Obstacle Detection based on Cooperative-Intelligent Transport System Data

Brice Leblanc, Hacene Fouchal, Cyril de Runz

Obstacle Detection based on Cooperative-Intelligent Transport System Data

Brice Leblanc, Hacène Fouchal and Cyril de Runz
CReSTIC, Université de Reims Champagne-Ardenne France
{brice.leblanc, hacene.fouchal}@univ-reims.fr,
BDTLN, LIFAT, University of Tours, France
cyril.derunz@univ-tours.fr

***Abstract*—Cooperative Intelligent Systems development is growing and the data they produce is increasing exponentially. This amount of data will soon be large enough to fall in big data paradigm. We propose to exploit these data as data stream. We aim to detect anomaly on the road using concept drift detection methods over data stream. To achieve this purpose, we create a data generation tool to obtain large data-sets of vehicles taking an avoiding behavior and detect obstacles through crowdsensing. We use two scenarios that we aim to detect: a stopped car and a growing pothole. We focus our study on the vehicle orientation information on which we apply Page-Hinkley and ADWIN methods. We obtain interesting detection results with ADWIN on the stopped car scenario. The Page-Hinkley algorithm is obtaining good results but with a latency that makes it unexploitable in real context. But for the pothole detection, both approaches are not providing significant results.**

I. Introduction

With the development of Cooperative Intelligent Transport Systems (C-ITS), many vehicular trajectories are produced. This data can be exploited to help road operators in the detection and prevention of events. In this paper, we aim to detect events happening on a road using crowdsensing. We assume that if a large enough number of vehicles switch from a classical driving behavior (with a low percentage of overtaking behavior) to a behavior with a high percentage of overtaking behavior, an obstacle on the road can be the cause of it. Thus, we aim to detect these changes in the collected data stream. To do so, we focus on the vehicles orientation information. This information was estimated important in our previous work on driving profile detection [8], [9], and we believe it can also apply to this context. We base our study on two different scenarios, a stopped car scenario, and a growing pothole scenario. To obtain a data-set of vehicle producing massive avoiding behavior and respecting the scenarios, we build a data generation tool. This tool generates messages containing the position, speed and heading of the vehicle, and respecting the ETSI[1] standardization for the frequency of emission of messages. The variables contained in the generated messages respect the Cooperative Awareness Message (CAM) [1] standard.

[1]https://www.etsi.org/

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

To built our data generation tool, we made the following assumptions:

- The generated vehicles maintain their speed at the maximum of the road limitation, even during the avoidance of the obstacle. If not, we could see the change in the speed of the vehicles which is a strong indicator of troubles happening on the road.
- The generated drivers have no previous knowledge of the obstacle and cannot anticipate it too much. This is why in the data generation tool, we estimate that the angle to avoid the obstacle can not be inferior to 4°. Otherwise, the time that will be required to avoid it will be too long.
- The vehicles once starting to avoid the obstacle do not change their direction until the vehicle is on the other lane.
- Our default number of vehicles overtaking other vehicles (number of avoiding behavior) is 5 percent.

The data that compose the stream are collected by a Road Side Unit (RSU) placed along the road and monitoring the traffic on a portion of the road. To detect anomaly on the road we provide to concept drift algorithms the generated headings of the vehicles. We select two algorithms and compare their results for the two scenarios.

The paper is organized as follows. In section II, the state of the art methods of concept drift detection in data streams is presented, section III describes the data generation tool we use, section IV presents the result of drift detection and in section V we give a conclusion and provide some perspectives to this work.

II. State of the art

It is noticeable that, in the literature, road obstacle detection approaches focus on smartphone data [11], ultrasonic sensors [10] or even image processing for autonomous vehicles [3]. In this study, we aim to explore the detection of road obstacles by using data produced directly by vehicles. To do so, we are handling these data as data stream.

There can be different kinds of change in a data stream. From time to time an outstanding value appear, this is called an outlier. When the data is changing from one behavior to another, this is called a concept drift. A concept drift detector is designed to find when the data is changing from one concept to another, but it shall not trigger on outlier data. Different

Fig. 1: Picture of Voie Taittinger

concept drift detection approaches are used for the different kind of data and stream. The parameters of these algorithms are used to tune the algorithm to avoid a trigger on outlier or not triggering at all. Parameters are varying depending on the context.

