Modeling Vehicle Fall Detection Event Using Internet of Things

Nikhil Kumar[1(✉)], Anurag Barthwal[1], Debopam Acharya[2], and Divya Lohani[1]

[1] Department of Computer Science and Engineering, Shiv Nadar University, Gautam Buddha Nagar, Greater Noida, Uttar Pradesh, India
{nk438,ab414,divya.lohani}@snu.edu.in
[2] School of Computing, DIT University, Dehradun, Uttarakhand, India
dr.debopamacharya@dituniversity.edu.in

Abstract. Research in the realm of accident prediction and analysis is mostly confined to the study of road accidents caused by motor vehicle collisions where the event of vehicle fall from high altitude is mostly ignored. An IoT system has been introduced in this work, which uses a budget smartphone and off-the-shelf sensors to accurately detect and notify vehicle fall event. The proposed system is low cost, reliable and can be easily retrofitted to any type of vehicle. It uses the vehicle speed, absolute linear acceleration and altitude to build an SVM classifier based model to detect the fall event. The proposed IoT system is found to be accurate with a MAPE of 1.8%.

Keywords: Vehicle fall detection · Accident detection · Context-aware · Internet of Things · Support Vector Machine · Absolute linear acceleration

1 Introduction

According to Global Status Report on Road Safety 2018 by World Health Organization (WHO), the total number of deaths due to road accidents is excessively high with more than 1.35 million people dying each year worldwide [1]. For all kind of death causes, road accident is the eighth biggest reason for the death of people and the biggest reason for the death of children and youth aged below 29 years [1]. According to the golden hour principle (the relationship between the time taken for the treatment after an accident and the death rate), timely notification reduces the time for medical treatment after the accident and significantly decreases the mortality rate [2]. Research in the area of accident prediction has mainly focused on the use of information and communication technologies to detect vehicle collisions. Such accident detection systems fail to report the accident of a vehicle due to a fall from a bridge, fly-over or a hill. A significant percentage of road accidents has been recorded when the vehicle either goes off the road or falls-off from an altitude. Such events may occur when the vehicle is at an altitude from the ground, plying on a bridge, fly-over, elevated highway or is traversing a hilly road. According to the road accident database management system (RADMS) report of the government of the State of Himachal Pradesh (HP) in India [3], from April 2018 to June

S. M. Thampi et al. (Eds.): SIRS 2019, CCIS 1209, pp. 220–233, 2020.
https://doi.org/10.1007/978-981-15-4828-4_19

2018, a total of 9076 accidents were reported in HP, out of which 1850 accidents were run-off-the-road accidents. In more than 80% of reported ran-off-the-road accidents, vehicle fell into the gorge. To the best of our knowledge, no research work has been carried out to detect and report such fall events.

In this work, the authors present an end-to-end IoT system which can detect the fall of a vehicle from an altitude. The IoT system can generate an instant alert to the nearby emergency medical services (EMS), relatives of the victims and police so that immediate help can be provided. The novelty of the approach lies in (a) its capability of using low-cost hardware and its ability to be retrofitted to any vehicle, (b) use of statistical models in accurately predicting the vehicle fall event, and (c) automatic reporting of the accident to relevant agencies.

2 Related Work

In this section, the latest research related to the detection of a road accident with the help of inputs from sensors is discussed. Jair Ferreira Júnior et al. [4] have per-formed driver behavior profiling by using different smartphone sensors and classification algorithms. The results have been compared to determine the optimum meth-od to characterize driver aggressiveness profile. Aloul et al. [5] have developed a smartphone-based system that uses accelerometer data to build a predictive model based on Dynamic Time Warping (DTW) and Hidden Markov Models (HMM).

Linayage et al. [6] have proposed a Bayesian quickest change detection approach to optimize the trade-off between average detection delay and false alarm rate. To reduce the latency in reporting an accident, Dar et al. [7] have proposed a fog computing-based approach to develop a low-cost, delay-aware system for detection and reporting of an accident. A smartphone-based early recognition system has been developed by Xu et al. in [8], which can help to avoid vehicle accident by identifying inattentive driving at early stage and alerting the driver.

