An Online Adaptive Algorithm for Change Detection in Streaming Sensory Data

Yasmin Fathy, Payam Barnaghi, *Senior Member, IEEE*, and Rahim Tafazolli, *Senior Member, IEEE*

***Abstract*—There has been a keen interest in detecting abrupt sequential changes in streaming data obtained from sensors in wireless sensor networks for Internet of Things applications, such as fire/fault detection, activity recognition, and environmental monitoring. Such applications require (near) online detection of instantaneous changes. This paper proposes an online, adaptive filtering-based change detection (OFCD) algorithm. Our method is based on a convex combination of two decoupled least mean square windowed filters with differing sizes. Both filters are applied independently on data streams obtained from sensor nodes such that their convex combination parameter is employed as an indicator of abrupt changes in mean values. An extension of our method (OFCD) based on a cooperative scheme between multiple sensors (COFCD) is also presented. It provides an enhancement of both convergence and steady-state accuracy of the convex weight parameter. Our conducted experiments show that our approach can be applied in distributed networks in an online fashion. It also provides better performance and less complexity compared with the state-of-the-art on both of single and multiple sensors.**

***Index Terms*—Cooperative (diffusion-based) strategy, mean change detection, multi-sensory data, streaming data.**

I. Introduction

RECENT advances in sensing and actuator technologies in wireless sensor network (WSN) and the further evolution of the Internet of Things (IoT) paradigm enable each sensor in a network to collect large quantities of measurement and observation data streams. This empowers monitoring and detecting a wide range of real-world phenomena in areas, such as environmental monitoring, segmentation, quality control, healthcare, and smart city [1]–[3].

Manuscript received April 30, 2018; revised August 22, 2018; accepted October 6, 2018. This work was supported in part by the European Commissions Horizon 2020 for the FIESTA-IoT project (http://fiesta-iot.eu/) under Contract 643943 and in part by the EU H2020 IoTCrawler project (http://iotcrawler.eu/) under Contract 779852. *(Corresponding author: Yasmin Fathy.)*

Y. Fathy is with the Department of Computer Science, University College London, London WC1E 6BT, U.K., and also with the Institution for Communication Systems and the Department of Electronic and Electrical Engineering, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: y.fathy@ucl.ac.uk).

P. Barnaghi is with the Centre for Vision, Speech and Signal Processing and the Department of Electronic and Electrical Engineering, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: p.barnaghi@surrey.ac.uk).

R. Tafazolli is with the Institution for Communication Systems and the Department of Electronic and Electrical Engineering, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: r.tafazolli@surrey.ac.uk).

Digital Object Identifier 10.1109/JSYST.2018.2876461

Changepoint detection is a problem of identifying time points where abrupt variations in the statistical properties (e.g., mean and power) of data streams occur. It has received considerable attention and has been widely applied [4]–[6].

Changepoint detection approaches are divided into two main categories: offline and online. Offline approaches are capable of detecting sequential changepoints when accessing and processing entire data streams sequence at once. These approaches cannot detect changes in an infinite sequence of data streams in near-real time. On the other hand, online approaches identify sequential changes in continuous streaming data by processing at least p data points ahead of the actual changepoints, where p depends on the amount of data an algorithm requires to detect a change (i.e., delay) [6].

The most common approach for detecting changes of mean values in WSNs is to monitor sensor data streams published by each sensor, such as moving average, Page's cumulative sum (CUSUM) [7], exponentially weighted moving average [8] and Page Hinckley [9]. However, these approaches tend to have a high false alarm (i.e., false positive) rate [10] and do not take advantage of sharing information globally between sensors in distributed environments that might lead to better detection of abrupt changes in sensor networks [11]. Other likelihood-based and density estimation change detection approaches have also been proposed in [12] and [13]. However, such approaches cannot detect changes in data streams in an online manner. Another algorithm for detecting changes in data streams has been proposed in [14]. This paper relies on using two fixed-size windows such that changes are detected by comparing data distribution in a current window with the data distribution in a reference window. This approach is slower than using a sliding-based window [15]. A global cooperative approach between multiple sensors that record different observations of a monitored phenomenon would be desirable to provide efficient detection of instantaneous changes [16]. Distributed change detection approaches that rely on communication between neighboring sensors have been presented in [2] and [17]. These approaches avoid a single point bottleneck of transmitting observations from every sensor to a central fusion node in a sensor network by adopting the cooperative scheme. In [2], the CUSUM algorithm is computed in a distributed fashion for detecting changepoints, whereas in [17], each sensor in the network runs an independent CUSUM and once a sensor node detects a changepoint, it broadcasts to other nodes to sleep the system. However, there is no cooperation between sensors in the former, and the system

does not provide sequential change detection where the system terminates once one of the sensors detects a change in the latter.

The nature of sensor data streams that are available continuously in IoT/WSN applications demands efficient and adaptive changepoint detection methods for detecting (significant) sequential changes with the smallest possible delay [18].

In this paper, we propose a novel online, adaptive filtering-based change detection (OFCD) algorithm for the efficient and accurate detection of sequential changes in data streams published by a single sensor. The algorithm is then extended by providing a cooperative scheme between multiple sensors for updating a weight parameter that employs as an indicator for abrupt changes in mean values. Such cooperative scheme aims at detecting changes in more accurate and with minimal possible delay, as well as, less complexity.

A. Motivation

Given a set of sensors that publish streaming data in the same environment (i.e., same area and at the same time), we are interested in the following.

1) Detecting abrupt changes in sensory data streams produced by a single sensor.
2) A cooperation between multiple sensors for better performance and accuracy in detecting sequential changes in their published data streams.

Addressing these two concerns in an online fashion will allow us to monitor and detect changepoints for real-world applications continuously. Examples include identifying instantaneous changes in temperature caused by a fire or detecting activities in time series and multi-sensor wearable data for activity detection applications. To this end, we propose a novel way of detecting changepoints in a sequence of data streams.

The proposed approach is also extended to support a cooperative strategy for sharing parameter (a weight parameter λ) estimation between multiple sensors in WSN. Our distributed and cooperative algorithm asymptotically minimizes the detection delay and global false detection (false positive) rate of a particular phenomenon in distributed WSN.

We compare our algorithm with the optimal single-sided and two-sided CUSUM algorithm for a single sensor case and with RuLSIF (a changepoint detection by relative density ratio estimation) [13] for single/multi-sensory cases. The results show that our proposed algorithm outperforms the state-of-the-art in both small and significant changes in mean values and can detect the changes with minimal delay and better accuracy.

B. Outline

This paper is structured as follows. The problem formulation is explained in Section II. Section III provides the required background and the related work. Our proposed algorithm is demonstrated and discussed in Section IV. The performance evaluation, parameter settings, and reproducibility descriptions are included in Section V. Moreover, the proposed algorithm is evaluated and analyzed against the state-of-the-art approaches in Section VI. The algorithm is also evaluated on a real-world dataset (i.e., human activity use-case), which is described in Section VII. We conclude this paper and explain the future directions of our research in Section VIII. We have summarized the parameters that are used for the equations in Table I.

TABLE I
SUMMARY OF PARAMETERS

Parameter	Definition
N	Total number of sensors
n	Sensor index, $n = 1, 2, \cdots, N$
T	Total time duration
t	Time index, $t = 1, 2, \cdots, T$
k	Total number of piecewise constant segments (i.e. total number of transition time instances)
l_i	Length of a piecewise constant segment, $i = 1, 2, \cdots, k$
L	Total length of k piecewise time segments
τ_i	Transition time instance where an instantaneous change in the mean value exists, $i = 1, 2, \cdots, k$
$x(t)$	A set of data-points that are published by N sensors at a time t.
$X(n)$	A sequence of a length L data-points published by a sensor n
$x_{n,t}$	A data-point that is published by a sensor n at a time t

II. Problem Formulation

Consider a network of N sensor nodes that are placed in a monitoring region/space. At each time instance $t \geq 0$, each sensor node s_n publishes a data stream. Data streams are a sequence of numerical data points in a consecutive order. Let $x(t) \in \mathbb{R}^N$ be data points that are published by N sensors at a time t. Data streams are drawn from the Normal distribution $\mathcal{N}(\mu, \sigma^2)$ with k piecewise constant segments (i.e., observation windows). Each segment has a length l_i data points ($L = \sum_{i=1}^{k} l_i, i = 1, 2, \ldots, k$). There exists unknown transition time instances $(\tau_1, \tau_2, \ldots, \tau_k)$ at which instantaneous changes in the mean values of the Normal distribution exist. Let $X(n)$ be a sequence of a length L time-dependent data points that are published by a sensor n and τ_i is the transition time instance where $i = 1, 2, \ldots, k$

$$X(n) = [x_{n,t}, x_{n,t+1}, \ldots, x_{n,T}] \in \mathbb{R}^L. \tag{1}$$

Let $x(t)$ represent a set of data points that are published by N sensors at a time t. To this end, $x(t)$ can be represented as follows:

$$x(t) = [x_{1,t}, x_{2,t}, \ldots, x_{N,t}] \in \mathbb{R}^N. \tag{2}$$

Fig. 1 gives an illustrative example of our notations. We also show an example of data points obtained from multiple sensors in Fig. 2. The vertical lines in the figures represent instantaneous changes in the mean value. Accordingly, we address a problem of detecting changepoints in piecewise constant variation in mean values in an online fashion. In other words, our OFCD algorithm is to detect sequential changes in data streams produced by an individual sensor and its cooperative scheme (COFCD) between multiple sensors is to detect changes in data streams produced by a set of N sensors in a monitored area.

