Memorias del 2022 Congreso Nacional de Control Automático
12-14 de Octubre, 2022. Tuxtla Gutiérrez, México.
Copyright© AMCA, ISSN: 2594-2492

Real-time Accelerometer-Based Event Detection Through KNN Classification Method

Diego M. Botín-Sanabria, Mauricio A. Ramírez-Moreno, Jorge de J. Lozoya-Santos

School of Engineering and Sciences, Mechatronics Department, Tecnológico de Monterrey, Eugenio Garza Sada 2501 Sur, Monterrey 64849, Mexico. botin@tec.mx (D.M.B.-S.); mauricio.ramirezm@tec.mx (M.A.R.-M.); jorge.lozoya@tec.mx (J.J.L.-S.)

Abstract: Real-time pattern and activity recognition techniques have experienced a recent surge in the use of advanced statistical techniques for analyzing data. More explicitly, Machine Learning classification methods have been used in a variety of applications and for different purposes, such as recognizing different types of events from accelerometer data (e.g. in smart watches, wearables, and vibration analysis). The objective of this work is to present a novel use of accelerometer-based pattern recognition techniques to detect the urban space infrastructure characteristics based on their effect on a vehicle's body frame dynamics. For this study, we focus on detecting speed bumps, potholes and curves based on a real-time data streaming with the objective of rendering a street digitalization to automatically update an urban space Digital Twin. The present study has achieved real time event detection for an urban trajectory with high accuracy which yields data that can be later fused with other sensors data. This approach is effectively contributing to the infrastructure layer of a multi-layered approach of digital modelling. Furthermore, the results contribute to expand the body of knowledge of real time accelerometer event recognition techniques.

Keywords: Machine Learning, Real-time Detection, k-nearest neighbor, accelerometer, space digitalization, digital twin

0.1 Abbreviations and Acronyms

- ML - Machine Learning
- SVM - Support Vector Machine
- KNN - K-Nearest Neighbor
- DT - Digital Twin
- IMU - Inertial Measurement Unit
- NED - North, East, Down
- RMS - Root-mean Square
- PSD - Power Spectral Density
- PCA - Principal Components Analysis
- LIDAR- Light Detection and Ranging

1. INTRODUCTION

This work is part of a larger research project focused on developing Digital Twin (DT) models for urban spaces, vehicle movement, and the interaction of the community with the physical environment. The main motivator of this work is generating sustainable development solutions based on modern technologies to enable data-driven decision-making for the improvement of urban spaces. Statistics from the UN 2020 report on Sustainable Development goals show that a rapid expansion of built-up area (physical expansion of a city) reflects unplanned urban sprawl in the last 20 years. With an average of 133.15 square meters of built area per person in Latin America, this problem directly affects delivery services, public health and mitigation plans for natural disasters (United Nations (2020)). For this purpose, a series of sensing devices are being used to capture data to digitalize and characterize different layers of an urban space: (1) the physical terrain, (2) buildings, (3) infrastructure, (4) the community's mobility, (5) real-time digital input layer, (6) digital twin deployment (Botín-Sanabria D.M. et al. (2021)). Traditional approaches for 3D mapping of urban spaces tend to rely on the usage of LiDAR sensors, but lack the precision to digitalize the conditions of the road. To overtake this obstacle, the present work aims to present the design and implementation of an accelerometer-based method using a KNN classification approach for the characterization of infrastructure in urban spaces. This approach will serve the purpose of accurately representing details of the condition of the street such as speed bumps, potholes, and how the community interacts with the physical space. Information that would not be detected by the other sensors in the system. With this digital layer of a physical space, the system enables a detailed and updated insight of the physical state of a street that will help understand the current situation in order to identify improvement opportunities. The final DT is expected to showcase the results from all data acquisition systems in a clear way to a user. Furthermore, the complete system is design to take data from other sensors to understand how the community interacts with the physical space.