Park et al. [9] have used in-built sensors of the driver's smartphone to develop an event-driven solution to prevent distracted driving. Bhatti et al. [10] have used speed, pressure, gravitational force, location, and sound sensors of a commodity smartphone to develop a low cost, portable solution that detects an accident and reports it to the nearest hospital.

In all these works, sensors of the passenger's smartphone have been used to build the accident detection system. None of these works has provided analysis and solution for the scenario where the passenger vehicle falls-off from a certain height. Hence, this work presents an IoT system, which uses in-built as well as connected sensors to build an intelligent system that can accurately detect the fall of a vehicle from an elevation.

3 Context-Aware IoT System Architecture

3.1 System Architecture

To address the problem of vehicle fall detection, a novel architecture has been proposed as shown in Fig. 1. Modern smartphones are furnished with a variety of in-built

sensors that can be used to measure different physical parameters. These devices can provide a commanding, economical and versatile research platform for data collection and edge computing. Our proposed IoT system can exploit the five standard sensors of a SAMSUNG Galaxy S8 smartphone (microphone, GPS, magnetometer, accelerometer, and gyroscope) and seven in-built (pressure, temperature, infrared, humidity, CO, illuminance, altitude) and one supplementary (CO_2) sensor of Sensordrone [11], and off-the-shelf sensor device.

In this system, the research focus will be with absolute linear acceleration, the altitude from sea level and speed of the vehicle to detect the fall of the vehicle from a

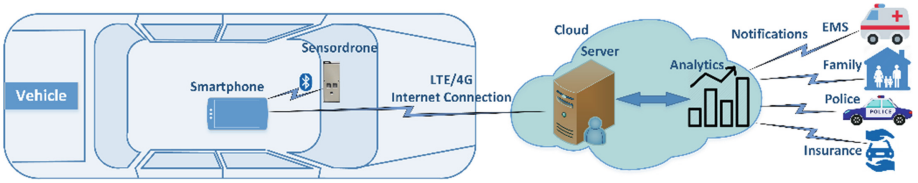


Fig. 1. Context-aware IoT system architecture

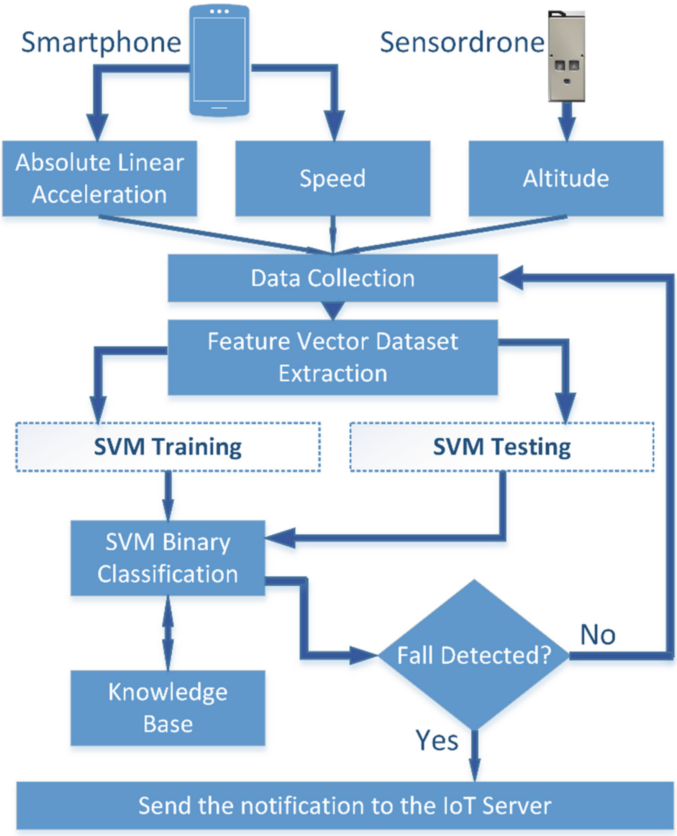


Fig. 2. Working flowchart of the system

certain height. Sensordrone's barometer communicates the atmospheric pressure to the smartphone via Bluetooth link. Atmospheric pressure is used to calculate the altitude of the vehicle with the help of a standard formula. Altitude measurement is fused with absolute linear acceleration and speed to accurately estimate the fall detection scenario.