III. Background and Related Work

This section briefly discusses some of the existing works in this area and describes the background information.

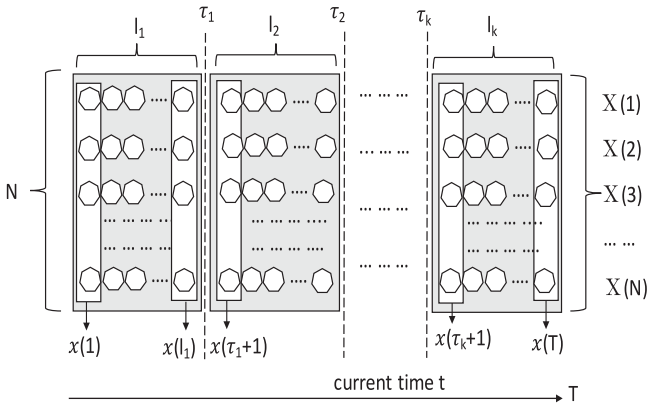

Fig. 1. Illustrative example of notations used for problem formulation.

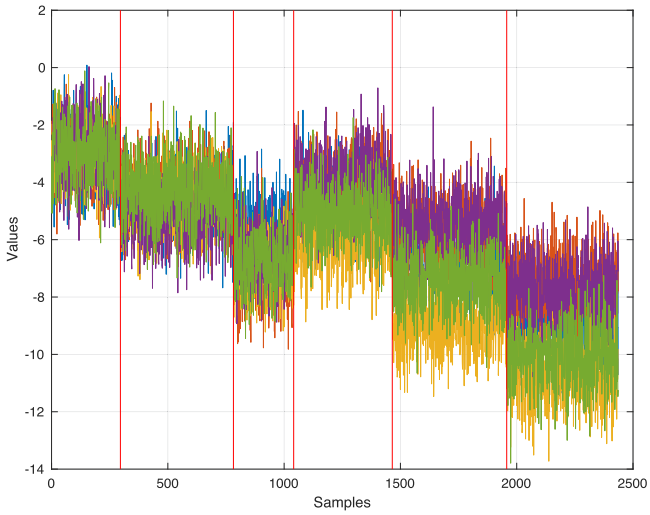

Fig. 2. Multiple data streams of five sensors with abrupt changes in the mean value.

A. Cumulative Sum (CUSUM)

The CUSUM algorithm has been well researched for detecting sequential changes in mean values [12], [18], [19]. There are different forms of the CUSUM; directed/recursive forms and (one/two)-sided forms. CUSUM has been shown to be optimal (regarding worst time delay) when the mean time between false positives (i.e., false alarms) goes to ∞ for detecting changes in mean value [18]. On the other hand, it has also been shown its optimality from the asymptomatic point of view in [20].

Following the work in [18] and [19], one-sided CUSUM has two main design parameters threshold h ($h > 0$) and a change in the magnitude of mean value δ. It has also a decision function $G(t)$ such that the detection of a changepoint at a time t relies on comparing $G(t)$ with the positive threshold h. Therefore, $G(t)$ can be formulated as follows:

$$G(t) = \max\{G(t-1) + s(t), 0\}. \tag{3}$$

Consider the changes in the mean value has μ_0 before the change and μ_1 after the same change, whereas the parameter δ takes the magnitude of the change. Therefore, instantaneous log-likelihood $s(t)$ ratio is calculated. It relies on the mean values prior and after a change at any time t and the current data point $x(t)$ that is published by a sensor

$$s(t) = \frac{\delta}{\sigma_t^2}\left(x(t) - \mu_0 - \frac{\delta}{2}\right) \tag{4}$$

where σ^2 is the constant variance of the data points so far. In this case, the CUSUM $S(t)$ can be defined through instantaneous log-likelihood $s(t)$ (4)

$$S(t) = S(t-1) + s(t). \tag{5}$$

One-sided CUSUM detects the change in only one direction (i.e., increase or decrease) for all data points. However, most of the applications such as automatic segmentation and detecting activities require detecting changes in both directions. Two one-sided CUSUM have been proposed in [7]. In this case, two decision functions ($G(t)^i$ and $G(t)^d$) should be compared with the positive threshold h. In addition, two instantaneous log-likelihood functions are used ($s(t)^i$ and $s(t)^d$), where i and d are the increase and decrease in the mean value, respectively

$$s(t)^i = s(t) \tag{6}$$

$$s(t)^d = -s(t). \tag{7}$$

Using the aforementioned equations and (4), $s(t)^i$ and $s(t)^d$ are formulated as follows:

$$s(t)^i = \frac{\delta}{\sigma_t^2}\left(x(t) - \mu_0 - \frac{\delta}{2}\right) \tag{8}$$

$$s(t)^d = -\frac{\delta}{\sigma_t^2}\left(x(t) - \mu_0 + \frac{\delta}{2}\right). \tag{9}$$

In this case, the decision functions $G(t)^i$ and $G(t)^d$ have similar formulas similar to (3) where

$$G(t)^i > h \text{ and } G(t)^d > h. \tag{10}$$

CUSUM has been used for detecting changes in different applications, such as surveillance, security, quality control, and power system applications [21]. Furthermore, two-sided CUSUM has been effectively applied to fault detection in power system applications. However, the CUSUM has a drift parameter δ that is a prior constant variable which is initialized once, and the algorithm continues to use its value all the time [21]. Such a global variable and a single calculation might limit the applicability of CUSUM for monitoring streaming sensory data. For more details about the CUSUM and its variants, we refer the readers to [12] and [19].

B. Least-Mean-Square (LMS) Algorithm and Adaptive Filters

LMS is a de facto adaptive filtering algorithm among others that has a set of filtering coefficients (i.e., weights) that are estimated continuously to minimize the least mean squared error (i.e., the difference between the desired and output/estimated data points). LMS has a low computational overhead. It relies on a stochastic gradient descent approach in which coefficients are updated iteratively to minimize the least mean squared error $e(t)$ of the filter at the current time t

$$e(t) = x(t) - y(t) \tag{11}$$

where $x(t)$ is a data point at time t and $y(t)$ is the output of applying an adaptive filter on $x(t)$ such that

$$y(t) = w(t)x(t) \tag{12}$$

where $w(t)$ is the adaptive filter weight for minimizing the error $e(t)$ with a step size α (i.e., learning rate) using the standard LMS rule

$$w(t) = w(t-1) + \alpha e(t)x(t). \tag{13}$$

Combination scheme of two filters instead of using one filter has been investigated to improve the steady-state characteristics and performance of LMS [22]. Following the work in [22]–[24], a convex combination employs two filters that are decoupled and simultaneously applied to the same input. Their weights are adjusted to minimize the overall errors of the filters. To this end, a convex combination scheme is used to combine the weights of the two filters using a parameter $\lambda(t)$. $\lambda(t)$ is a mixing scalar parameter ($0 \leqslant \lambda(t) \leqslant 1$) to preserve the convexity of this combination [24]. In this case, the overall weight $w(t)$, which is the mixture filter weight, is represented as follows:

$$w(t) = \lambda(t)w_1(t) + [1-\lambda(t)]w_2(t) \tag{14}$$

where $w_1(t)$ and $w_2(t)$ are the weights of the first filter and the second filter at a time instant t, respectively. $\lambda(t)$ is updated as a convex combination parameter with a step size of α using the standard LMS adaptation rule [25] similar to (13) as follows:

$$\lambda(t+1) = \lambda(t) + \alpha e(t)x(t). \tag{15}$$

Adaptive combination of two filters (i.e., fast and slow) have been used in plant identification applications, where $w_1(t)$ is a fast filter and $w_2(t)$ is a slow filter [23]. In such a case, λ is near 1 for high tracking situations and is near 0 for slow tracking [see (14)]. The scheme has been improved and refined in [26] to enable more robust results by transferring the coefficients of the faster LMS filter to the coefficients of the slower LMS filter to accelerate its convergence. Another convex combination scheme is to use two filters independently in which one different step size is used for each filter [27]. It is worth noting that the idea of a combinational scheme of two filters is not new. It has been proposed earlier in [28]. However, the scheme is composed of one filter that has an adaptive weight, whereas the other filter has a fixed weight. If the former performs better than the latter, the latter's weights are updated by the adaptive weights.