To handle data stream, the algorithms can store some global value relative to the stream that are updated for each new data or rely on a window model to store part of the stream and calculate the values on the stored data. In windows model, data is stored until the window is full and since the memory is limited older data will be removed from the window [6]. Here are some windows model that can be used:

Sliding window model : In this window model the data is treated in a first-in first-out manner. The size of the window can be fixed or variable but when the window is full, oldest data are deleted so new data can be treated.

Damped window model : This model of window associate an exponentially decaying weight to the observations and delete the data when the weight totally fades out.

Landmark window model : The landmark model rely on chunks of data separated by landmarks. A landmark can be a time value (hour, day, month, ...) or a number of element observed. Every data in the landmark is treated until the next landmark is reached. When it is reached, the old data is removed and replaced by the new one.

In this study, we used two algorithms: Page-Hinkley and ADWIN. These are really popular approaches due to their effectiveness on many types of data, and we aim to know if they are adapted to our type of data:

The Page-Hinkley algorithm [12] [5] analyzes sequentially the data to detect change and do not use a window model since no data is stored except a mean and a sum. It recalculates the mean value of the data at each input. And it also recalculates the sum of the difference to the mean with the alpha and the delta parameters to adjust the sensitivity. The alpha and the delta parameters help to mitigate outliers both in different ways, the greater they are the more outliers will be needed to detect a drift. If this sum passes over the lambda threshold value then a drift signal is raised. The greater the threshold is, the fewer false positives are but actual errors could be missed or the detection delayed. Also the higher the alpha and delta values are the harder it is to detect small variations. Page-Hinkley is consuming very few resources since it is not storing any part of the data stream. But its strongest issues are its sensitivity to outlier when trying to detect concept drift on low varying data and its delay on the detection of concept drift when tuned to resist to outliers.

The ADaptative WINdowing (ADWIN) algorithm [4] is based on a sliding window system. The size of the window, instead of being fixed, is recomputed: if a drift is detected, the window is reduced, if not, it is growing to its maximal size defined by the user. To change the window size, it is made into a bucket list that is split in bucket row of the same size, and these buckets row contains buckets. The algorithm takes data as input, stores it in a bucket that is put in the last bucket row. If the bucket list is full, the two oldest bucket row are reduced and merged. The process to detect the drift is triggered every *clock* number of new data, only if the length of the window is greater than the minimal sub-window length. To detect a drift, the buckets are separated in two sub-windows, one containing the oldest data (this one is bigger than the second one) and the other containing newer data. If the data between these two windows are too different (according to the delta value) then a drift signal is raised, and the window size is reduced. ADWIN has a small memory consumption due to its bucket system and can detect quickly concept drift since part of the stream is stored. But since a small part is stored long and slow drift is hard to detect because the buckets are updated with more and more drifting data without noticing it. And if the algorithm is more sensitive to detect such change the rate of false-positive will be higher.

Different approaches also exist such as ECDD or SONDE among others :

The Exponentially weighted moving average (EWMA) [14] for Concept Drift Detection (ECDD) [13] algorithm is learning from the data stream using a classifier. The stream is parsed once and not stored unlike ADWIN and when an event is detected, the classifier is updated to handle it (it can be cleared to restart the learning).

The Self-Organizing Novelty Detection (SONDE) [2] algorithm is based on Self Organized Map (SOM) [7] to learn and detect new patterns in the data stream.

III. Data Generation

For our data generation we use real data issued from open road testing (collected within the scope of the InterCor European project) as basis. We select a part of 300 meters of the *Voie Taittinger* in Reims which is a 4 ways road limited to 90km/h that can be seen in Fig.1.

We limit to 300 meters the road portion for two reasons: Firstly we suppose that our roadside unit should be placed in the middle of the collection area. Secondly, our vehicles have a 150 meters communication range. This communication range has been decided according to the result of a French governmental study on the C-ITS [2]. This study shows that after 150 meters the packet delivery ratio starts declining for vehicle to infrastructure communication in highway scenario.

This road has been selected because it is straight, and the latitude from the start to the end of the portion is not varying a lot. This provides us with a worst-case scenario regarding the CAM generation frequency management (GFM) explained in III-A. The CAM GFM will not trigger and not increase the frequency of the message sending because of the road natural curve.

The data generation tool is producing a data-set containing the position (latitude, longitude), the heading, a timestamp and a vehicle id. We also add a boolean value as label to know if a generated message is a taking an avoiding behavior or not. The data generation has a defined duration, a number of vehicle and a percentage of obstacle avoidance behavior.