This whole processing of sensors' streamed data is done within the smartphone itself. In the case of the vehicle fall event, the smartphone immediately sends the fall details like time, coordinates, device ID, etc. to the IoT server in the cloud for further processing. The smartphone uses a 4G/LTE connection to send this valuable information to the IoT server. After analyzing the received data, the IoT server sends the notification with the location and other relevant data to the nearest EMS, family and other emergency services such as police, insurance, etc. As shown in Fig. 2, absolute linear acceleration and speed parameters are obtained from inbuilt sensors of the smartphone while the information about altitude is derived from the Sensordrone. Features extracted from these input parameters are used for training and testing of the SVM classifier model. Occurrence of the fall event is notified to the end users. The process keeps reiterating until either the system shuts down or a vehicle fall event is detected.

3.2 Hardware Setup

Samsung Galaxy S8 Smartphone. Samsung Galaxy S8 smartphone is equipped with a 1.9 GHz Exynos 8895 octa-core processor, 4 GB of RAM and 64 GB of storage, which makes it a suitable platform for edge processing. It houses multiple standard sensors such as inertial sensors, microphone, GPS, light sensor, proximity sensor, etc. Proposed system utilizes two standard sensors of the smartphone: LM6D1 six-axis inertial sensor (with range $\pm$ 16 g) for measuring the absolute linear acceleration and BCM4774 GPS sensor for measuring the location of the fall and speed of the vehicle.

Sensordrone. Sensordrone is a small sensor hub that can be programmed according to the research needs. It can measure eleven different environmental parameters including atmospheric pressure, altitude, temperature, humidity, non-contact object temperature, CO, illuminance, proximity, capacitance, oxidizing and reducing gases. External devices can also be added using the serial port expansion. It can be connected easily to any android smartphone via Bluetooth 4.0 link. Android/Java library for Sensordrone is available at [12]. The setup of the experiment has used the barometer LPS331AP (range 26 kPa to 126 kPa) of the Sensordrone to measure the altitude of the vehicle.

RC Car. As it is not feasible to perform the experiment on a full-size, real vehicle and collect streaming data, we have used a well-configured 1:12 scaled RC car to imitate a real-life vehicle fall from an altitude. Configuration of RC car is shown in Table 1. This off-road toy monster truck is made up of high-performance ABS material and has a strong front bumper structure and very good suspension.

Table 1. RC Car configuration [13]

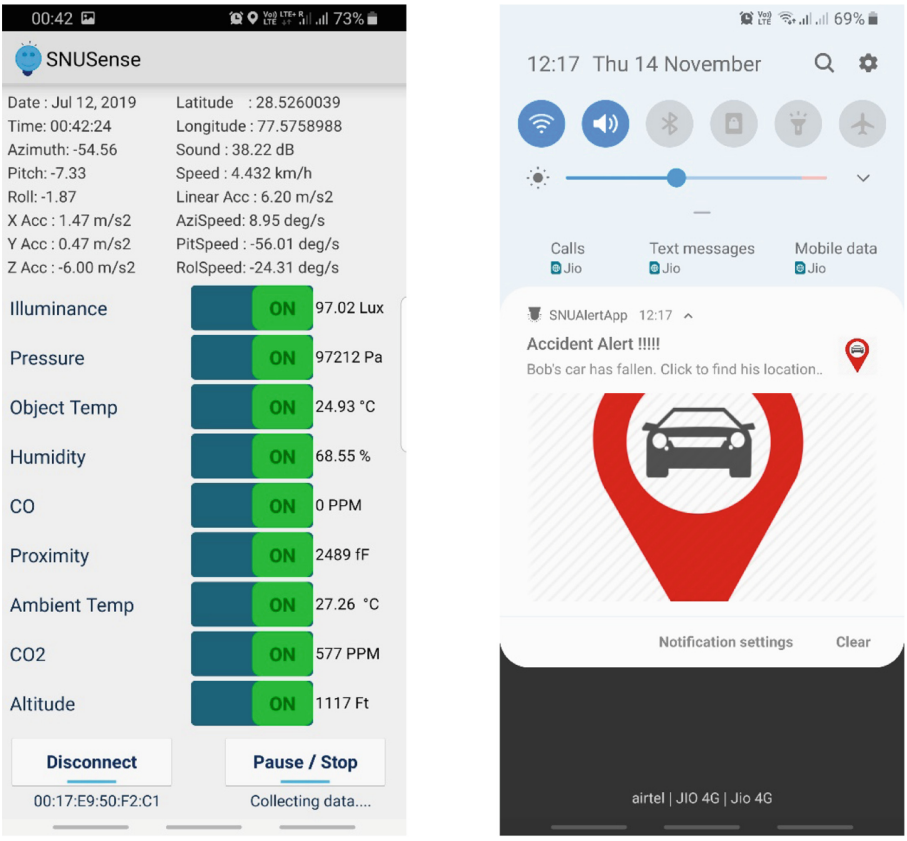
Vehicle Parameter	Value
Brand Name and Item No.	GPTOYS Foxx S911 RC Car
Dimension (L * W * H)	310 mm * 265 mm* 150 mm
Weight	1078 gm
Scale	1:12
Track Width	220 mm
Wheelbase	207 mm
Ground Clearance	40 mm
Max Speed	33 MPH
Transmitter	2.4 GHz 4Ch
Operating Range	80 m
Max Turning Angle	45°

