory data.

Göran Falkman received the Ph.D. degree in computing science from Chalmers University of Technology, Sweden, in 2003. He holds a position as an Associate Professor in Computer Science, with a speciality in interactive knowledge systems, at the University of Skövde, Skövde, Sweden, where he works as a Researcher at the Informatics Research Centre. His current research interests include intersection of applied artificial intelligence, knowledge systems, interaction design, and information fusion. This includes work on the design, implementation, and use of formal knowledge representation and knowledge-based systems, as well as the use of interactive visualization for supporting knowledge-based reasoning processes.