A. CAM Generation Frequency Management (GFM)

In the C-ITS, CAM frequency is, by default, set at 1 per second. A system to send more messages per second can be triggered in specific cases up to a maximum of 10 messages per second. This system is called Generation Frequency Management (GFM) and allows a vehicle to send more information about itself to other vehicles in its neighborhood when an abnormal behavior is detected. It triggers if one of the following conditions is met :

- The difference between the last emitted position and the current position is greater than 4 meters.
- The difference of speed between the last emitted speed and the current speed is greater than 0,5 m/sec.
- The absolute difference of heading between the last emitted message and the current value is greater than 4°.

When the GFM is triggered, the frequency of sending is update and equal to the time elapsed between the last message emission and the detection. This frequency is maintained for 3 messages but the GFM can be triggered again when a new event is detected to a minimum frequency of 100ms. In our data generation tool, we assume that the GFM will not trigger again.

B. Initial configuration

The data generation tool uses as landmark 12 fixed positions (composed of latitude, and longitude) manually selected at a distance of 25 meters each (90km/h in one second = 25 meter) on *Voie Taittinger*. A vehicle heading was also added to the landmark and was selected from real data generated by C-ITS vehicles during open road tests on the same road. For each generated vehicle, its first message contains the latitude, longitude, and heading of the first landmark with an initial hazard. This hazard is from -200 to +200 for the latitude, -3000 to +3000 for the longitude, and -2 to +2 for the heading. The timestamp follows this linear equation : Let d the duration of the generation, i the identifier of the vehicle, and v the total number of vehicle.

$$d * i/v \tag{1}$$

C. Default behavior

In default behavior, the following messages of the vehicle are separated by 1 second. The new messages are composed of the corresponding landmark plus the initial hazard plus the new hazard (this last one is not kept for further messages). The latitude has a hazard of -200 to +200, the longitude -3000 to +3000, and the heading of -2 to +2.

D. Avoiding behavior

Based on the CAM GFM specification, our vehicles must have a minimum of 4° heading modification to start the emission of the 3 higher frequency messages. For a vehicle to change lanes in one second, the minimum angle is 10° on our 90Km/h example. Similarly, if a vehicle changes lanes with a maximum angle of 4°, it will take 50 meters (2 seconds). Therefore, a vehicle cannot change lanes without a trace in the data. We have set the maximum angle at 45°.

In our case of avoiding behavior, the vehicle will produce 4 default messages before sending its first message with a different timestamp. After sending the three avoiding behavior messages, the vehicle will return to the default behavior. The process of generating an avoiding message starts with the random selection of the time difference from the last message sent (between 100 ms and 1000 ms). Then, the angle taken by the vehicle for overtaking is randomly selected between 4° and 45°. If the timestamp is less than 500 ms, an additional default message will be replaced by an avoiding behavior message; otherwise, two messages will be replaced. For each avoiding message, the change in latitude and longitude is calculated according to this formula :

$$lat = latOfLastMessage + 300/(timestampInterval/10) \tag{2}$$

$$lon = lonOfLastMessage + 3000/(timestampInterval/10) \tag{3}$$

Also the remaining default behavior messages have their positions updated according to the position of the last avoiding behavior message.

E. Used data-set

To produce a large data-set with a frequency of change in avoiding behavior, we run multiple generations (that creates blocks which will be concatenated). The timestamp and vehicle identification are not reset between each data block. The data-set is always ordered by timestamp before being passed to the detection algorithms.

We generate two data sets that represent two different events: In the first event, a stopped car that blocks the lane appears before being removed with a traffic getting back to

[2]https://www.ecologique-solidaire.gouv.fr/sites/default/files/Rapport%20GT%20technologies%20STI-vfin.pdf p.18/19