Classification tasks for detecting patterns, events or certain activities have experienced a surge since the integration of advanced statistical techniques and algorithms that integrate Machine Learning (ML) predictive models. The existing ML models have the capability of identifying complex relations and nonlinearities in data sets and provide a deeper insight which facilitates data analytics and pattern recognition (Ferrahi V. et al. (2019)). However, several obstacles and implications arise when using ML algorithms for this purpose due to the increased complexity of systems. There are several types of predictive models such as Neural Networks, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Decision Trees. In this work, a study of classification models was conducted and the chosen one is a KNN model with optimized hyperparameters. This model "is a popular classification method in data mining and statistics because of its simple implementation and significant classification performance" (Zhang S. et al. (2018)). In essence, the method employs a certain distance metric and weight to calculate the distance between a test sample (new data) and all training samples in order to obtain their nearest neighbors and then conduct the classification (Zhang S. et al. (2018). In this sense, this type of model has been chosen to detect events from a driving trajectory with the objective of characterizing the current state of the infrastructure in an urban space. For this purpose, the system should have a stream of inertial data from the car´s movements as input and should give a clear results of the detected events for the analyzed trajectory. This work presents the graphic results through time but there is work being carried out to represent the same results through the distance traveled.

This paper is organized as follows: Section 2 presents the problem statement; Section 3 presents the methodology for performing event recognition from an inertial measurement unit (IMU) device; Section 4 presents the results from the design, deployment and testing of the detection model and Section 5 discusses the most important takeaways and implications from the results.

2. PROBLEM STATEMENT

There is a need of characterizing the infrastructure of the urban space to detect anomalies on the road and make an accurate virtual replica of it. With this task at hand, the objective of this work is to present the design and deployment of a ML-based classifier capable of detecting certain events on the road based on the vehicle dynamics. Activity or event recognition is a challenging method to generalize due to the unique characteristics of each event (Sukor A.S. et al. (2018)). However, the classifier is expected to generate actionable data to identify specific conditions from a physical space in real-time. The future output enables an external user to remotely analyze the infrastructure state on an area supplied by data from a fleet of vehicles like in the case on urban areas.

3. METHODOLOGY

The development of ML models consists of: (1) data acquisition, (2) model training and testing, and (3) model deployment (see Figure 1). The data acquisition phase consists mainly of using sensors to gather information that represents the environment where certain variables are to be measured and recorded (e.g. gathering IMU data from a driving cycle for street events of interest). The model training and testing phase is conformed of three subphases: (1) data preprocessing, (2) extracting features and predictors, and (3) training, validation and optimization of the model. Then, there is the deployment phase when the ML model detects the specific activity it can be deployed on the field.

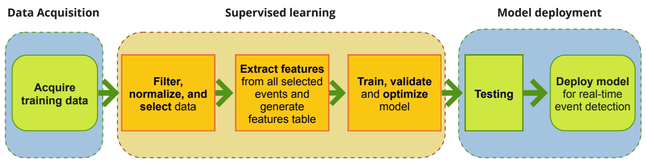
Data Acquisition
Supervised learning
Model deployment
Acquire training data
Filter, normalize, and select data
Extract features from all selected events and generate features table
Train, validate and optimize model
Testing
Deploy model for real-time event detection

Fig. 1. Machine Learning model phases.

In the process of selecting features and training a model, known techniques are supervised methods, semi-supervised methods, and unsupervised methods (Miao J. & Niu L. (2016)). In the case of supervised training and feature selection, supervised methods allows to "effectively select discriminative and relevant features to distinguish samples from different classes" (Miao J. & Niu L. (2016)).

Given the objective of this work, several ML models were researched and explored to find the most adequate one to perform the task of classification given a set of sensor signals. This model must be fast (in terms of computational time), and accurate to ensure real-time detection. Meaning it should be able to process a 3 second time window fast as to provide a detection result without causing delays in the sensor streaming measurements. For this, many tools are available now days. MathWorks Inc. provides a the Classification Learner App (MathWorks Inc. (2022)) in Matlab, where it is possible to train several models and optimize them in a very efficient way. Using this tool, models such as Fine Tree, KNNs, SVMs, Neural Networks, and Naive Bayes were trained and compared in terms of accuracy. The chosen model was an optimized KNN model which is discussed in Subsection 4.1.

