Error Detection in Sequential Laser Sensor Input

G. Gatto, O. Hadjiliadis

Abstract In this paper, we build two, low complexity and robust, error-detection algorithms that adjust for sensor faults and inaccuracies in real time. The algorithms can detect consecutive and abrupt sensor slips, and provide a paradigm for real-time corrections. In addition to reliability, the algorithms do not require prior data knowledge, nor do they rely on any assumptions about the distribution of the data. The runtime is independent of the input size, which is ideal for large volumes of data, and allows for implementation at the sensor level.

1 Introduction

A major obstacle to progress in robotics is a machines limited ability to interact with the environment. Fortunately, the cost of laser sensors has decreased, making them an attractive choice as detection tools. Laser sensors are low-cost devices that are readily available and simple to operate. However, laser sensors can suffer from jitter and are not very reliable, especially when they encounter reflective surfaces. The trade-off for their convenience is the rudimentary quality of the data received and thus, the motivation for research on the topic of increasing sensor accuracy.

Prior research on sensor accuracy has been in either engineering or data processing. Engineering papers focus on the physical aspects of sensor construction and data acquisition, with the goal to obtain better data through

G. Gatto
Department of Mathematics and Statistics, Hunter College, CUNY, USA, e-mail: ggatto2@gmail.com

O. Hadjiliadis
Department of Mathematics and Statistics, Hunter College, CUNY, USA e-mail: olympia.hadjiliadis@gmail.com

hardware improvements (see, for instance, Morita *et al.*[10] and Shirafuji *et al.*[14]). This paper focuses on data processing. Given existing hardware, we construct two algorithms that correct sensor slips in real time, and therefore refine the quality of the data gathered in an online fashion. While other papers also focus on data-processing (see, for instance, James *et al.*[9] and Veiga *et al.*[16]), our algorithms do not require prior offline training of a classifier before they can be used in real time. That is, our algorithms are non-anticipative and do not rely on any distributional assumptions or prior training.

In particular, in this paper we construct low complexity error-detection algorithms to make data obtained from a two-dimensional laser sensor more accurate. Our improvement relies on the ability to detect in real time both persistent and instantaneous sensor slips, without the need of prior data knowledge or the assumption of a specific distribution. The first algorithm that we construct is a consecutive changes in mean algorithm and the second is a burst detection algorithm. Both algorithms are based on literature from the area of quickest detection which is concerned with detecting changes in the distribution of data received in real time (see Basseville and Nikiforov[1] and Poor and Hadjiliadis[12]). The consecutive changes in mean algorithm is based on Page's [11] CUSUM algorithm applied in repetition as in Carlisle *et al.*[3]. The second burst-detection algorithm is based on modeling the data using Hidden Markov Models (HMM) (see Ghahramani[7] and Rabiner and Juang[13]) and uses a CUSUM-like algorithm to detect a change from one HMM to another (see Chen and Willett[4] and Hadjiliadis and Stamos[8]).

In section 2 we discuss the data gathered for this research, which contains jitters and spikes; two particularly common errors encountered in laser sensor data. In section 3 we describe the two algorithms, and in section 4 we showcase our results and real-world performance.

2 Data Description

A SICK TiM571 scanning range finder, manufactured by Fetch, was mounted on a robot, and provided the laser sensor data. It is a two-dimensional sensor that measures distance from its origin using the time it takes for a laser beam to reflect back. The location of the sensor defines the origin of the axis of reference and an x-y plane is defined by the sensor. The laser has a range of 25 meters and a 220 degrees field of view. Its angular resolution is 1/3 of a degree with a 15Hz update rate. This means that one scan contains 660 points that span 220 degrees, and 15 scans are executed each second. In our data, one time step corresponds to one full scan. Each of the 660 points, defined by their (x, y) coordinates, is obtained by firing a laser beam in a straight line from the sensor until the beam collides with an object that reflects it back

to the sensor. The delay between the time the beam is sent and received is converted to distance in meters.

In the raw data, each time step represents one full scan. The 660 (x, y) pairs of coordinates represent distance in meters and are the result of one full scan to which the Berlotto [2] clustering algorithm is applied. This algorithm is divided into three sub-steps: data pre-processing, edge detection, and leg detection. In the data pre-processing sub-step, a local minimization operator is applied to reduce some of the noise from the data, such as noise from sloped surfaces, and a local maximization operator is applied to remove thin objects, such as table legs. Once the data has been preprocessed thusly, a vertical edge-detection filter is applied to obtain a sequence of vertical edges. From this sequence, the Bellotto-Hu clustering algorithm then categorizes groups of vertical edges in one of three patterns: LA (Legs Apart), FS (Forward Straddle), or SL (Single Leg / Legs Together). Each labelled grouping of vertical edges forms a cluster. Each cluster is then characterized by its centroid. In our case, the application of the Belotto-Hu algorithm results in the reduction of the 660 (x, y) points of the scan into at most 7 (x, y) cluster centroids. Each such centroid is given an index from $\{0, 1, \ldots, 6\}$.