3.3 Software Setup

Android application, SNUSense, has been developed for collecting and processing the received data from smartphone and Sensordrone. SNUSense has a user-friendly interface as shown in Fig. 3(a).

SNUSense collects the data and stores it in a CSV file in the memory of the smartphone. It continuously processes the streamed data and makes an inference using three attributes namely speed, absolute linear acceleration, and altitude of the vehicle. After successfully identifying the vehicle fall event, SNUSense sends the identity of the user and location of the vehicle to the SNUSense IoT server in the cloud (Google Firebase [14]). SNUSense has a broad vision and is developed by keeping in mind other kinds of vehicle accidents such as collision, rollover, fire, etc. As shown in Fig. 3(a), by exploiting sensors of smartphone and Sensordrone, SNUSense can measure time, date, location, sound, speed, linear acceleration, yaw, pitch, roll, illuminance, pressure, temperature, humidity, CO, CO_2, proximity, object temperature, and altitude.

We have developed another android application, SNUAlertApp, for receiving the alert notifications of accident as shown in Fig. 3(b). The notification contains the location of the accident, name of the driver, and type of the accident. A single tap on notification area opens the Google Map with location marker so that the user can trace the accident location easily.



(a) **(b)**

Fig. 3. (a). Screenshot of SNUSense App, (b). Screenshot of SNUAlertApp Notification

4 Testbed, Data Sensing and Pre-processing

4.1 Experiment Setup and Data Collection

A high-speed 1:12 scaled RC car has been used for the experiment to emulate real-life scenarios of vehicle fall. As shown in Fig. 4, the smartphone and the Sensordrone have been tied up with the chassis of the RC car because the optimum value of accelerometer can be obtained when it is placed near the center of gravity (CG) of the vehicle.

The experiment has been performed at 15 feet elevated running track at Indoor Sports Complex of the university (shown in Fig. 4). Multiple experiments have been performed to collect data for characterizing the vehicle fall event. RC car was operated using a 2.4 GHz 4Ch wireless transmitter and dropped on a badminton wooden court by turning it to the left on the track at a 45-degree angle. The trends of ALA, altitude, and speed of the vehicle are shown in Figs. 5, 6 and 7 respectively.