3.1 Experimental Setup

To validate the performance and accuracy of the model in real-world scenarios and as a first step to final system implementation, tests were conducted in different streets close to the Tecnológico de Monterrey, Monterrey Campus area. Testing trajectories were chosen with the objective of putting to test the model in detecting the four cases (speed bumps, normal driving, curves and potholes). In this sense, 6 trajectories (of approximately 500 m each) were chosen which contain all events and accurately represent a normal drive cycle in the area. The experiment consisted on traveling the set trajectory at different times of the day on different days (to eliminate any bias related to traffic conditions)and correlating the real (i.e. true classes) vs detected events and their score percentage to validate the model's performance. All tests had a set duration of 1 minute with a sampling frequency of 50 Hz and with control variables such as the IMU sensor, the sensor's location on the vehicle, driver, vehicle and driving style (provided it was the same driving conditions).

In terms of the location of the sensor, it was positioned as close as possible to the center of gravity of the vehicle. This is important in order to eliminate linear acceleration bias when representing the vehicle's movement.

3.2 Data Acquisition System

The system consists of a set of devices: (a) an IMU, (b) a computer for real-time processing of the model, (c) a vehicle, (d) an Arduino Nano (Arduino, United States, Somerville) to enable connectivity between the sensor and the processing unit. The IMU device is the MPU9250 (IvenSense, United States, California) mounted on the roof of the vehicle. It is housed on a designed support that is magnetically attached to the vehicle to ensure stability and accessibility. The MPU9250 is an IMU that integrates accelerometer, gyroscope and magnetometer sensors that measure linear acceleration [m/s^2], angular velocity [rad/s], and magnetic field intensity [μT] on 3 axis (X, Y, Z) respectively.

When collecting IMU signals, they are calibrated to eliminate the effect of gravity (in the Z axis), to align the axes in accordance with NED (North, East, Down) coordinates, and to eliminate hard iron distortions in the magnetometer (MathWorks Inc. (2022)). The NED coordinate system ensures that the IMU is aligned with the Earth's geodetic directions (north pole, east, downward). Furthermore, it is important to note that the signals for training have not been filtered to provide a training data set that more accurately represents real-life an accelerometer signals from a moving car. Authors in (Sukor A.S. et al. (2018)), mention that features need to be selected from raw signals in order to increase recognition accuracy and computational efficiency.

3.3 Feature extraction and predictor selection

Using a supervised learning method, accelerometer signals for a specific time window are evaluated in time and frequency domains to extract 5 features for each axis (X, Y, Z) of acceleration. These time windows are manually selected from signals to represent the specific events that the model should detect. Therefore, the model can be trained with 3 positive cases (speed bumps, curve, and potholes) and a normal driving case for uneventful time windows. The features being extracted are the following:

- Arithmetic mean ($\overline{x}$).
- Root-mean square value (RMS).
- Power spectral density (PSD) estimate using Welch's method (Pw): power and frequency for 6 highest peaks.
- Autocorrelation features (covariance) (R): magnitude of main peak, and magnitude and frequency of second peak.
- Total power (Pf) in 5 adjacent and pre-defined frequency bands (0.5, 1.5, 5, 10, y 15 Hz).

In summary, this matrix of predictors contains the following values for each axis: the arithmetic mean and RMS, 6 power magnitudes and their frequencies (PSD), 3 covariance values, and 5 power values for pre-defined frequency bands giving a total of 66 predictors. This matrix is X_{nx66} where n represents the number of sample data points or cases with which the model is trained (rows) and 66 is the number of predictors extracted from each case (columns). Each of the five predictors already mentioned and shown in Equation 1, are calculated for all three axis of the accelerometer.