More recently, in consonance with the idea of the convex combination of two filters, a change detection approach based on a combination of two models have been proposed in [2]. The approach relies on long-term (LT) memory and short-term (ST) memory models with the aim of detecting changepoints based on using a growing window for LT model and fixed window size for ST model to obtain a better change detection using the collaboration between the two models.

It is worth noting that there are other types of methods for change detection approaches, such as supervised learning approaches that are based on labeled training data. If the number of states is specified, the learning approach is trained to find each state boundary, and consequently, detects the changepoint [6].

Algorithm 1: Online, adaptive Filtering-based Change Detection (OFCD).

Input : Input signal X
Initialisation of w_s, w_f, α, h, λ
Output: time instances $(\tau_1, \tau_2, \cdots, \tau_T)$

while $t < T$ **do**
  $\hat{y}_f \leftarrow \frac{1}{w_f} \sum_{i=t-w_f}^{t} X(t)$
  $\hat{y}_s \leftarrow \frac{1}{w_s} \sum_{i=t-w_s}^{t} X(t)$
  $\hat{y}(t) \leftarrow \lambda(t)\ \hat{y}_f + [1-\lambda(t)]\ \hat{y}_s$
  $e(t) \leftarrow X(t) - \hat{y}(t)$
  $\lambda(t+1) \leftarrow \lambda(t) + \alpha\ e(t)\ [\hat{y}_f - \hat{y}_s]$
  if $\lambda(t+1) > h$ **then**
    $\tau_t \leftarrow 1$
    $\lambda(t+1) \leftarrow 0$
  end
  $t \leftarrow t+1$
end

IV. Proposed Algorithm

In this paper, we use the concept of an adaptive LMS algorithm, which is based on using a convex combination of two adaptive filters. The adaptive filters are so-called "fast" and "slow" filters according to the speed of convergence of the filters when transient changes in the mean values occur. The convex combination of the two filters is to find the best fit of their linear models to an input streaming sensory data where the abrupt changes in the mean values are unknown. To this end, the aim is to achieve a better steady-state performance and keep monitoring the sensory data streams continuously to detect instantaneous changepoints in the mean values in an online fashion.

Both filters are applied separately on data streams that are produced by a set of sensor nodes. However, in contrast to the existing solutions, there is a cooperation between neighboring sensor nodes for estimating the convex combination parameter of both filters. The parameter employs as an indicator for detecting sequential changes in mean values, which is the key difference here compared with the slow/fast filters explained in the previous section. We also use the moving average estimator for LMS filters.

Given the input data streams $X(t), X(t+1), \ldots, X(T)$ (as explained previously in Section II), the output of our proposed approach is to find the unknown transition time instances $(\tau_1, \tau_2, \ldots)$ at which instantaneous changes in the mean values exist, with minimal possible delay.

Our proposed algorithm: OFCD is summarized in Algorithm 1. The main idea is to detect changepoints in mean values as fast and accurate as possible.

We adopt using a convex combination of two LMS adaptive filters and a moving average. The two component filters are decoupled and have different window sizes. One of the adaptive filters is fast, whereas the other is slow. The fast filter has a short-term observation memory based on using a relatively small fixed window size w_f, whereas the slow has a long-term observation memory based on an increasingly large window size w_s.

The fast filter has higher tracking capabilities that allow detecting changepoints in streaming sensory data with fast changes. On the other hand, the slow filter has a better steady-state that minimizes the detection error. The advantage of the convex combination of both filters is retained such that fast filter provides a fast convergence, whereas the slow filter provides a better steady-state through minimizing the mean squared error.

Running moving average with a fixed window $\hat{y}_f$ and an increasing window $\hat{y}_s$ for fast and slow adaptive filters, respectively, is to have a good combination for estimating the next observation value based on the previous set of values (i.e., window size). To this end, the outputs of the fast $\hat{y}_f$ and slow $\hat{y}_s$ filters are as follows:

$$\hat{y}_f = \frac{1}{w_f} \sum_{i=t-w_f}^{t} X(t) \tag{16}$$

$$\hat{y}_s = \frac{1}{w_s} \sum_{i=t-w_s}^{t} X(t) \tag{17}$$

where w_f is a fixed window size for fast filter and w_s is an increasing window size for slow filter such that $w_f < w_s$. Similar to [23], the overall output for filters $\hat{y}(t)$ is a convex combination of the outputs of both filters mentioned before

$$\hat{y}(t) = \lambda(t)\ \hat{y}_f + [1 - \lambda(t)]\ \hat{y}_s \tag{18}$$

$$e(t) = [X(t) - \hat{y}(t-1)] \tag{19}$$

where the mixing parameter λ of their combination is adaptively updated in an online manner that aims at minimizing the overall filters error $e(t)$ between the desired signal $X(t)$ and overall output of both filters $\hat{y}(t-1)$. In addition, λ is considered as a forgetting factor that determines how to treat streaming data by giving more weight to the recent streaming sensory data and down-weighting (i.e., forgetting) earlier observations.

The motivation of our proposed approach is to extract the best properties of the decoupled fast $\hat{y}_f$ and slow $\hat{y}_s$ filters by assigning and updating λ that is a combination of both filters by an appropriate value at time t.

The fast filter with a small fixed window size w_f can track the quick transition time instances. However, the fast filter cannot provide a steady-state performance for the detection of long-term trends, and consequently, it will not be very accurate for long-term observations. On the other hand, the slow filter with a large growing window size provides a stable steady-state observation. Therefore, the convex combined weight parameter λ is adaptively estimated at each time t such that it can detect an abrupt changepoint and consequently the fast filter (with a small window size) can provide an optimal solution with the aim to minimize the mean squared error

$$\lambda(t+1) = \lambda(t) + \alpha\ e(t)\ [\hat{y}_f - \hat{y}_s] \tag{20}$$

where α is the learning rate parameter. The learning rate influences the stability and the convergence of the model. It was noted that LMS filters do not converge if $\alpha > 1.0$ [29]. The mixing parameter λ is initialized by zero. While monitoring streaming data, the $\lambda(t)$ is normalized to be independent of the data streams scale. Therefore, $\lambda(t)$ is guaranteed to be within an interval of $[0, 1]$. It is compared with a threshold h such that the updated weight λ is mapped to an output of a decision functions $s(t)$ as follows:

$$s(t) = \begin{cases} 0,\ \lambda(t) < h \\ 1,\ \lambda(t) \geq h \end{cases} \tag{21}$$

where $s(t) = 1$ corresponds to detect a change in streaming data and $s(t) = 0$ otherwise. If λ is above a threshold h, which is prior information that depends on the problem definition, this indicates that a detection alarm occurs and that a changepoint is detected (τ_t) at the time t. The algorithm resets after a change occurs such that λ reinitializes to zero after the detection to allow the algorithm to forget all the old information instantaneously and to get a fresh start.

The advantage of convex combination is to minimize the excess mean squared error (EMSE) of the overall filter $e(t)$ comparing to the EMSE for both filters $e_1(t), e_2(t)$ as discussed in [22] such that

$$e(t) \leq \min[e_1(t), e_2(t)]. \tag{22}$$

We extend our algorithm by using cooperation between multiple sensors. The extended approach called cooperative online, adaptive filtering based change detection (COFCD) (see Algorithm 2) is based on a diffusion cooperation scheme in which neighboring sensor nodes can communicate and share information between each other. At each node, an estimation of λ value is calculated and fused to other nodes. Their local estimated values are then fed into the local adaptive filters with the aim to have a shared estimator ψ between neighboring sensors for improving detection of instantaneous changepoints. The updates of λ for each sensor node is based on the combine-then-adapt (CTA) approach [30]. To this end, $\lambda(t)$ is updated by $\psi(t)$.