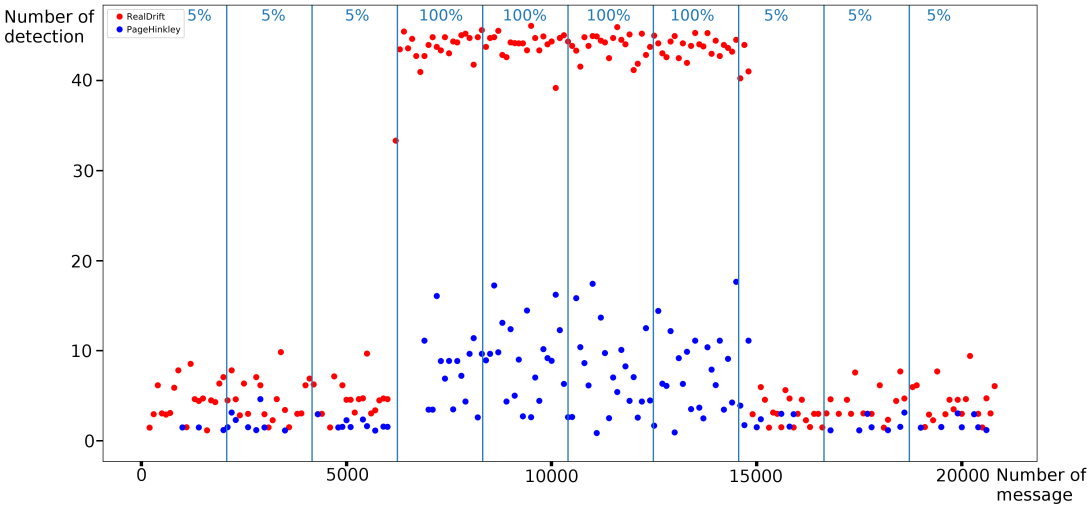

Fig. 2: Detection of Page-Hinkley algorithm for stopped car scenario

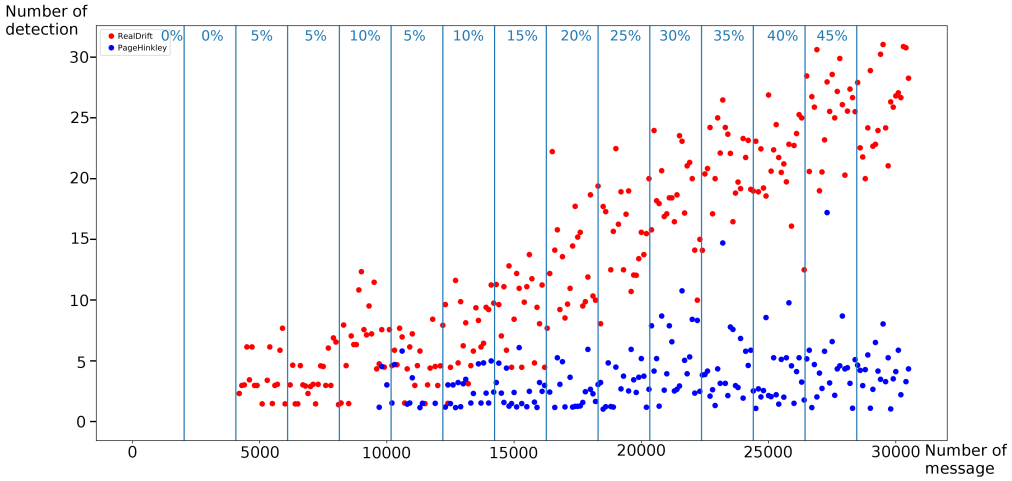

Fig. 3: Detection of Page-Hinkley algorithm for pothole scenario

normal. The second event is a growing pothole that is avoided by an increasing number of vehicles.

The detailed parameters used are presented in table I.

The algorithms, their configurations, and the results they produce will be explained in the next section.

IV. Drift Detection

To detect the avoiding behaviors and the change of frequency of them, we use ADWIN, and Page-Hinkley algorithms.

The data-set previously generated by the data generation tool is sorted by timestamp and then fed to the algorithms. When the algorithms detect a change in the data it is treating, a notification is given.

The notification and the real drifting messages are counted by windows of 600 messages to have a better representation of the results. These results are depicted in the figures 2, 3, 4 and 5. The red dots represent the number of messages with an avoiding behavior in the last 600 messages and the blue dots the number of drift detected by the algorithm. The higher on the y-axis the dots are, the stronger the change is on the period.

We use a python implementation of ADWIN and Page-Hinkley algorithms[3]. A C++ and java implementation of ADWIN developed by the authors is also available[4]. The algorithms are used with the parameters in table II.