One issue we experienced with the Bellotta-Hu clustering algorithm is its inability to reliably order the clusters it detects across scans. To remedy this defect, we implemented our own reordering algorithm (see table 1 for an example of the correction of cluster 0 at time-step 205) that assigns each cluster centroid at time step n the same index of its nearest neighbor at time $n - 1$, using the Pythagorean distance from the robot-mounted laser as the measure. This results in time step continuity of cluster centroids across scans. Our data is then the time series (x_n, y_n) for $n = 1, 2, \ldots, 739$, corresponding to the centroids of cluster 0.

Table 1 Sorting example at time step 205.

####	Before our sorting algorithm				After our sorting algorithm			
####	cluster 0 (x,y)		cluster 1 (x,y)		cluster 0 (x,y)		cluster 1 (x,y)	
time/scan	x	y	x	y	x	y	x	y
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
203	0.460388	-0.422849	2.455110	-0.352946	0.460388	-0.422849	2.455110	-0.352946
204	0.460214	-0.422685	2.430700	-0.342202	0.460214	-0.422685	2.430700	-0.342202
205	2.599970	-0.433634	0.460402	-0.422850	0.460402	-0.422850	2.599970	-0.433634
206	0.460663	-0.423155	2.414260	-0.332548	0.460663	-0.423155	2.414260	-0.332548
207	0.460762	-0.423176	2.402750	-0.345458	0.460762	-0.423176	2.402750	-0.345458
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Once the time series for cluster 0 was determined, we converted our data to polar coordinates:

$$(x_n, y_n)_{n=1,2,3,\ldots,739} \to (r_n, \theta_n)_{n=1,2,3,\ldots,739,}$$

where

$$r_n = \sqrt{x_n^2 + y_n^2}$$
$$\theta_n = \arctan(y_n/x_n)$$

at each time step.

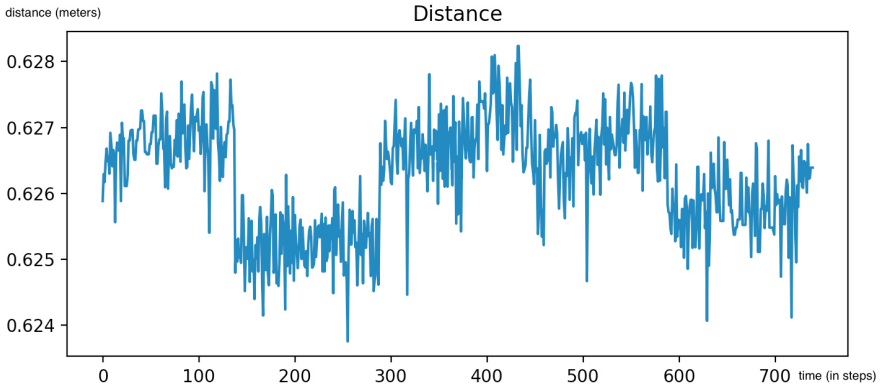

Fig. 1 Distance from the robot, in meters, for cluster no. 0 at each time step.

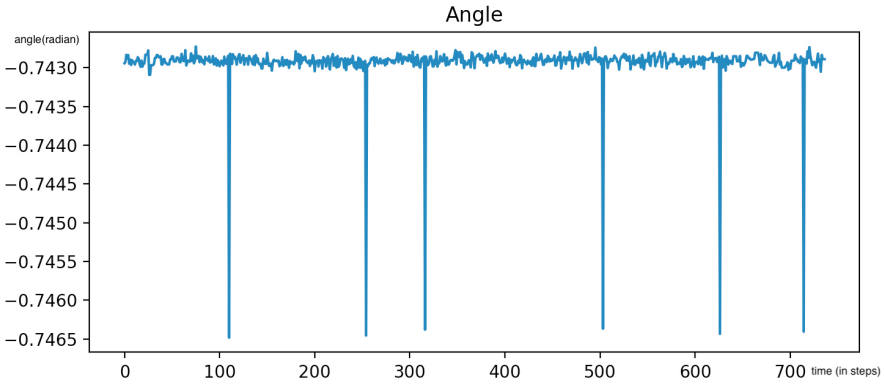

Fig. 2 Angle orientation from the robot, for cluster no. 0 at each time step.

We observe two types of sensor slips. The distance plot of figure 1 shows an up-and-down jitter pattern and the angle plot of figure 2 contains spikes that occur over intervals as short as one point. These inputs are known to be faulty since both the robot and the scene are static. The error shown in the distance plot pertains to a change in the mean, while the error illustrated in the angle plot can be described as a localized spike in the variance. Thus, we need to capture both types of errors to correct for faults in the input. In the next section we will build non-parametric algorithms to detect sensor slips in real time without making any assumption on the underlying distribution of the data.

3 Algorithms

In section 2, we observed that the sensor slips related to the distance display themselves as consecutive changes in the mean of the distribution of the data, while the faulty inputs related to the angles display themselves as localized spikes in the variance or as bursts (see figures 1 and 2). Therefore, we think of our data as the sequences of random variables $\{R_n\}_{n=0,1,2,...}$ and $\{\Theta_n\}_{n=0,1,2,...}$, whose realizations are the $\{r_n\}_{n=0,1,2,...}$ and $\{\theta_n\}_{n=0,1,2,...}$ described in section 2, respectively.