Algorithm 2: Cooperative Online, adaptive Filtering-based Change Detection (COFCD).

Input : Input signal X
Initialisation of w_s, w_f, α, h, λ
Output: time instances $(\tau_1, \tau_2, \cdots, \tau_T)$

while $t < T$ **do**
  for $n = 1, 2, ..N$ **do**
    $\hat{y}_{f,n} \leftarrow \frac{1}{w_f} \sum_{i=t-w_f}^{t} X(t)$
    $\hat{y}_{s,n} \leftarrow \frac{1}{w_s} \sum_{i=t-w_s}^{t} X(t)$
    $\hat{y}_n(t) \leftarrow \psi(t)\ \hat{y}_{f,n} + [1 - \psi(t)]\ \hat{y}_{s,n}$
    $e_n(t) \leftarrow X(t) - \hat{y}_n(t)$
    $\lambda_n(t+1) \leftarrow \psi(t) + \alpha\ e_n(t)\ [\hat{y}_{f,n} - \hat{y}_{s,n}]$
  end
  $\psi(t+1) = \frac{1}{N} \sum_{n=1}^{N} \lambda_n(t+1)$
  if $\psi(t+1) > h$ **then**
    $\tau_t \leftarrow 1$
    $\psi(t+1) \leftarrow 0$
  end
  $t \leftarrow t + 1$
end

Algorithm 3: Generate piecewise constant variation in mean value.

Input : s_1, s_2, μ_1, μ_2, d_1, d_2, S, N, ρ, $\sigma^2 = 1$
Output: $\{\tau_1, \tau_2, \cdots, \tau_S\}$, $\{X(1), X(2), \cdots, X(N)\}$
while $s < S$ **do**
  Select a segment length uniformly between $[s_1, s_2]$
  Select μ uniformly between $[\mu_1, \mu_2]$
  Select a scale for μ uniformly {increase or decrease}
  Select a magnitude of the scale uniformly between $[d_1, d_2]$
  Generate a segment for each n, $\{n \in N\}$ drawn from $\mathcal{N}(\mu, \sigma^2)$ with selected μ, scale and magnitude
  Mark the transition time instance τ_s of the current piecewise constant mean
  $s \leftarrow s + 1$
end
Combine S segments of each sensor n to generate $X(n)$
Apply Cholesky factorization on $\{X(1), X(2), \cdots, X(N)\}$ if $\rho > 0$

The following equations describe the CTA diffusion scheme

$$\begin{cases} \psi(t) = \frac{1}{N} \sum_{n=1}^{N} \lambda_n(t) \text{ (diffusion step)} \\ \lambda_n(t+1) \leftarrow \psi(t) + \alpha\, e_n(t)\, [\hat{y}_{f,n} - \hat{y}_{s,n}] \text{ (incremental step)} \end{cases} \tag{23}$$

where n is a sensor index and each sensor has λ_n that is updated adaptively (in the incremental step) based on cooperation between sensors to estimate ψ (in diffusion step). It is worth noting that the extended algorithm assumes that all sensors observe the same phenomena.

It is worth mentioning that we consider a detection alarm is correct if there is a true alarm at step t at which $t \in [t, t+N]$, where N is considered the maximum number of samples that might cause a time delay (i.e., detection latency) before detecting a true changepoint. Furthermore, duplication in detection alarm might occur. Similar to [13], we remove ith detection alarm such that $\{t_i - t_{i-1}\} < 20$ (verified experimentally). Both of these steps depend on the time that is needed for estimating parameters to detect a change in mean value and for a slow filter to converge after a change has occurred. Since the convex combination parameter of the adaptive filters employs as an indicator for a detection alarm for abrupt changes, a threshold parameter h has to be initialized beforehand to filter out all alarms whose convex parameter is $\psi(t) \leq h$.

V. Evaluation

As discussed in the previous section, the proposed solution is composed of a convex combination of two adaptive filters relying on diffusion and cooperative approach. The experiments are conducted by first evaluating our algorithm (OFCD) on univariate (single sensor) data. We then evaluate (COFCD) on a multi-sensor use case. We compare our algorithm and its extension to the following approaches in the state-of-the-art.

1) *Baseline 1 (B 1):* One-sided CUSUM recursive form ([19, Algorithm 3]).
2) *Baseline 2 (B 2):* Two-sided CUSUM ([19, Algorithm 5]) with a fixed (i.e., a short term) and an increasing (i.e., a long term) observation window sizes.
3) *RuLSIF:* Changepoint detection by relative density-ratio estimation, which was proposed in [13].

Our motivation to compare our proposed algorithm with these baselines is that baseline 1 calculates the cumulative sum recursively and efficiently, which makes it suitable for online applications [19]. However, one-sided CUSUM can only detect the changes in one direction (as discussed earlier). To this end, we have conducted some experiments in which the change in mean values is only in one direction. Furthermore, other experiments are carried out in which the changes in mean are in both directions. Therefore, we compare our algorithm in the latter case with baseline 2 (two-sided algorithm) to show the usefulness of our proposed approach. Two-sided CUSUM can be implemented using either a fixed window or an increasing observation window. We have implemented our algorithm to support both window size setting methods and compare it with two-sided CUSUM. It worth noting that different observation windows are applied to the slow filter in our algorithm because the fast filter typically observes the fast changes in data streams and consequently it is not suitable to have a long-term observation window.

Some other methods and approaches use density functions to detect changes [12], [31]. RuLSIF approach is proposed in [31]. It is a statistical approach for detecting changes based on Pearson divergence that is estimated by a method of a direct density-ratio estimation. RuLSIF code is available via http://www.ism.ac.jp/∼liu/software.html, which makes it more convenient for the performance comparison.

A. Dataset

In our experiments, we assume that multiple sensors observe the same phenomena. Therefore, we consider having a set of sensors N in which we generate a sequence of data points for each sensor. The data points are drawn from a Normal distribution with the same piecewise constant mean and a constant global variance σ^2. In other words, a mean value is selected uniformly between $[\mu_1, \mu_2]$ per segment across N sensors. The number of samples per segment (i.e., segment's length) is selected uniformly between the interval $[s_1, s_2]$ and the mean for each segment has a scale of increase or decrease that is selected uniformly at which its magnitude is also selected uniformly between the interval $[d_1, d_2]$. The scale of a segment is selected according to the mean of the previous segment. This will create the same transition time instances $(\tau_1, \tau_2, \ldots, \tau_S)$ across the sensors at which instantaneous changes in the mean values of the Normal distribution exist.

We also consider that streaming sensory data for multiple sensors can be generated with a different inter-sensor correlation ρ between sensors. We use the Cholesky factorization [32] to generate such streaming data. A summary of how we generate piecewise constant variation in mean values for N sensors is demonstrated in Algorithm 3 and we have summarized the parameters used in the algorithm in Table II.

TABLE II
SUMMARY OF PARAMETERS

Parameter	Definition
s_1	is the lowest number of samples in a segment
s_2	is the highest number of samples in a segment
μ_1	is the lowest value of the mean value
μ_2	is the highest value of the mean value
d_1	is the lowest magnitude of the mean scale
d_2	is the highest magnitude of the mean scale
N	is the total number of sensors (channels)
n	is the sensor index, $n = 1, 2, \cdots, N$
S	is the total number of transition time instances (i.e. total number of segments)
ρ	inter-sensor correlation between sensors

The following are the specific values that we have used for each of the parameters.

1) Range of each segment's length $[s_1, s_2] \leftarrow [100, 500]$.
2) Mean value $[\mu_1, \mu_2] \leftarrow [-3, 3]$.
3) Magnitude of the scale $[d_1, d_2] \leftarrow [1, 3]$.
4) Number of transition time instances (i.e., change points) $S = 10$.

It is worth mentioning that we have used different values of ρ and N during the simulation experiments. Therefore, we have mentioned the specific values that we have used in each experiment in the following sections.

B. Performance Evaluation

The performance evaluation is based on the following criteria.

1) False positive rate (FPR): It is the percentage where the algorithm detects changepoints that actually not exist

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \tag{24}$$

2) False negative rate (FNR): It is the percentage where the algorithm fails to detect changepoints that actually exist

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}. \tag{25}$$

3) Detection latency (L): It is the number of samples that is required to detect a true changepoint that has occurred. Obviously, by dividing the number of samples by the sampling frequency, we can also obtain the latency time in seconds.