$$\mathrm{X}_{nx66} = \begin{bmatrix} \overline{x}_1 & RMS_1 & Pw_1 & R_1 & Pf_1 \\ \overline{x}_2 & RMS_2 & Pw_2 & R_2 & Pf_2 \\ . & . & . & . & . \\ . & . & . & . & . \\ \overline{x}_n & RMS_n & Pw_n & R_n & Pf_n \end{bmatrix} \quad (1)$$

For this application, being that training data does not have a high dimensionality, the Principal Components Analysis (PCA) process has been discarded. However, the Classification Learner App enables features selection algorithms to analyze the importance of features for the prediction of the response variables. For this work, the ReliefF algorithm which is a type of filter method for feature selection (Urbanowicz R. et al. (2018)), was implemented since it is best for estimating feature importance for distance-based supervised models such as a KNN model. The results for this algorithm have concluded that features from the frequency domain analysis of the data are more relevant for prediction. The reduction of features based on this algorithm is yet to be implemented in the system. The expected result is increasing the computational efficiency and performance of the system.

Response or target values are the classes where events have been classified. For instance, the algorithm is designed to detect speed bumps, normal driving, potholes, and curves. Their respective numerical labels are 1, 2, 3, and 4. The vector of response variables is given by Θ_n.

$$\Theta_n = \begin{bmatrix} \theta_1 \\ \theta_2 \\ . \\ . \\ \theta_n \end{bmatrix} \quad (2)$$

Note that to provide faster model processing with the objective of full filling real-time detection, classes have been numerically labeled. Table 1 presents this labeling where each class was given a specific numerical value. The first type of event denoted as "None" corresponds to predictions made by the model that do not satisfy a score threshold of 70%. This value of threshold was chosen based on experimentation. This means that through various experiment setups and tests, supported by results validation tasks, it was concluded that if the score was less than 70% the algorithm will sometimes provide false positives.

Table 1. Numeric class labels.

Type of Event	Numeric Label
None	—
Speed bump	1
Normal driving	2
Curve	3
Pothole	4

For the final step of data processing prior to model training, the features matrix was generated by horizontally concatenating X_{nx66} and Θ_n in data set $Features$ which contains n rows corresponding to the number of training samples (i.e. observations) and 67 columns corresponding to 66 features and response for each sample.

$$\text{Features}_{nx67} = [\ X|\Theta\ ] \quad (3)$$

4. RESULTS

Prior to selecting the KNN model, various types of models were trained with the same features matrix to explore their prediction accuracy. In this sense, some of the tested models and their validation accuracies were the following. Note that other models were trained but have been excluded from this list.

- Weighted KNN - 95.3%
- Fine KNN - 92.2%
- Optimizable KNN - 96.9% (chosen model)
- Linear SVM - 95.3%
- Trilayered Neural Network - 95.3%
- Fine Decision Tree - 90.6%

Based on this comparison and the prediction speed, the optimizable KNN model was chosen to continue.

4.1 KNN Optimized Model

Through the Bayesian optimization method with 30 iterations, the KNN model was optimized to increase accuracy in classification. Fig. 2 shows that the bestpoint hyperparameters were reached after 7 iterations where both the estimated and the observed classification error were minimized, thus, yielding an acceptable observed minimum classification error of 0.036.

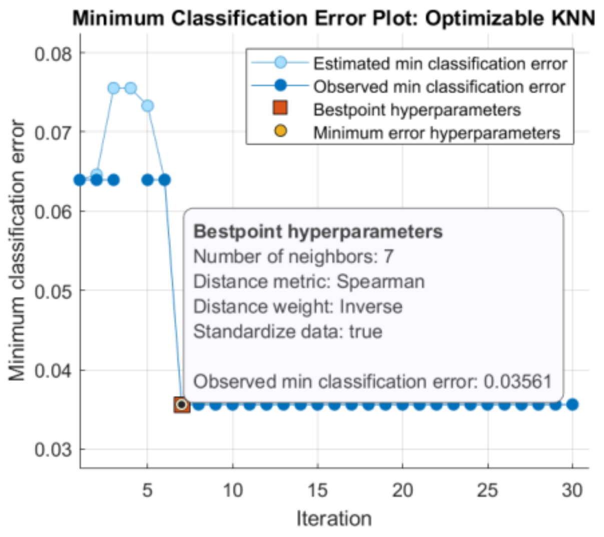

Fig. 2. Minimum Classification Error Optimization Graph.