First, we devise an algorithm that can detect changes in the mean of the distribution of $\{R_n\}_{n=0,1,2,...}$ when it occurs. Secondly, we devise an algorithm to detect the localized variance spikes in $\{\Theta_n\}_{n=0,1,2,...}$. To achieve both goals, we use *quickest change detection*.

The field of *quickest change detection* (QCD) aims to identify the times when the probability distribution of a stochastic process changes. To formalize the quickest change detection problem, consider the sequentially observed variable $\{X_n\}_{n=0,1,2,...}$. We introduce the following hypothesis testing problem:

$$\begin{aligned} H_0 &: X_n \sim f_0, & n &= 1, 2, ..., \nu - 1 \\ H_1 &: X_n \sim f_1, & n &= \nu, \nu + 1, ... \end{aligned}$$

where f_0 is the starting distribution of the data and f_1 is the distribution the data moves to at time ν. The objective is to detect ν by minimizing the delay between ν, the change point, and the time the change point is declared. Detection of the change is declared by using a stopping time chosen to carefully balance the tradeoff between time of detection and the frequency of false alarms Poor and Hadjiliadis[12]. The stopping time is usually based on the threshold crossing of an appropriately chosen statistic process.

A popular stopping time to detect a change in distribution is known as the CUSUM stopping time (see, for instance, Chen and Willett [4] or Page[11] for the original concept or chapter 2.6 in Siegmund[15]). It is based on the

CUSUM statistic process $\{L_k^n\}_{n=1,2,...}$, which, for each n, is the likelihood ratio from k to n, $k \leq n$, where k is the index at which a change occurs(see equation (4) in Chen and Willett[4]). The CUSUM stopping time then takes the form:

$$N = \min\{n : \max_{0 \leq k \leq n} L_k^n \geq h\},$$

where $h > 0$ represents the threshold parameter. This leads to the construction of two algorithms based on the CUSUM stopping time, the first focuses on detecting consecutive changes in the mean and the second focuses on detecting bursts.

3.1 Algorithm for consecutive changes in mean

Consider the random variables $\{R_n\}_{n=0,1,2,...}$ whose realizations are the $\{r_n\}_{n=0,1,2,...}$ described in section 2. Without any assumptions about their distribution, we focus on detecting a change in their mean. Our hypothesis test then becomes:

$$\begin{aligned} H_0 &: E_0[R_n] = \mu_0, & n &= 0, 1, 2, ..., \nu - 1 \\ H_1 &: E_1[R_n] = \mu_1, & n &= \nu, \nu + 1, ... \end{aligned}$$

where μ_0 is the expected value of $\{R_n\}_{n=0,1,...,\nu-1}$ under the null hypothesis, and μ_1 is the expected value of $\{R_n\}_{n=\nu,\nu+1,...}$ after the change occurs.

In order to detect either case in which $\mu_1 > \mu_0$ or $\mu_1 < \mu_0$, we introduce the constant k to capture the sensitivity to a change upward or downward, and construct two CUSUM stopping times to detect each of the changes (see chapter 16 in Devore[5]).

1. $\mu_1 > \mu_0$.

$$N^+ = \min\{n : d_n \geq h\}, \tag{1}$$

where

$$d_n = \max[0, d_{n-1} + (r_n - (\mu_0 + k))], \text{with } d_0 = 0. \tag{2}$$

The $\{d_n\}_{n=0,1,...}$ constitute the CUSUM statistic process, r_n is the realization of the data at time n, μ_0 is the starting mean of the data, and k is the upward change that is allowed before the process is considered out of control.

2. $\mu_1 < \mu_0$.

$$N^- = \min\{n : e_n \geq h\}, \tag{3}$$

where

$$e_n = \max[0, e_{n-1} - (r_n - (\mu_0 - k))], \text{ with } e_0 = 0. \quad (4)$$

The $\{e_n\}_{n=0,1,\ldots}$ is the CUSUM statistic process, r_n is the realization of the data at time n, μ_0 is the starting mean of the data, and k is the downward change that is allowed before the process is considered out of control.