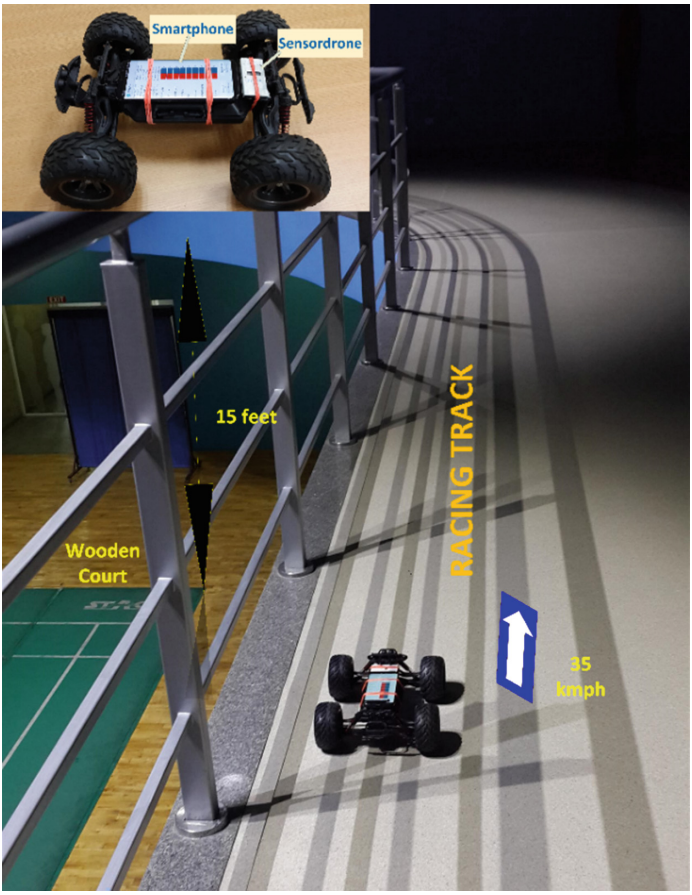


Fig. 4. Experiment Setup

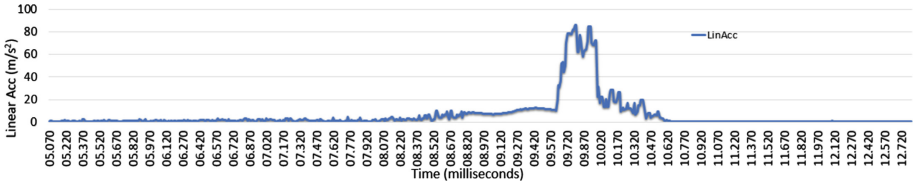


Fig. 5. Linear acceleration after fall from 15 feet elevated track

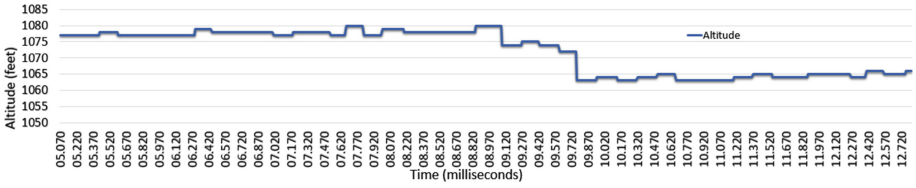


Fig. 6. Altitude measured after fall from 15 feet elevated track

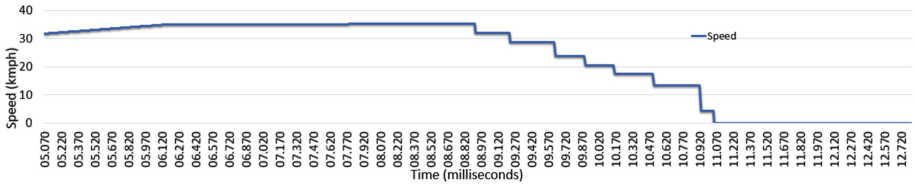


Fig. 7. Change in speed after falling from 15 feet elevated track

4.2 Parameters Used for Fall Detection

In this work, three attributes (viz. absolute linear acceleration, change in the speed and altitude of the vehicle) are used to detect the fall event of the vehicle. If we do not consider any one of them, the system may generate a false alarm.