In (24) and (25), TN is the number of true negative changepoints, FP is the number of false positive changepoints, FN is the number of false negative changepoints, and TP is the number of true positive changepoints.

C. Parameter Settings and Reproducibility

We have considered two kinds of experiments for detecting instantaneous changes in streaming sensory data while conducting the evaluations. It is worth mentioning that the results are the average of 1000 independent trials.

1) *Case 1:* Univariate streaming data (for a single sensor) at which a scale of μ is at one direction (e.g., increase), and at both directions (i.e., increase and/or decrease). Results of applying the OFCD (see Algorithm 1) algorithm are reported in Table III for one direction and in Table IV for both directions.
2) *Case 2:* Multivariate streaming data (from N sensors) at which a scale of μ has both directions and with a cooperation-based diffusion strategy (COFCD: Algorithm 2). Results are reported in Tables V and VI.

TABLE III
CASE 1 (a): SINGLE SENSOR RESULTS

		Baseline 2			OFCD	
	B 1	(a)	(b)	RuLSIF	(a)	(b)
FPR (%)	0.009	0.01	0.01	4	0.006	0.004
FNR (%)	2	23	3	0	7	0.5
Latency	7	23	4	26	14	7

TABLE IV
CASE 1 (b): SINGLE SENSOR RESULTS

	Baseline 2			OFCD	
	(a)	(b)	RuLSIF	(a)	(b)
FPR (%)	0.01	0.01	4	0.005	0.004
FNR (%)	23	2	20	7	0.5
Latency	21	7	25	14	7

TABLE V
CASE 2: 10 SENSOR RESULTS WITH $\rho = 0$

		COFCD	
	RuLSIF	(a)	(b)
FPR (%)	3	0	0
FNR (%)	1	0.2	0
Latency	21	8	7

TABLE VI
CASE 2: 10 SENSOR RESULTS WITH $\rho = 0.5$

		COFCD	
	RuLSIF	(a)	(b)
FPR (%)	4	0	0
FNR (%)	22	0	0
Latency	25	2	2

The first set of simulation results (Case 1) considers a univariate case $X(n)$, which is a sequence of data points of a sensor n, and the data are drawn from a Normal distribution $\mathcal{N}(\mu, 1)$ with a set of transition time instances $(\tau_1, \tau_2, \ldots)$ at which instantaneous changes in the mean values of Normal distribution exist. Using Algorithm 3, we generate streaming sensory data to represent a single sensor ($N = 1$). Because there is a single sensor in this set of experiments, the ρ value does not have any effect.

It is worth noting that detection threshold, the size of windows, and learning rate parameters are highly dependent on the characteristics of streaming data. To this end, these parameters might need fine-tuning based on the application domain. Therefore, we will discuss the sensitivity analysis of the performance of our algorithm on different choices later. For this set of experiments, we have used specific values ($h = 8, d = 2, w = 50$) for

baselines 1 and 2 and ($h = 0.6, \alpha = 0.1, w_s = 50, w_f = 4$) for our algorithm (OFCD). However, for RuLSIF, we have used the recommended values ($n = 50$, $k = 10$, $\alpha = 0.01$) [13].

The second set of simulation results (Case 2) aims to analyze the performance of our extended algorithm (COFCD) that applies collaborative adaptive filtering approach for detecting changepoints in a fully connected network of N sensors that produced data streams $\{X(1), X(2), \ldots, X(N)\}$ with an assumption that sensors observe the same piecewise constant variation in the mean values. Similar to the first set of simulations, we have used Algorithm 3 to generate streaming sensory data for multiple sensors ($N > 1$). In this set of results, we assume $N = 10$ and other parameters have the same values similar to previous simulation results.

To ensure the reproducibility of our results, we have made the code and datasets of our implementation and baselines available and have also provided details of a configurable experimental setup at http://github.com/YasminFathy/Adaptive-Filtering-Based-ChangeDetection.

VI. Results and Discussion

We evaluate the performance of the OFCD algorithm against baselines 1 and 2. Table III demonstrates the performance evaluation for detecting sequential changes in one direction (i.e., increase) data streams produced by an individual sensor. Similarly, Table IV demonstrates the same result, but for detecting instantaneous changes in both directions.

In Table III, OFCD (case b) offers better accuracy in terms of false positive and false negative rates than one-sided CUSUM (baseline 1: B1). Both of the algorithms have the same number of samples for latency detection (i.e., seven samples). Although, two-sided CUSUM (baseline 2) in case b has better latency than OFCD (case b), it has more false negative and false positive rates 6 times and 2.5 times, respectively. On the other hand, OFCD (case a) performs better than baseline 2 (case a) in which a fixed observation window size is considered in all evaluation criteria (i.e., FPR, FNR, and latency). RuLSIF provides 0% for FNR; however, it requires the highest number of samples to detect changepoints compared to all other algorithms. We believe that RuLSIF requires more time to detect changes, since it relies on a cross-validation mechanism for model selection, which tends to add more complexity and requires more time. Overall, it is evident in the Table III that OFCD with an increasing observation window for the slow filter (case b) outperforms all other approaches and its fixed observation window (case a) performs better than baseline 2 (case a).

Table IV shows that OFCD (case b) outperforms all other algorithms. OFCD (case a) provides a better result than baseline 2 (case a). When the scale of the mean values are in both directions, RuLSIF provides higher FNR and detection latency than OFCD (cases a and b) and baseline 2 (case b). It is worth mentioning that the experimental results of RuLSIF in [13] show that it outperforms other change-detection algorithms.

The second set of results (case 2) is included in Tables V and VI where we assume having a fully connected network of ten sensors with inter-sensor correlations $\rho = \{0, 0.5\}$. The comparison includes our algorithm COFCD that provides collaborative adaptive filtering method between $N = 10$ sensors for detecting changes against the RuLSIF that supports detecting changes in multi-sensor data.

In these results, COFCD outperforms RuLSIF in false positive, false negative, and detection latency criteria. Moreover, RuLSIF requires more (approximately three times) number of samples to detect sequential change points compared with COFCD that requires only seven samples (case b) for detecting the same changepoints (see Table V). COFCD offers better accuracy in terms of false positive and false negative rates compared to RuLSIF.

On the other hand, RuLSIF performs better when there is no correlation (see Table V) between data streams produced by a multiple sensors set than if there is a correlation. Generally, RuLSIF requires running on the full data in an offline manner because it splits the data into a set of windows as an initial step before running the algorithm. It also relies on a cross-validation mechanism for model selection, which tends to add more complexity and requires more time. Another drawback of RuLSIF is that it requires running the algorithm in two directions; forward and backward. The algorithm runs from the beginning of the streaming data until the end and then starts in the reverse order (i.e., from the end of the streaming data to the beginning). The detected changepoints are based on the accumulative values of density functions for both of the directions.

COFCD performs slightly better when there is a correlation in streaming sensory data produced by multiple sensors (see Table VI) compared with the situations that there are no correlations (see Table V). Our proposed algorithm OFCD and its extension COFCD perform better than the state of the art for sequential detection in data streams produced by an individual and multiple sensors, respectively. There are two main and interesting observations from the last set of results. The first is that COFCD in Table VI has the same performance with a short-term (case a) and long-term (case b) observation windows when there is an inter-sensor correlation ($\rho > 0$). The second observation is that the results of COFCD in Table V have a quite different behavior compared to the first observation. We have conducted extensive experiments to study the behavior of our algorithm (i.e., FPR, FNR, and average latency) with different ρ values.