We notice that $d_n \geq 0$ and $e_n \geq 0$ for all $n = 0, 1, 2, \ldots$; i. e., both of the CUSUM statistics processes appearing in equations (2) and (4) are capped below at 0. We can thus select a positive threshold h in the above equations. We then compare both processes to the threshold h. A different threshold could be set for (2) and (4). However, since we have no reason to believe that a change in the mean of the distribution is more likely to be positive or negative, we choose to set the same threshold for both CUSUM statistic processes. The first time $d_n \geq h$ or $e_n \geq h$, an alarm is raised. The threshold h can be set either with domain knowledge, or as an appropriate multiple of the deviation k. In our application, we set h to be $5 * k$ and the mean μ_0 is initialized from the sample mean of the first 20 data points. In practice, the parameter k represents the sensitivity of the algorithm to changes in the mean, and can thus be set to a percentage of the original mean μ_0. In particular, in our paper it is set to 1% of μ_0. After an alarm is raised, both CUSUM statistic processes in equations (2) and (4) are reset to zero, and the new mean μ_1 is set to

1. $\mu_1 = \mu_0 + k$, if $N^+ \wedge N^- = N^+$ (see (1),(3)) and
2. $\mu_1 = \mu_0 - k$, if $N^+ \wedge N^- = N^-$ (see (1),(3))

Note that the algorithm is robust with respect to the choice of threshold. This is further discussed in section 4 of this paper.
The algorithm is detailed below using pseudocode notation[1]:

1. Set the values of the following parameters
   - μ_0: the starting mean of the data.
   - k: the deviation allowed from the mean (usually taken to be $\frac{1}{2}$ the allowed deviation from the process mean).
   - h: the threshold whose crossing triggers and alarm (usually set as a multiple of k).
2. Initialize an upper and lower CUSUM statistic.
   - Set d_0 to 0
   - Set e_0 to 0
   - Set μ to μ_0
3. Read in data of r_n's and compute, at every time step $n = 1, 2, \ldots$,

[1] For the implementation of the algorithm please refer to Gatto[6].

- $d_n = \max[0, d_{n-1} + (r_n - (\mu + k))]$
- $e_n = \max[0, e_{n-1} - (r_n - (\mu - k))]$

4. If at time n, $d_n \geq h$
   - Raise alarm
   - Set d_n to 0
   - Set e_n to 0
   - Set μ to $\mu + k$
5. If at time n, $e_n \geq h$
   - Raise alarm
   - Set d_n to 0
   - Set e_n to 0
   - Set μ to $\mu - k$
6. Repeat step 2 through 5 until data stream ends.

3.2 Algorithm for Burst Detection

In this subsection, we build an algorithm to detect sensor slips as manifested in the angles data. The sensor slips in the angles data are manifested as bursts, or spikes, which occur over a minute time interval (see figure 2). Such slips can be described as localized increases in the variance of the data. To detect such changes, we therefore must capture changes in that variance that occur over minute intervals. This makes the typical CUSUM algorithm we used previously ineffective in this case. As we can see in figure 3, the typical CUSUM algorithm fails to detect the first and fourth angle spikes and further falsely detects a spike around time step 475. Even a CUSUM algorithm used to capture changes in the variance parameter itself, would either be blind to such changes or be too sensitive to be practical.

Therefore, to facilitate the construction of a CUSUM algorithm which detects the spikes, we must introduce a model that allows for spikes. This leads us to the consideration of Hidden Markov models (HMM).

A discrete time finite-state HMM is a probabilistic model of a sequence of random variables $\{X_t\}_{\{t\geq 0\}}$ whose underlying structure is a Markov chain. A key advantage of HMMs is the assumption that an observation at time t originates from a process whose inner working is hidden from the observer Ghahramani[7]. Therefore, the distribution of the X_t's for each $t = 0.1.2., \ldots$ depends on the value of a sequence of state variables $\{s_t\}_{t=0,1,\ldots}$, with $s_t \in \{1, ..., N\}$ which are governed by the N x N transition matrix A . A discrete HMM can be described by the triple,

$$\lambda = (A, B, \pi), \tag{5}$$

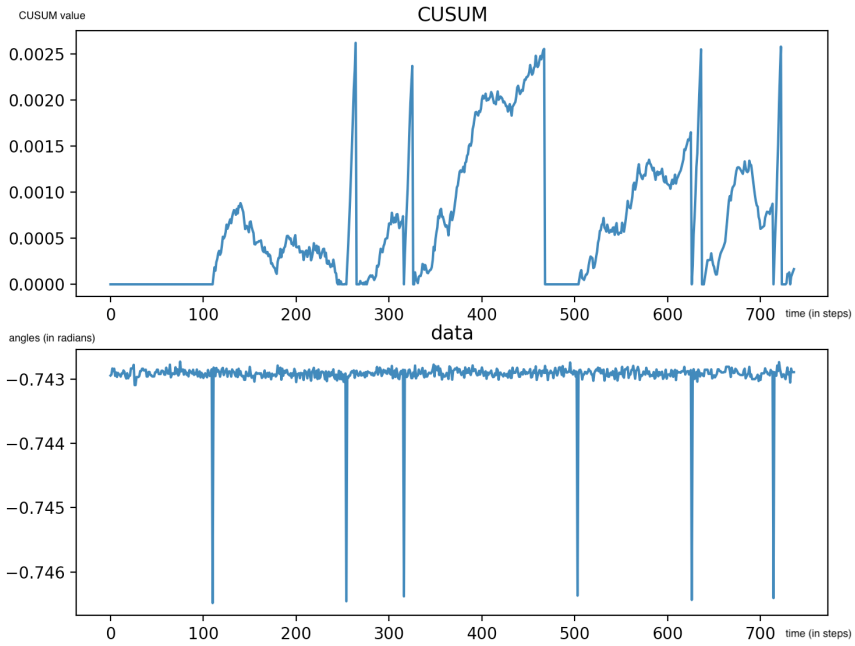

Fig. 3 The CUSUM algorithm of section 3.1 fails to detect the first and fourth spikes, and falsely detects a spike around time step 475.