Absolute Linear Acceleration (ALA). When a moving vehicle falls from a height, the orientation of the vehicle does not remain the same as on the road. In such a situation, it can not stay parallel to the gravitational axes and the acceleration characteristic can be distributed over two or three axes. The static acceleration value on different axes can also change with the change in the orientation of the vehicle. Therefore, it is very difficult to estimate the exact peak of the acceleration of particular axes. To deal with this problem, ALA (or Signal Magnitude Vector [15]) is calculated from all axes X, Y, and Z. ALA is independent of the fall orientation of the vehicle and shows the acceleration characteristic parallel to the gravitation. It is the resultant vector of accelerations of X, Y, and Z-axes. It is the square root of the sum of squares of accelerations at X-axes (ACC_X), Y-axes (ACC_Y) and Z-axes (ACC_Z) respectively. It is a positive quantity, which is calculated by the following equation:

$$ALA = \sqrt{(ACC_X)^2 + (ACC_Y)^2 + (ACC_Z)^2} \tag{1}$$

The measurement unit of ALA is g, where g is equal to 9.80665 m/s^2. In this work, the threshold value for ALA is considered 6 g for fall event detection because it is known from the literature that if a vehicle impact with a non-movable obstacle with more than 23 km/h (i.e. 21 feet/s) then its deceleration always crosses 5 g [16]. It can be demonstrated that when an object falls from a height of more than 10 feet, its final velocity is always more than or equal to 25 feet/second.

Speed. It is observed that when a vehicle falls from a height, it will eventually stop and its speed would eventually become zero. The system has used GPS to measure vehicle speed. Every GPS device receives NMEA (National Marine Electronics Association) sentences from the satellites, which contains data related to position, velocity and time of the device [17]. Each device category has its own NMEA sentence. Each standard NMEA sentence has a two-letter prefix (e.g., GPS receivers uses GP), all proprietary sentences start with P and are followed by three letters, which identifies the device manufacturer (e.g. Garmin sentences start with PGRM). GPRMC, The Recommended Minimum, is the NMEA sentence, which is used for obtaining the velocity by most of the

Android devices. Android library has specific functions to fetch speed from the NMEA sentence. An example of GPRMC NMEA sentence is shown below:

```
$GPRMC,122844,A,1831.336,N,07734.332,E,018.9,054.4,280919,
003.1,W*6A
```

Here, velocity is 018.9 knots, which is 35 km/h.

Altitude. When a vehicle falls from a height, altitude is most prominent attribute whose value decreases immediately. GPS could be used to measure the altitude but it is not suitable when location remains the same while the altitude is changing. No dedicated sensor is available that can measure altitude directly; it can be measured by using atmospheric pressure p and temperature. In the conducted experiment, altitude is measured using the pressure p measured by the Sensordrone. Here, we are neglecting the atmospheric temperature because it hardly changes when altitude changes less than 50 feet. Therefore, the final formula would be:

$$altitude = 44330.77 * \left(1 - \left(\frac{p}{p_0}\right)^{0.190263}\right) \tag{2}$$

Where p_0 is the standard reference pressure measured at sea level.

4.3 Ten Millisecond Moving Maximum

Generally, the accelerometer generates signals at a very high frequency, which is more than 2000 Hz. It is very difficult to process and model the data at such a high speed because there can be extreme fluctuations in the readings and such processing becomes resource-intensive. Data pre-processing is required to deal with this situation. Iyoda et al. [2] have used a 10 ms moving average method to pre-process the data generated at 2800 Hz frequency. However, the moving average can downgrade the peak value generated by the accelerometer and cannot identify the fluctuation generated for 1 ms to 2 ms. To take the peak value into account, the system has used 10 ms moving maximum method in which the maximum value of every 10 ms interval has been recorded that is shown in Fig. 8 as …, N-2, N-1, N, N + 1….