The experiments are done with different number of sensors $N = \{2, 3, \ldots, 10\}$ and inter-sensor correlation values $\rho = \{0, 0.1, 0.3, 0.5\}$ for COFCD cases (a and b). Figs. 3–5 demonstrate the behavior of COFCD (case b) with differing inter-sensor correlation values between multiple sensors. It is clear that COFCD performs better in terms of FPR and FNR, as shown in Figs. 3 and 4, respectively, regarding ρ value. On the other hand, if the number of sensors $N < 4$, FNRs decrease with higher ρ values (see Fig. 3). However, it is a contradiction to Fig. 4 where FPRs increase with higher ρ values. It is sometimes the case that reducing FNRs come at the expense of increasing FPRs. Moreover, the algorithm can detect the changepoints faster when ρ values are higher (see Fig. 5). Fig. 6 gives a closer look into how the average detection latency is influenced by different N and ρ values. The algorithm detects instantaneous changepoints faster while the number of sensors

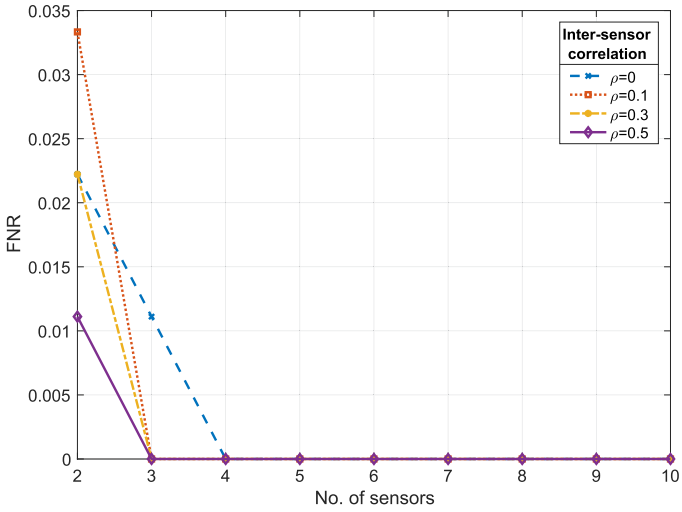

Fig. 3. False negative rate of different number of sensors with inter-sensor correlation values ($\rho = \{0, 0.1, 0.3, 0.5\}$).

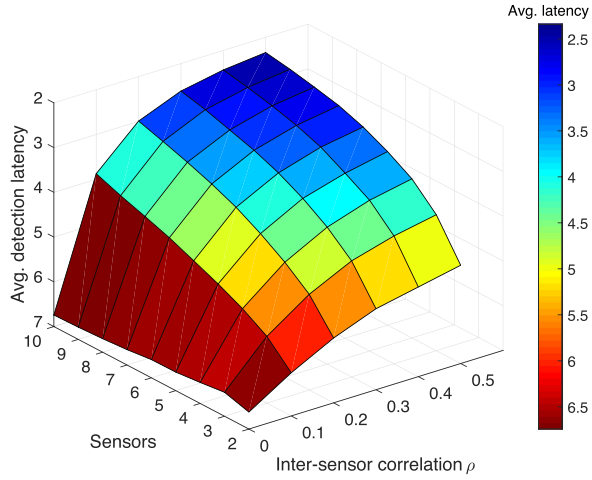

Fig. 6. Average detection latency (i.e., number of samples) for different number of sensors using different inter-sensor correlation values.

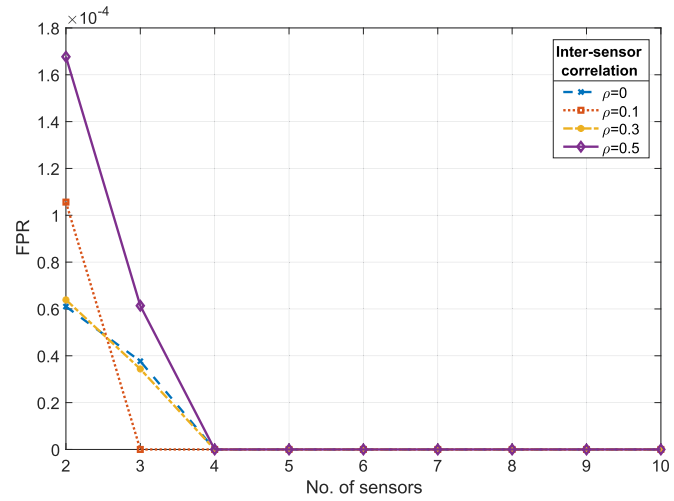

Fig. 4. False positive rate of different number of sensors with inter-sensor correlation values ($\rho = \{0, 0.1, 0.3, 0.5\}$).

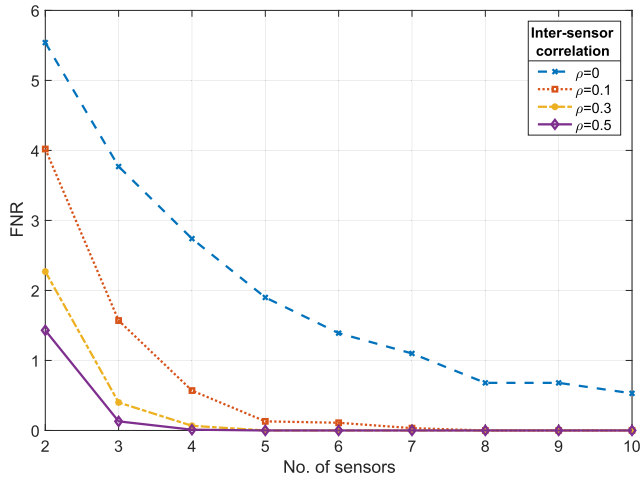

Fig. 7. False negative rate of different number of sensors with inter-sensor correlation values ($\rho = \{0, 0.1, 0.3, 0.5\}$).

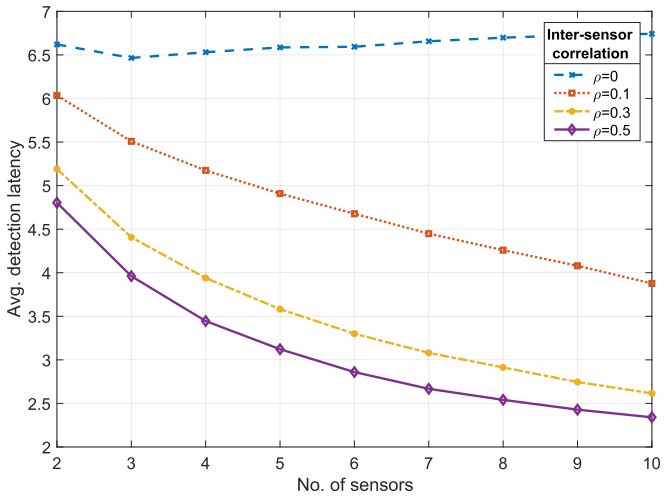

Fig. 5. Average detection latency of different number of sensors with inter-sensor correlation values ($\rho = \{0, 0.1, 0.3, 0.5\}$).

N increases. This shows how collaborative adaptive filtering strategy performs better when multiple sensors cooperate for updating a weight parameter that employs as an indicator for abrupt changes in the mean values. Moreover, COFCD can detect changes between highly correlated data streams ($\rho = 0.5$).

Similarly, we have studied the behavior of COFCD (case a). Figs. 7–9 demonstrate the behavior of COFCD (case a) with differing inter-sensor correlation values between multiple sensors. COFCD (case a) has quite similar behavior to (case b). For example, FNRs decrease with higher ρ values when $N < 4$ (see Fig. 7). However, FPRs increase with higher ρ values (see Fig. 8). In addition, with highly correlated data streams produced from multiple sensors, COFCD (case a) detects changepoints faster (see Fig. 9). Overall, COFCD (case a) has higher false positive and false negative rates than (case b). In addition, the former detects abrupt changes slower than the latter.

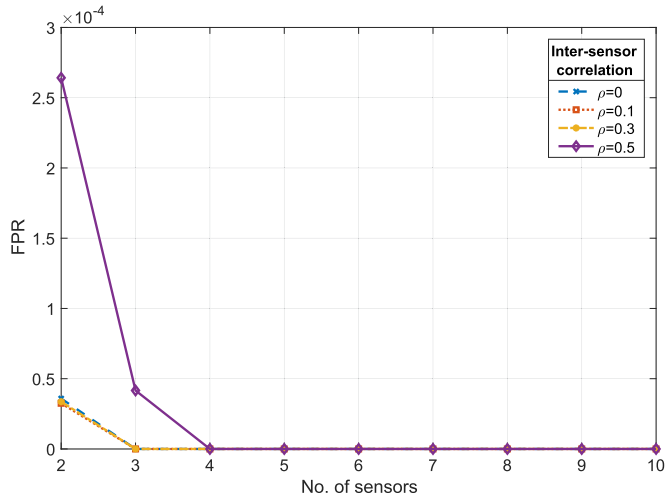

Fig. 8. False positive rate of different number of sensors with inter-sensor correlation values ($\rho = \{0, 0.1, 0.3, 0.5\}$).