where A is the state transition matrix of the underlying Markov chain,

$$A = [a_{ij}] = [P(s_{t+1} = j | s_t = i)], \qquad i, j = 0, ..., N,$$

B is the observation matrix,

$$B = [b_{ix_t}] = [P(X_t \in dx_t | s_t = i)], \qquad i = 1, ..., N,$$

and π is the initial probability distribution of the underlying Markov states,

$$\pi = [\pi_i = P(s_t = i)], \qquad i = 0, ..., N.$$

In our case, we set $N = 1$. That is, there are two states, 0 and 1. We then use two HMM models to describe the probabilistic behavior of our sequence of angle random variables $\{\Theta_t\}_{\{t \geq 0\}}$; one that does not allow spikes in variance, and one that does. In other words, one in which state 0 is the only possible state and one in which both states 0 and 1 are possible. To simplify the model for Θ_t's we first normalize by subtracting the sample mean of the first few observations (the first 20). We can now represent the "low" and "high" variance regimes by using two gaussian models in the observation matrix, one with a "low" and one with a "high" variance parameter. Note

that the Gaussian assumption in the observation matrix is redundant. All we need, is a distribution with a finite second moment so that the "low" and "high" variance regimes can be distinguished (in our case with a low or high variance). Arbitrarily, we chose state 0 to be the state with low variance and state 1 to be the state with high variance. Our observation matrix is:

$$B = \begin{bmatrix} \frac{1}{\sqrt{2\pi}\sigma_{\text{low}}} e^{\frac{-x}{2\sigma^2_{\text{low}}}} \\ \frac{1}{\sqrt{2\pi}\sigma_{\text{high}}} e^{\frac{-x}{2\sigma^2_{\text{high}}}} \end{bmatrix}. \tag{6}$$

The low variance is the baseline variance, obtained by computing the sample variance of the first few points of data (in our case the first 20 points). An alternative method is to set the variance by using domain knowledge. The high variance is set as a multiple of the low variance. We used a multiple of 5 in our case. The algorithm is extremely robust with respect to the multiple that is used. Although the observation matrix uses the Gaussian kernel, the data from a normal distribution is not necessary for adequate real-world performance.

Now, let A_0, π_0 be the state transition matrix and the initial probability distribution vector for the HMM that does not allow spikes in the variance, and A_1, π_1 be the state transition matrix and the initial probability distribution vector for the HMM that allows spikes in the variance.

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \pi_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{7}$$

$$A_1 = \begin{bmatrix} 0.9 & 0.1 \\ 0.9 & 0.1 \end{bmatrix}, \pi_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}. \tag{8}$$

Under model $\lambda_0 = (A_0, B, \pi_0)$, state 1 is impossible as the initial probability distribution vector π_0 begins in state 0 which is an absorbing state according to the state transition matrix A_0. On the other hand, under the model described by $\lambda_1 = (A_1, B, \pi_1)$, both state 0 and state 1 are possible and, although transitions from state 0 to itself are more likely to occur, it remains possible to transition from state 0 to state 1, which is a "high" variance state. The choice of values for the transition probabilities in matrix A_1 reflects the possibility to transition from any state to any state with a higher probability to remain in state 0 when in it, and a higher probability to transition to state 0 when in state 1. Such a choice of transition matrix A_1 describes the nature of the possibility of instantaneous bursts, since state 1 is a high-variance state. Our algorithm is robust with respect to the exact choice of the transition probabilities in the A_1 matrix above. A demonstration of the robustness of our algorithm to the exact choice of the A_1 matrix is seen in section 4.

This gives us the following hypothesis testing:

$$H_0 : \Theta_t \sim \lambda_0, \qquad t = 1, 2, \ldots$$
$$H_1 : \Theta_t \sim \lambda_1, \qquad t = 1, 2, \ldots$$

Θ_t's are the sequence of random variables representing the angles as described in section 2, $\lambda_0 = (A_0, B, \pi_0)$, and $\lambda_1 = (A_1, B, \pi_1)$.

Our goal is to determine which model at each point in time is most likely to describe the data. To that end, we introduce a conditional form of the log likelihood ratio $g(n; k)$ (refer to equation (24) in Chen and Willett[4])

$$g(n; k) = \sum_{t=k}^{n} \log \left(\frac{f_1(\theta_t | \theta_{t-1}, \ldots, \theta_k)}{f_0(\theta_t | \theta_{t-1}, \ldots, \theta_k)} \right), \tag{9}$$

where f_0 and f_1 are the conditional distributions of the Θ_t's under H_0 and H_1, respectively, which enables the construction of the CUSUM-like statistic algorithm of equation (23) in Chen and Willett[4]:

$$S_n = \max\{0, S_{n-1} + g(n; k)\}. \tag{10}$$

In what follows we will use this paradigm to build an algorithm that detects bursts in real time.