The optimized hyperparameters calculated were the following:

- Number of neighbors: 7
- Distance metric: Spearman
- Distance weight: Inverse
- Standardize data: true

Through a cross-validation scheme of 3 folds, the data set was partitioned for validation and testing. The model got a calculated validation accuracy of 96.9% and the validation confusion matrix is shown in Fig. 3.

Furthermore, once the KNN optimized model has been trained, a scatter plot of the model's predictions was generated to analyze classification errors in a graphical way such that class clusters are made evident. In Fig. 4, two features (X-axis RMS and Y-axis RMS) have been plotted for all observations. Each class has been given a color and all dots represent correct classifications of events. Incorrect classifications are represented with crosses.

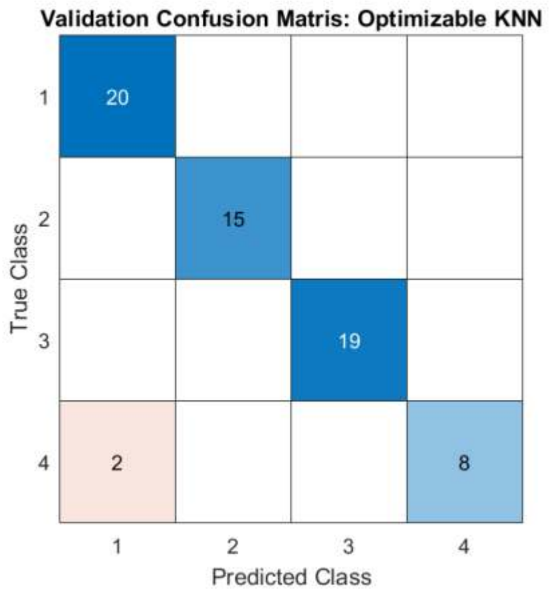

Fig. 3. Validation confusion matrix. Numeric labels correspond to the following events: (1) Speed Bumps, (2) Normal Driving, (3) Curves, (4) Potholes.

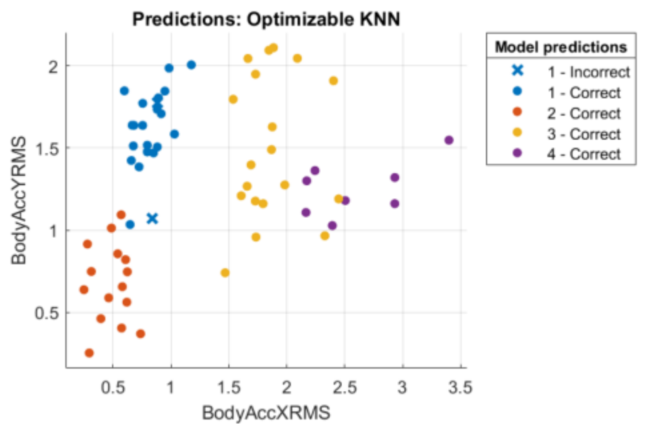

Fig. 4. Scatter plot results of prediction with validation set.

4.2 Real-time testing

In terms of the user interface, a Matlab script with graphical interface elements was developed to run in real-time using a fixed time window for event detection. In this sense, the user may modify some parameters prior to initialization such as the time window duration (set at 3 seconds by default), the duration of the experiment (set at 1 minute by default), and the sampling frequency for the IMU (set at 50 Hz by default) (Ferrahi V. et al. (2019)). The time window of 3 seconds was set through experimentation and testing. It was concluded that all the event that the model is design to detect can happen within a time window of 3 seconds. Furthermore, it was observed that if the time window is too small (less than a second), the algorithm ran into delays due to computing power as there is a time needed to process the information before a new time window is tested. When the time window was too large (over 5 seconds), there was a possibility that two or more events that were contiguous in physical space, will be present in the measurements for the same time window. In this sense, the algorithm is not capable of detecting more than one event per time window.