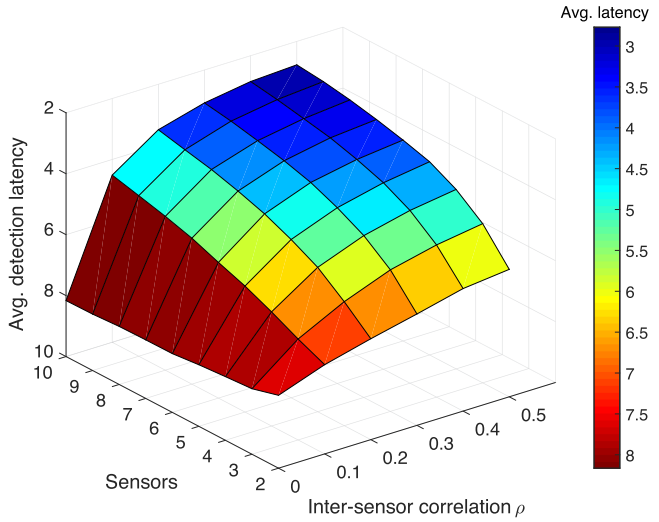

Fig. 9. Average detection latency (i.e., no. of samples) for different number of sensors using different inter-sensor correlation values.

A. Sensitivity Analysis

We have investigated the sensitivity of the different choices for COFCD (case b). In this section, we discuss the sensitivity analysis by varying the key parameters and show their impact on the stability and the accuracy of our proposed algorithms.

During our empirical experimentation, we have investigated the parameters that affect our results the most. The following parameters affect the behavior of the algorithm.

1) Size of windows (e.g., w_s and w_f for slow and fast filters, respectively, such that $w_f < w_s$).
2) Learning rate α (it has often 0.1 value; verified experimentally).

It is worth noting that the adjustment of such parameters is also application dependent.

1) Window Sizes: The length of the sliding window for fast filter w_f should be long enough to achieve a fast convergence and a good tracking while fast changes are taking place; however, it should not be too long in order to detect fast abrupt changes. On the other hand, the slow filter w_s should provide a good approximation for detecting slow changes. As mentioned earlier, the simulation results are provided with $w_s = 50$ samples and $w_f = 4$ samples. Tables VII and VIII show the effect of slow filter (w_s) and fast filter (w_f) with differing window sizes in the performance of the algorithm. From the tables, we can see that if the length of the slow filter w_s is too long, which is approximately larger than the number of samples between two consecutive abrupt changes, there will be a poor approximation of the weight parameter and consequently the number of false alarms may increase. On the other hand, increasing the length of the fast filter such that $w_f < w_s$ adds more delay in detecting sequential changepoints.

TABLE VII
COFCD (b): DIFFERENT WINDOW SIZES FOR SLOW FILTER w_s WITH $N = 10$ AND $\rho = 0$

Window sizes	FPR (%)	FNR (%)	Latency
$w_s = 150, w_f = 4$	0.04	5	9
$w_s = 200, w_f = 4$	0.04	18	9
$w_s = 250, w_f = 4$	0.06	31	10

TABLE VIII
COFCD (b): DIFFERENT WINDOW SIZES FOR FAST FILTER f_s WITH $N = 10$ AND $\rho = 0$

Window sizes	FPR (%)	FNR (%)	Latency
$w_f = 10, w_s = 50$	0	0	7
$w_f = 20, w_s = 50$	0	0	10
$w_f = 30, w_s = 50$	0	0	12

2) Learning Rate: Learning rate (i.e., step size α) is an adaptive step for the convex combination parameter that combines the higher tracking capabilities of the fast filter (w_f) with a better steady-state performance of slow filter (w_s). α affects the convergence and the accuracy of the algorithm. In other words, it controls how to move fast/slow toward obtaining optimal values of the mixing parameter λ. If α is too large, the algorithm could skip the optimal weight for the mixing parameter and if it is too small, it could misadjust the mixing parameter λ that affects the stability of the algorithm. Small α allows the convergence of λ with some values that might be near the optimal values. A suitable value for α provides a faster convergence for the adaptive filters as well as a good accuracy. When $\alpha = 1.0$, the adaptive filters cannot converge [29]. During our experiments, we noticed that our algorithm has a stable performance when $\alpha = 0.1$.

We have shown how different learning rate values have a direct effect on the performance of our algorithm in Fig. 10. The experimental results are obtained with different learning rates $\alpha = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ while other parameters have the same values as other experiments ($N = 10, \rho = 0, S = 10, w_s = 50, w_f = 4$). If α is zero, the weight of the mixing parameter will still be near where it was initialized. Therefore, FNR is almost 100%, and consequently, FPR is 0%. The FNR drops to 0% when $\alpha = 0.1$, whereas FPR starts to rise when α is roughly 0.2. We believe that the algorithm is stable when $\alpha \leq 0.1$ and this means that with only 10%

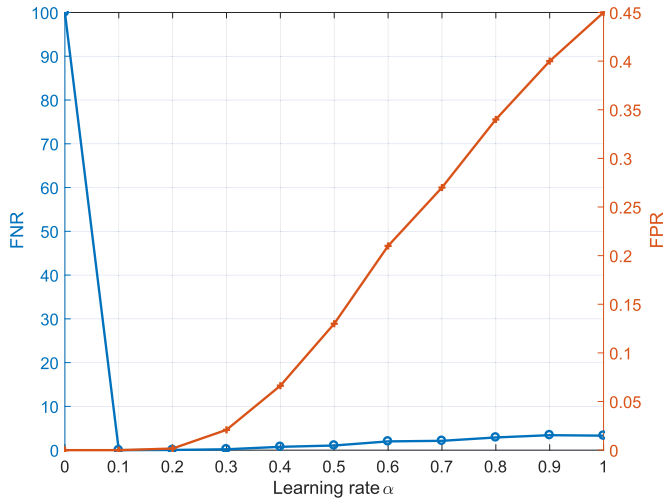

Fig. 10. False positive rates on the right-hand side and false negative rates on the left-hand side with different learning rates ($\alpha = \{0, 0.1, 0.2, \ldots, 1\}$).

of recent values and 90% of old values, the algorithm can have a good performance for detecting changepoints. A smaller step size α of adaptive filters is required to preserve its stability [33].

It should be noted that although our cooperative scheme has shown a better performance for detecting changes when multiple sensors cooperate for updating a weight parameter that employs as an indicator for abrupt changes, the accuracy of our approach might be affected if some of these sensors are out of battery or disconnected from the network. In such a case, the weight parameter is not correctly estimated, and consequently, this degrades the performance and accuracy of our algorithm.

VII. Use Cases: Human Activity

In this section, we show and analyze our algorithms against RuLSIF on a real-world sensors activity dataset [34]. The dataset is available at http://ps.cs.utwente.nl/Datasets.php. The dataset is collected for seven physical activities: walking, sitting, standing, jogging, biking, walking upstairs, and walking downstairs of different participants. Each activity was collected for each participant at a rate of 50 samples per second. We have examined the dataset, and we have noticed that all the participants were doing the same activity at the same time. For example, each participant was walking upstairs from time index 45 001 to 54 000 and was biking from time index 36 001 to 45 000 and the same for other activities. The total number of samples that we have used in our experiment is 54 000 in which each activity has 9000 samples for the following 6 physical activities: standing, jogging, sitting, biking, walking upstairs, and walking downstairs.

The data are collected by tri-accelerometer (x, y, z). We have transformed the three orthogonal planes into a vector magnitude such that

$$A = \sqrt{x^2 + y^2 + z^2} \tag{26}$$

where A is a vector magnitude. We have tested our algorithm COFCD (cases a and b) and RuLSIF on six physical activities (i.e., standing, jogging, sitting, biking, walking upstairs, and walking downstairs) for five participants and we have reported the performance evaluation in Table IX. We have tested the algorithms with the same parameter settings for our simulation results reported previously in Section V-C. However, we used $\alpha = 0.05$ while performing COFCD experiments which required adjustment as it is application dependent.

TABLE IX
Human Activity Dataset

	RuLSIF	COFCD (a)	COFCD (b)
FPR (%)	9	1	1
FNR (%)	0	0	0
Latency	28	19	10

TABLE X
Chest-Mounted Dataset

	RuLSIF	COFCD (a)	COFCD (b)
FPR (%)	11	0	0
FNR (%)	0	0	0
Latency	17	11	5

As shown in Table IX, RuLSIF and COFCD have the same FNRs. On the other hand, COFCD has a lower FPR compared with RuLSIF. Moreover, COFCD can detect changepoints faster than RuLSIF. RuLSIF requires more (i.e., 2.8 times more) number of samples to detect sequential change points compared with COFCD (case b). COFCD requires only 10 samples (case b) and 19 samples (case a) for detecting the same changepoints. Overall, such results match our simulation results as reported previously in Tables V and VI in which COFCD (case b) outperforms all other approaches and its fixed observation window (case a) performs better than RuLSIF.