The joint probability density of the history of the stochastic process and the state at a certain time is known as the forward variable. It is defined as:

$$\alpha_t(i) = p(\theta_1, \theta_2, \ldots, \theta_t, s_t = i | \lambda), i = 0, 1, \ldots, N, t = 0, 1, \ldots, \tag{11}$$

where s_t is the state occupied at time t, i is the state index, and N is the total number of states. Due to the Markovian property, the forward variable can easily be computed since it depends solely on the last state, and not on the joint probability of all the states visited so far.

In our case the number of states is 2; that is $N = 1$. According to (25) in Chen and Willett[4], the likelihood function of an HMM with parameter triple λ can be written as

$$f(\theta_1, \theta_2, \ldots, \theta_t | \lambda) = \sum_{i=0}^{1} \alpha_t(i), \tag{12}$$

where $\alpha_t(i)$'s are the sequence of forward variable that we use to compute the likelihood of the HMM described by λ in (5).

Furthermore, note that (9) is equivalent to

$$f_l(\theta_t | \theta_{t-1}, \ldots, \theta_1) = f(\theta_t | \theta_{t-1}, \ldots, \theta_1, \lambda_l) = \frac{\sum_{i=0}^{1} \alpha_t(i)}{\sum_{i=0}^{1} \alpha_{t-1}(i)}, l \in \{0, 1\}, \tag{13}$$

where f_l is the conditional distribution of Θ's described by λ_l for $l = 0, 1$ corresponding to H_0 and H_1 respectively. In other words, the conditional

form of the log-likelihood ratio $g(n;1)$ of (9) can be expressed as the ratio of the sum of forward variables, which can easily be computed as a running sum.

Due to the limitation of machine precision, computations with increasingly small values cause numerical underflow. In order to circumvent this issue, we use the scaled forward variable α'_t as defined in equation (27) in Chen and Willett[4]. That is, $\alpha'_1(i) = \alpha_1(i)$ and for $t > 1$:

$$\alpha'_{t+1}(j) = \frac{\left[\sum_{i=0}^{1} \alpha'_t(i) a_{ij}\right] b_{j\theta_{t+1}}}{\sum_{i=0}^{1} \alpha'_t(i)}, j = 0, 1 \tag{14}$$

where a_{ij} is the state transition probability to state i from state j, and $b_{j\theta_{t+1}}$ is the 1st entry in the matrix B of (6) for $j = 0$ and the 2nd entry of the matrix B of (6) for $j = 1$, evaluated at the observation θ_{t+1}. Please note that there are two sets of α'_t's corresponding to each state for each of the hypothesis H_0 and H_1, resulting in four arrays of α'_t' s indexed by the time variable t.
The algorithm is given below in pseudocode notation [2] :

1. Set the values of the follow parameters:
   - t to 1
   - l_0 to 0, where l_t denotes the LLR at time t.
2. Initialize the scaled foward variable α'_t:

$$\alpha'_t(j) = \pi_j b_{j\theta_t} \tag{15}$$

   for each state $j = 0, 1$ and for both hypotheses H_0 and H_1.
3. Update the log-likelihood ration process $\{l_t\}_{t\geq 0}$ as follows:

$$l_t = l_{t-1} + \log\left(\frac{\sum_{i=0}^{1} \alpha'_t(i|H_1)}{\sum_{i=0}^{1} \alpha'_t(i|H_0)}\right) \tag{16}$$

4. If $l_t > h$, declare detection of high variance.
   If $l_t < 0$
   - l_t to 0
   - t to $t + 1$
   - goto 2
5. Else (if $0 < l_t < h$)
   - set t to $t + 1$
   - Update α'_t using (14)
   - goto 3

[2] For the implementation of the algorithm please refer to Gatto[6].

Please note that $g(t;1)$ of (9) is

$$g(t;1) = \log\left(\frac{\sum_{i=1}^{N} \alpha_t'(i|H_1)}{\sum_{i=1}^{N} \alpha_t'(i|H_0)}\right)$$

of step 3 and the CUSUM-like algorithm is constructed by the sequence of l_t's above by setting the left-hand threshold to 0 (see Siegmund[15]); that is, $\{l_t\}_{t\geq 0}$ represents the log likelihood ratio process LLR.

4 Results

We applied the consecutive changes in mean CUSUM algorithm of section 3.1 to detect consecutive changes in the mean of the distances of the data. Figure 4 is the graph of figure 1 presented in section 2 with the changes we aim to capture highlighted. We ran our algorithm on the data of distances and compiled the results in figure 5 for different thresholds ($h = 2.5k$, $h = 5k$, and $h = 10k$). Our algorithm managed to capture the three changes we intended to capture, and it did so for various values of h, demonstrating robustness of the consecutive changes in mean CUSUM algorithm with respect to the threshold (i. e., the threshold does not need to be perfectly tuned). There is a slight lag in detection when the threshold is increased, which is expected.