Fig. 2 presents the results for the Page-Hinkley algorithm for the stopped car scenario. And Fig. 3 the result of Page-Hinkley on the pothole scenario. The x-axis represents the generation time (corresponding to the number of messages). The y-axis represents, in red, the number of messages in the 600 last

[3]https://github.com/blablahaha/concept-drift

[4]https://github.com/abifet/adwin

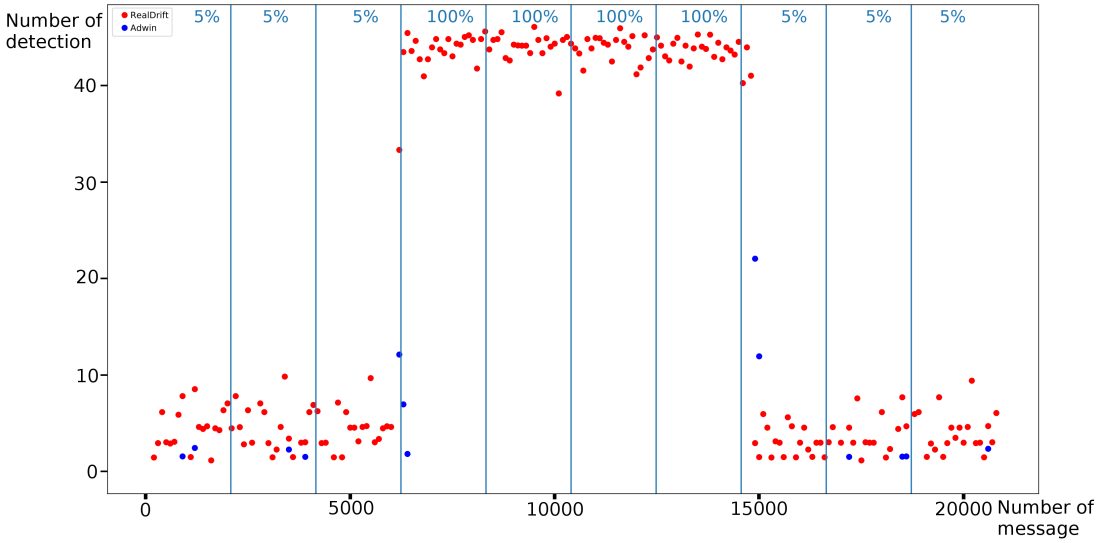

Fig. 4: Detection of ADWIN algorithm for stopped car scenario

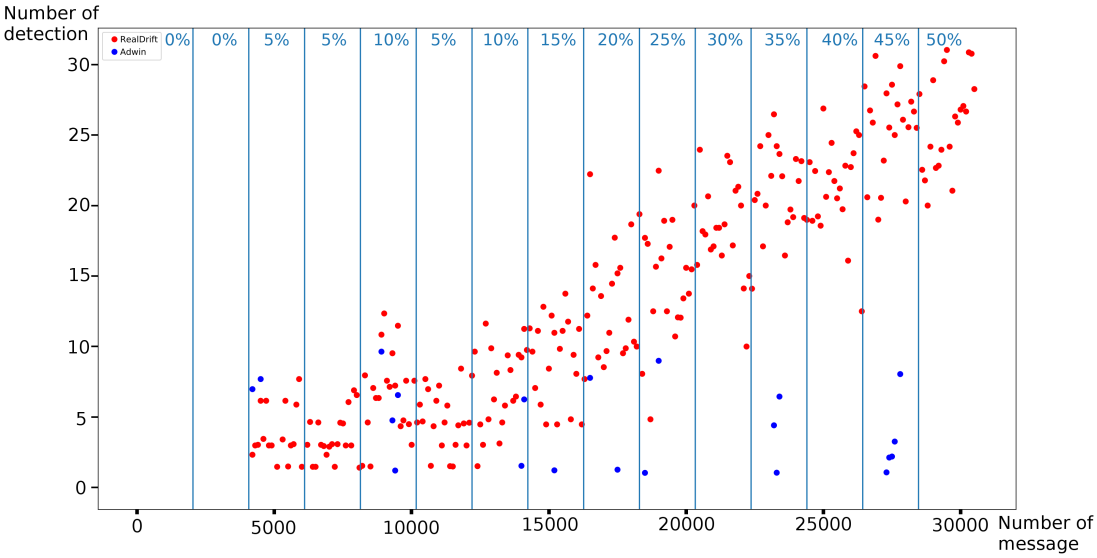

Fig. 5: Detection of ADWIN algorithm for pothole scenario

Scenario	number of chunk	duration of each chunk	number of vehicle per chunk	avoidance rate per chunk
Stopped car	10	10	1000	5, 5, 5, 100, 100, 100, 100, 5, 5, 5
Pothole	15	10	1000	0, 0, 5, 5, 10, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50

TABLE I: Parameters of the data generation

Algorithm	Parameters				
Page-Hinkley	delta 0.01			lambda 200	alpha 0.99
ADWIN	delta 0.0001	maxBuckets 300	minClock 20	minWindowLength 150	minSubWindowLength 75

TABLE II: Parameters used for drif detection

messages that contain an avoiding behavior and in blue the number of detection of this behavior by the algorithm.