When the script is run, the user is prompted to click the start button whenever they are ready. Once it has started, a popup window will inform the user of the current time window, the detected event and the score for said event. When the script reaches the set experiment duration, it will cease to stream sensor signals and it will present the user with a series of figures such as Fig. 6 where they can correlate the model's predictions with the real sensor signal measurements, and a histogram that summarizes the detected events for the duration of the experiment.

The model provides as output 3 results:

- **Predicted Event:** It is presented as the numerical label corresponding to each event.
- **Prediction Scores:** Are the percentages of certainty of prediction for each event. There is a value assigned for each event so this variable contains a $1x4$ vector. The class with the highest score is taken to be the predicted event since it means it has the highest level of certainty.
- **Post Probabilities:** These are merely the percentages of error of the model when predicting a specific event. This variable contains a $1x4$ vector containing the error of prediction for each event. This means that the class with the least amount of error is selected as the predicted event $(1 - score)$.

For the final real-time model deployment, a score threshold of 70% was set to ensure that the detected events had the highest score of certainty.

A summary of all trajectories and their specific infrastructure characteristics is presented in Table 2 where each predicted event has a calculated score from the model. Furthermore, Fig. 5 illustrates the area around the Tecnológico de Monterrey campus where testing has taken place. For instance, the dotted line along the central campus corresponds to the testing trajectories for this work. Note that the vehicle direction is always clockwise along the streets in Figure 5.

The following results correspond to the second trajectory of testing (Eugenio Garza Sada - Fernando García Roel street - see Table 2). From this trajectory, the user interface designed in Matlab, allows a user to analyze the recorded signals from the specific trajectory as presented in Fig. 6 (top). Furthermore, the system displays the prediction results and the individual prediction scores to the user as shown in Fig. 6 (middle and bottom respectively). As seen, there is a threshold limit shown as a dotted line along the 70% mark in the bottom graph and any event above the threshold will be selected as the

Table 2. Testing results summary.

#	Trajectory	Real events	Predicted events	Score
1	Luis Elizondo street	Speed bump	Speed bump	84.33%
2	Eugenio Garza Sada - Fernando García Roel	Curve Speed bump Speed bump	Curve Speed bump Speed bump	100% 88.28% 100%
3	Junco de la Vega	Curve Speed bump	Curve Speed bump	100% 85.55%
4	Luis Elizondo street	Speed bump	Speed bump	100%
5	Eugenio Garza Sada street	Speed bump	Speed bump	100%
6	Fernando García Roel street	Speed bump	Speed bump	86.95%

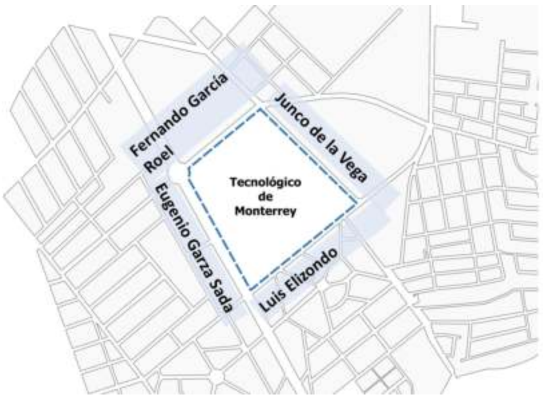

Fig. 5. Tecnológico de Monterrey, Campus Monterrey map.

predicted event. In this test, the model detected a speed bump in time window 9 that did not full fill the threshold limit. This specific time window will be labeled with a "None" event detected label and no specific numerical label.