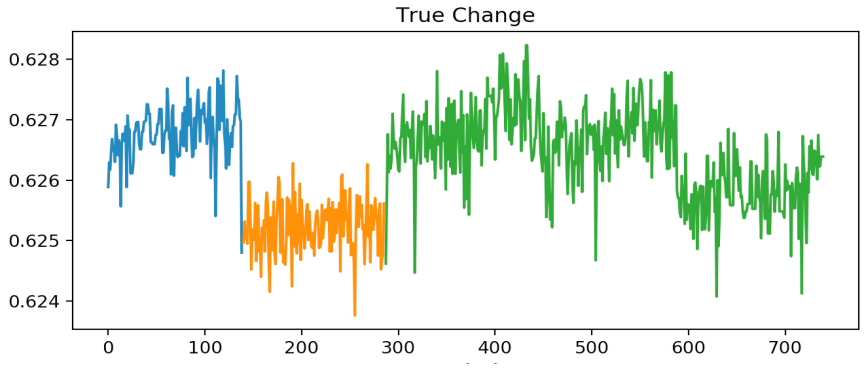

Fig. 4 The boundaries of the highlighted sections mark when a change happens.

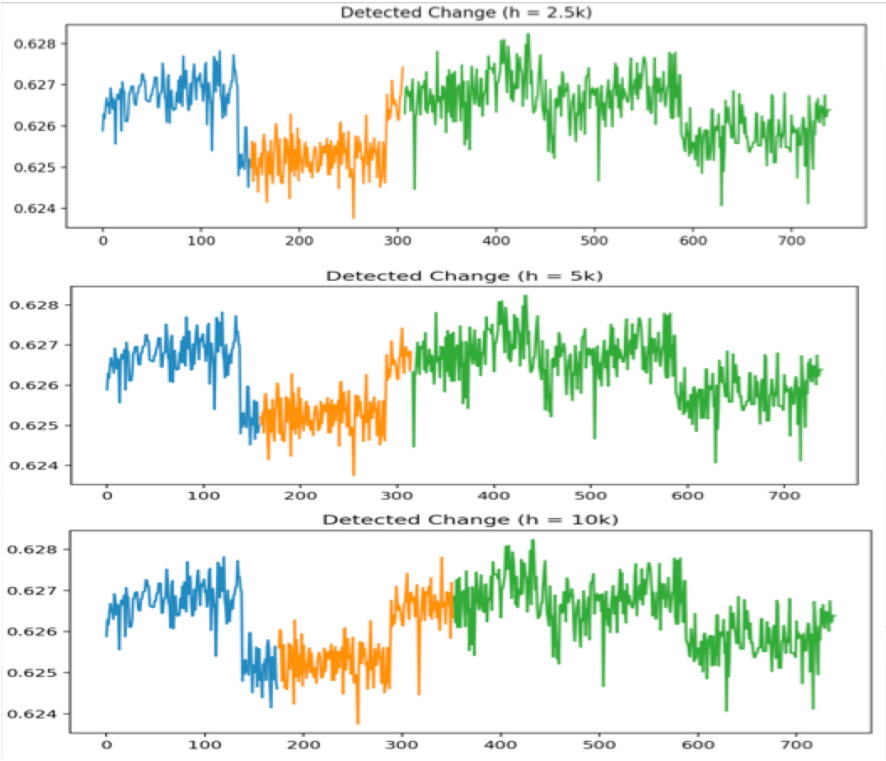

Fig. 5 The boundaries of the highlighted sections represent the different levels in mean detected by our algorithm with thresholds set to $h = 2.5k$, $h = 5k$, and $h = 10k$, respectively.

We ran the burst-detection algorithm to detect instantaneous sensor slips on the angle component of the data, shown in figure 6. As demonstrated in the graph, the LLR statistic spikes up whenever there is a sensor slip in the angles data. Therefore, almost any threshold would be sufficient in detecting a sensor slip. To test the robustness of the algorithm with respect to the choice of transition matrix A_1 (see (8)), we ran the algorithm with the matrix in (8) and the following matrices:

$$A_1' = \begin{bmatrix} 0.1 & 0.9 \\ 0.1 & 0.9 \end{bmatrix}, A_1'' = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