For the Page-Hinkley algorithm, it is difficult to have an accurate detection of large changes because the detection is highly delayed. That is why we used parameters to detect the smallest changes. This allows us to track the frequency

of changes in the overtaking behaviors. We can see that the frequency increases as we enter the period when the overtaking rate is the highest. But by design, the Page-Hinkley algorithm has a certain delay in detecting new concepts, so the points do not directly follow the change. The performance of this algorithm is encouraging for the stopped car scenario since we can see the increase in the number of detections when the change occurs. But the delay in the detection of the events is a strong backlash because we need a filtering step that will delay the detection even more. For the pothole scenario, we can see a slight increase in the detection rate of changes with few spikes but this is not enough to be significant. And with the delay in detection, it is not possible to have a correct view of the detection until the concept stabilizes.

Fig. 4 presents the results for the ADWIN algorithm for the stopped car scenario and Fig. 5 the result for ADWIN on the pothole scenario.

For the ADWIN algorithm, in the stopped car scenario, there is little detection in the low avoiding rates, but they do not hide the high number of detections when large changes occur. In this scenario, the results are really convincing and should be tested in real cases. But in the case of the pothole scenario, the detection is not accurate. Initially, the algorithm manages to detect the change, but when the avoidance rate reaches 25%, the algorithm cannot detect the changes correctly. This is because the algorithm is designed to adapt to changes, and then, it fails to detect the next changes accurately because the avoiding behavior has become part of the concept it has learned and the difference in rates is no longer large enough for it to detect them. For this type of behavior, other algorithms may be better suited. Such algorithms should use a window model with a fixed historical window as a basis for learning since we want to detect abnormal behavior compared to typical behavior. But the loss of adaptability to change will require reconfiguration of the history window each time a change is made to the road, its environment, or driver behavior (the latter change could be due to an increasing number of C-ITS, automated vehicles or other technical improvements or recommendations).

V. Conclusion

In this paper, we described how our trajectory generator based on C-ITS standardization works. We then explored our generated data with concept drift detection algorithms to find the generated avoiding behavior. We focused only on the heading of the vehicles to perform our detection. We show that in different scenarios the algorithms we used can perform better or worse. Different parameters or approaches can allow the detection of specific kinds of events. In further work, we plan to use other algorithms and more scenarios. We will also explore different variables such as the speed of the vehicle and combine it with the heading variable.

References

[1] Etsi en 302 637-2; intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service.

[2] Marcelo Keese Albertini and Rodrigo Fernandes de Mello. A self-organizing neural network for detecting novelties. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, page 462–466, New York, NY, USA, 2007. Association for Computing Machinery.

[3] C. S. Arvind and J. Senthilnath. Autonomous vehicle for obstacle detection and avoidance using reinforcement learning. In Kedar Nath Das, Jagdish Chand Bansal, Kusum Deep, Atulya K. Nagar, Ponnambalam Pathipooranam, and Rani Chinnappa Naidu, editors, *Soft Computing for Problem Solving*, pages 55–66, Singapore, 2020. Springer Singapore.

[4] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[5] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3):317–346, 2013.

[6] Lukasz Golab and M Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[7] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[8] B. Leblanc, S. Ercan, and C. de Runz. C-its data completion to improve unsupervised driving profile detection. In *2020 IEEE 91th Vehicular Technology Conference (VTC2020-Spring)*, pages 1–4, 2020.

[9] B. Leblanc, H. Fouchal, and C. de Runz. Driver profile detection using points of interest neighbourhood. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–4, Sep. 2019.

[10] R. Madli, S. Hebbar, P. Pattar, and V. Golla. Automatic detection and notification of potholes and humps on roads to aid drivers. *IEEE Sensors Journal*, 15(8):4313–4318, Aug 2015.

[11] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo. Real time pothole detection using android smartphones with accelerometers. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6, 2011.

[12] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[13] Gordon J. Ross, Niall M. Adams, Dimitris K. Tasoulis, and David J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191 – 198, 2012.

[14] Arthur Yeh, Dennis Lin, Honghong Zhou, and Chandramouliswaran Venkataramani. A multivariate exponentially weighted moving average control chart for monitoring process variability. *Journal of Applied Statistics*, 30(5):507–536, 2003.