5. DISCUSSION AND FUTURE WORK

There have been 3 main takeaways with regards to the use of ML classifiers for the task of detecting events on the road: (1) frequency domain features have proven to be more suitable for detecting the desired events in this study, (2) Although other models proved to yield comparable detection accuracy, the KNN model was selected based on its characteristics, increased accuracy and efficiency, (3) ML is a powerful tool when it comes to classification capabilities in applications of event or activity detection. In this work, the model has been trained to detect based on accelerometer data; however, as part of future work, other activities may be included in the detection algorithm such as driving style, speed humps, and even accident analysis. This could be implemented using the other sensors in the IMU such as the gyroscope and magnetometer.

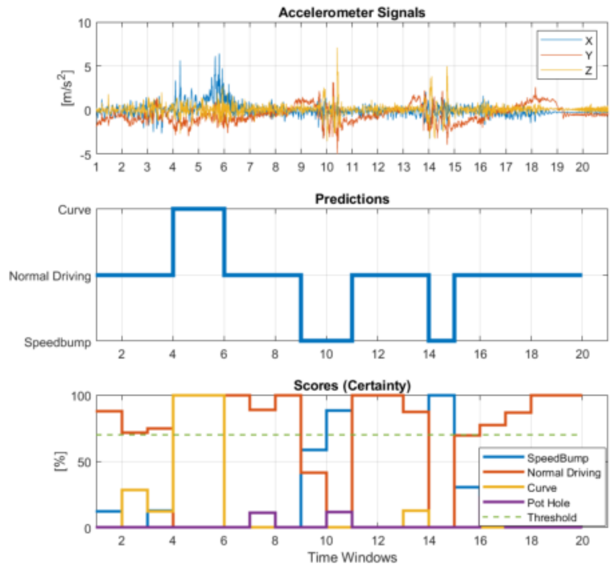

Fig. 6. Trajectory 2 testing results - sensor signals (top), model predictions (middle), prediction scores (bottom).

Furthermore, the next steps for this work are to finish validation and evaluation testing in order to start the integration of this model to the digital twin system for urban space digitalization. In this way, the discussed model will interact with a LIDAR mapping system that is designed to generate a 3D point cloud digitalization of a physical space, and to calculate an estimated trajectory for the vehicle within the space. By enabling this fusion, the system will be able to determine not only at which point in time an event was detected, but also the location of such event. This will effectively yield a more accurate and insightful digitalization of the urban space and its infrastructure. Furthermore, by incorporating the detection of driving styles, the mobility layer of the digital twin architecture may be targeted to generate information on the interactions of driver with the urban infrastructure.

REFERENCES

United Nations The Sustainable Development Goals Report 2020. *UN Department of Economic and Social Affairs*; 2020.

MathWorks Inc. Estimating Orientation Using Inertial Sensor Fusion and MPU-9250. *Matlab Documentation*; 2022; R2022a.

MathWorks Inc. Clasification learner. *Matlab Documentation*; 2022; R2022a.

Zhang S.; Li X.; Zhong M.; Zhu X. & Wang R. Efficient kNN Classification with Different Numbers of Nearest Neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29; 2022.

Farrahi V.; Niemela M.; Kangas M.; Korpelainen R. & Jamsa T. Calibration and validation of accelerometer-based activity monitors: A systematic review of machine-learning approaches. *Gait & Posture*, vol. 68, pages 285-299; 2019.

Miao J. & Niu L. A Survey on Feature Selection. *Procedia Computer Science*, vol. 91, pages 919-926; 2016.

Sukor A.S.; Zakaria A. & Rahim N. Activity Recognition Using Accelerometer Sensor and Machine Learning Classifiers. *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*; 2018.

Botín-Sanabria D.M.; Lozoya-Reyes J.G.; Vargas-Maldonado R.C.; Rodríguez-Hernández K.L.; Ramírez-Mendoza R.A.; Ramírez-Moreno M.A. & Lozoya-Santos J.J. Digital Tiwn for Urban Spaces: an Application. *Proceedings of the International Conference on Industrial Engineering and Operations Management, Monterrey, Mexico*; 2021.

Urbanowics R.; Meeker M.; La Cava W.; Olson R. & Moore J. Relief-based feature selection: Introduction and review. *Journal of Biomedical Information*; 2018.