Similar observations are obtained while conducting experiments on a real-world chest-mounted accelerometer dataset [35][1] where COFCD outperforms RuLSIF (as shown in Table X). The data are collected from a wearable accelerometer mounted on the chest of different participants performing different activities. The accelerometer has also been transformed into a vector magnitude (as explained previously) and we have used samples for walking, going up/down stairs, walking, and talking with someone activities. We used $\alpha = 0.001$, and other parameters are the same for COFCD experiments and RuLSIF.

It should be noted that our algorithm only focuses on detecting the changes to identify an activity, but we did not use any methods to label the type of activity. To label and identify the type of activity, different classification methods such as support vector machines can be used (we have labeled data to train the classifiers). The classification is not in the scope of this paper and to keep our description focused on change detection, we did not describe this aspect in this paper. The interested readers can refer to several well-established works in the domain including [6], [36], and [37].

[1]http://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer

VIII. Conclusion and Future Work

We have introduced a novel, online, and adaptive filtering-based change detection algorithm. The proposed algorithm employs a convex combination of two LMS adaptive filters with differing window sizes. We have extended our algorithm with a cooperative scheme between multiple sensors. We have then provided a comparison between our proposed algorithm and other baseline algorithms. The comparison includes two sets of results. The first set provides a comparison between the proposed algorithm and other state-of-the art algorithms in which the instantaneous changes have an increase and/or decrease in the mean value. The second set provides a comparison between the extended version of our algorithm with another state-of-the-art algorithm across multiple sensors. Through our experiments, our algorithm has provided a higher performance to remain fast responding and more accurate for detecting changes in the mean values compared with other baseline algorithms. Our algorithm also does not require any extra computational complexity. We have also investigated the parameters that affect our results the most in our sensitivity analysis. For future work, we will also consider applying our proposed algorithm to a larger scale and more dynamic data.

Although our cooperative scheme has shown a better performance for detecting changes, the future work will focus on an implementation for providing cooperation among neighboring sensors in a large-scale sensor networks that publish high-dimensional data streams. To deal with the dimensionality problem in streaming data, the algorithm will be further improved by incorporating dimensionality reduction techniques. We also plan to provide dynamic window sizes that can adapt based on the statistical properties of streaming sensory data.

References

[1] Y.-S. Lee and W.-Y. Chung, "Automated abnormal behavior detection for ubiquitous healthcare application in daytime and nighttime," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Inform.*, 2012, pp. 204–207.

[2] D. Kalus, M. Muma, and A. M. Zoubir, "Distributed robust change point detection for autoregressive processes with an application to distributed voice activity detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 3906–3910.

[3] U. Satija, B. Ramkumar, and M. S. Manikandan, "Robust cardiac event change detection method for long-term healthcare monitoring applications," *Healthcare Technol. Lett.*, vol. 3, no. 2, pp. 116–123, 2016.

[4] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statist. Anal. Data Mining*, vol. 5, no. 2, pp. 114–127, 2012.

[5] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Process.*, vol. 99, pp. 215–249, 2014.

[6] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, 2017.

[7] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[8] J. M. Lucas and M. S. Saccucci, "Exponentially weighted moving average control schemes: Properties and enhancements," *Technometrics*, vol. 32, no. 1, pp. 1–12, 1990.

[9] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system," in *Proc. 5th Asian Control Conf.*, 2004, vol. 2, pp. 815–818.

[10] C. Zou, Z. Wang, X. Zi, and W. Jiang, "An efficient online monitoring method for high-dimensional data streams," *Technometrics*, vol. 57, no. 3, pp. 374–387, 2015.

[11] J.-F. Chamberland and V. V. Veeravalli, "Wireless sensors in distributed detection applications," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 16–25, May 2007.

[12] F. Gustafsson and F. Gustafsson, *Adaptive Filtering and Change Detection*, vol. 1. New York, NY, USA: Wiley, 2000, ch. 3.

[13] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Netw.*, vol. 43, pp. 72–83, 2013.

[14] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 180–191.

[15] K. Yang and C. Shahabi, "A PCA-based similarity measure for multivariate time series," in *Proc. 2nd ACM Int. Workshop Multimedia Databases*, 2004, pp. 65–74.

[16] A. H. Sayed, *"Diffusion adaptation over networks," in Academic Press Library in Signal Processing*, vol. 3. Amsterdam, The Netherlands: Elsevier, 2013.

[17] P. Braca, S. Marano, V. Matta, and P. Willett, "Consensus-based Page's test in sensor networks," *Signal Process.*, vol. 91, no. 4, pp. 919–930, 2011.

[18] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, vol. 104. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[19] P. Granjon, "The cusum algorithm: A small review," Gipsa-Lab, Grenoble, France, Team SAIGA, Tech. Rep. hal-00914697, 2012. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00914697/

[20] M. Pollak, "Average run lengths of an optimal method of detecting a change in distribution," *Ann. Statist.*, vol. 15, pp. 749–779, 1987.

[21] M. Noori and S. Shahrtash, "Adaptive cumulative sum-based fault detector for transmission lines," in *Proc. 6th PSPC Conf.*, 2012.

[22] J. Arenas-García, A. R. Figueiras-Vidal, and A. H. Sayed, "Steady state performance of convex combinations of adaptive filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005, vol. 4, pp. iv/33–iv/36.

[23] M. Martinez-Ramon, J. Arenas-Garcia, A. Navia-Vázquez, and A. R. Figueiras-Vidal, "An adaptive combination of adaptive filters for plant identification," in *Proc. 14th Int. Conf. Digit. Signal Process.*, 2002, vol. 2, pp. 1195–1198.

[24] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Trans. Signal Process.*, vol. 54, no. 3, pp. 1078–1090, Mar. 2006.

[25] R. K. Martin, W. A. Sethares, R. C. Williamson, and C. R. Johnson, "Exploiting sparsity in adaptive filters," *IEEE Trans. Signal Process.*, vol. 50, no. 8, pp. 1883–1894, Aug. 2002.

[26] J. Arenas-García, V. Gomez-Verdejo, M. Martínez-Ramón, and A. R. Figueiras-Vidal, "Separate-variable adaptive combination of LMS adaptive filters for plant identification," in *Proc. IEEE 13th Workshop Neural Netw. Signal Process.*, 2003, pp. 239–248.

[27] J. Arenas-García, M. Martínez-Ramón, Á. Navia-Vázquez, and A. R. Figueiras-Vidal, "Plant identification via adaptive combination of transversal filters," *Signal Process.*, vol. 86, no. 9, pp. 2430–2438, 2006.

[28] K. Ochiai, T. Araseki, and T. Ogihara, "Echo canceler with two echo path models," *IEEE Trans. Commun.*, vol. COM-25, no. 6, pp. 589–595, Jun. 1977.

[29] Y. Wu, R. M. Rangayyan, Y. Zhou, and S.-C. Ng, "Filtering electrocardiographic signals using an unbiased and normalized adaptive noise reduction system," *Med. Eng. Phys.*, vol. 31, no. 1, pp. 17–26, 2009.

[30] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[31] T. Kanamori, T. Suzuki, and M. Sugiyama, "Statistical analysis of kernel-based least-squares density-ratio estimation," *Mach. Learn.*, vol. 86, no. 3, pp. 335–367, 2012.

[32] J. C. Loehlin, "The Cholesky approach: A cautionary note," *Behav. Genetics*, vol. 26, no. 1, pp. 65–69, 1996.

[33] S. M. Kuo and D. R. Morgan, "Active noise control: A tutorial review," *Proc. IEEE*, vol. 87, no. 6, pp. 943–973, Jun. 1999.

[34] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10146–10176, 2014.

[35] P. Casale, O. Pujol, and P. Radeva, "Personalization and user verification in wearable systems using biometric walking patterns," *Pers. Ubiquitous Comput.*, vol. 16, no. 5, pp. 563–580, 2012.

[36] Y. Kim and H. Ling, "Human activity classification based on micro-Doppler signatures using a support vector machine," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 5, pp. 1328–1337, May 2009.

[37] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1082–1090, Aug. 2008.

Authors' photographs and biographies not available at the time of publication.