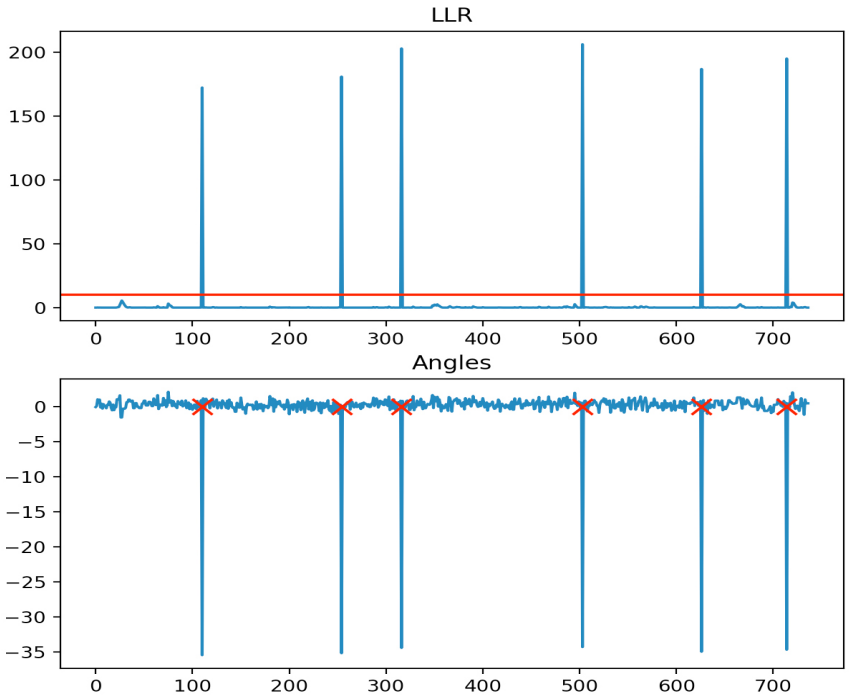

Fig. 6 Comparison of LLR values to the plot of angles. The transition matrix is the same as in (8).

Regardless of which transition matrix was used, our algorithm detected the spikes at the same locations. Therefore, the burst-detection algorithm is very robust with respect to the choice of parameters in the models of subsection 3.2.

5 Acknowledgements

We would like to acknowledge Professor Ioannis Stamos for sharing his data and giving access to his lab, along with his graduate students Jaspal Singh and Jamie Canizales. We would also like to acknowledge Andrew J. Pearson, ALM, for his thorough proof-reading of the paper.

References

1. M. Basseville, I. Nikiforov. *Detection of Abrupt Changes: Theory and Practice*, Prentice Hall,1993.
2. N. Bellotto, H. Huosheng. Multisensor-Based Human Detection and Tracking for Mobile Service Robots, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 1, **39**, 167 – 181, 2009. DOI: 10.1109/TSMCB.2008.2004050.
3. M. Carlisle, O. Hadjiliadis and I. Stamos. Trends and Trades, *Handbook of high-frequency trading and modeling in finance*, Chapter 1. Editors: F. Viens, M. C. Mariani and I. Florescu, Publisher: John Wiley and Sons, 2016.
4. B. Chen and P. Willett. Detection of hidden Markov model transient signals, *IEEE, Transactions on Aerospace and Electronic Systems*, 4, **36**, 1253 – 1268, 2000. DOI: 10.1109/7.892673.
5. J. Devore. *Probability and Statistics for Engineering and the Science*, 8th edition, Brooks/Cole, Boston, MA, 2012.
6. G. Gatto. Error detection algorithm, *https://github.com/Revidyloh/error-correction*, v.1.0, 2019. DOI: https://zenodo.org/badge/latestdoi/194529788.
7. Z. Ghahramani. An Introduction to Hidden Markov Models and Bayesian Networks", , *International Journal of Pattern Recognition and Artificial Intelligence*, 1, **15**, 9 – 42, 2001. DOI: 10.1142/S0218001401000836.
8. O. Hadjiliadis and I. Stamos. Sequential Classification in Point Clouds of Urban Scenes, *Proceedings of the 5th International Symposium on 3D Data Processing, Visualization and Transmission*, Paris, France, May 17-20, 2010.
9. J. W. James, N. Pestell, N. F. Lepora. Slip Detection With a Biomimetic Tactile Sensor, *IEEE Robotics and Automation Letters*, 4, **3**, 3340 – 3346, 2018. DOI: 10.1109/LRA.2018.2852797.
10. N. Morita, H. Nogami, E. Higurashi, R. Sawada. Grasping Force Control for a Robotic Hand by Slip Detection Using Developed Micro Laser Doppler Velocimeter, *Sensors*, 2, **18**, 326 – 343, 2018. DOI: 10.3390/s18020326.
11. E. S. Page. Continuous Inspection Schemes, *Biometrika*, 1-2, **41**, 100 – 115, 1954. DOI: 10.2307/2333009.
12. H. V. Poor and O. Hadjiliadis. *Quickest Detection*, Cambridge University Press, UK.
13. L. Rabiner, B. Juang. An introduction to hidden Markov models, *IEEE ASSP Magazine*, 1, **3**, 4 – 16, 1986. DOI: 10.1109/MASSP.1986.1165342.
14. S. Shirafuji, K. Hosoda. Detection and prevention of slip using sensors with different properties embedded in elastic artificial skin on the basis of previous experience, *Proceedings of the 15th International Conference on Advanced Robotics (ICAR)*, , Estonia, Jun 20-23, 2011. DOI: 10.1109/ICAR.2011.6088598.
15. D. Siegmund. *Sequential Analysis*, Springer-Verlag, New York,1985.
16. F. Veiga, H. Van Hoof, J. Peters, and T. Hermans. Stabilizing novel objects by learning to predict tactile slip, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5065 – 5072, Hamburg, Germany, Sep 28 - Oct 2, 2015.