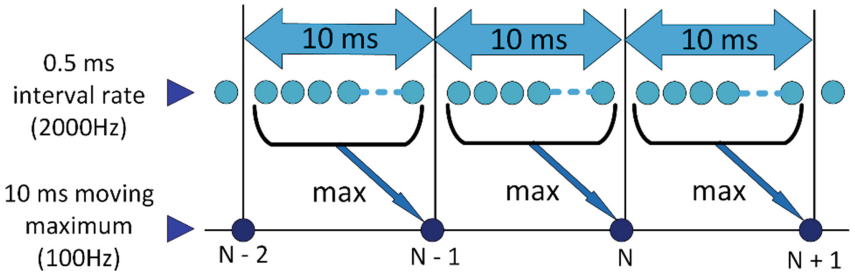


Fig. 8. Ten millisecond moving maximum

5 Statistical Modeling of Fall Detection

This work investigates support vector machines (SVM) to model the event of the fall of a vehicle from an altitude. Proposed model utilizes three inputs viz. vehicle speed, linear acceleration, and altitude to detect the occurrence of a fall event. There are only two possible outcomes of the proposed forecast model, (1) fall occurs, and (2) fall does not occur. Hence, it is deduced as a binary classification problem in which the samples are represented as points in an n-dimensional space and mapped in a way that the samples of the two categories are separated by a gap called hyperplane, which is as wide as possible.

SVM fits an optimal separating hyperplane (OSH) between the two occurrences by utilizing the training samples that are present at the edge of the class distributions - the support vectors. The hyperplane is oriented in a way that it is placed at maximum distance from the sets of support vectors. This orientation helps SVM generalize on unseen cases more accurately in comparison to the classifiers such as the neural networks that perform by minimizing the training error, such as neural networks [18]. Another advantage with SVM classification is that, in contrast with statistical classifiers such as the maximum likelihood classifiers that use all training cases to characterize the classes, only a small number of training samples present at the edge of the class distributions (support vectors) in the feature space are needed to establish the decision surface [19]. Hence, SVM classification requires a much smaller training sample size than the conventional maximum-likelihood classification models. The Naïve Bayes, KNN and random forest classifiers work faster and are more accurate only with a large number of training samples, but in our case the training samples are less. So, SVM is best suited as it requires less training samples and is the most accurate even when the data for training and testing is less [20, 21].

The hyperplane in SVM is defined as

$$W.X + B = 0 \tag{3}$$

Where X is a data-point lying on the hyperplane, W is normal to the hyperplane and B is the bias.

The linearly separable case defines a separate hyperplane for two classes: $W{\cdot}X_i + B \geq +1$ (for $Y_i = +1$) and $W{\cdot}X_i + B \leq -1$ (for $Y_i = -1$). The two equations can be combined as

$$y_i(W \cdot X_i + B) - 1 \geq 0 \tag{4}$$

The training sample points on the two hyperplanes, $W{\cdot}X_i + B = \pm 1$, that are parallel to the OSH, are the support vectors. The gap between the planes is 2/|W| and its maximization is achieved by the constrained optimization problem, under the inequality constraints of Eq. (4).

$$\min\left\{\frac{1}{2}\|W\|^2\right\} \tag{5}$$

To restrict the lower and upper bounds of input, slack variables $\{\xi_i\}$ are introduced, so that the Eq. (4) becomes

$$y(W.X_i + B) > 1 - \xi_i \tag{6}$$

The solutions for which ξ_i are large, are penalized by adding the penalty term $C \sum_{i=1}^{r} \xi_i$. The optimization problem from Eq. (4), under the inequality constraints of Eq. (6) thus becomes,

$$\min\left[\frac{\|W\|^2}{2} + C\sum\nolimits_{i=1}^{r} \xi_i\right] \tag{7}$$

The training data X is mapped into a high-dimensional feature space H through a mapping function ϕ to allow for non-linear decision surfaces. The input data point X is represented as $\phi(X)$ in the high-dimensional space H. As the computation of $(\phi(X)\cdot\phi(X_i))$ is expensive, it is lowered in the high-dimensional space by using a kernel function such that

$$(\phi(X) \cdot \phi(X_i)) = k(X, X_i) \tag{8}$$

Solving the above equations, the decision function is obtained in the form

$$f(x) = \text{sgn}\left(\sum_{i=1}^{r} \alpha_i y_i k(X, X_i) + B\right) \tag{9}$$

Here, αi is the Lagrange's multiplier. The output function f(x) is using the tuning function, SVR model with a cost of 2, 23 support vectors, radial basis kernel, and an epsilon value of 0.2 is used to develop our statistical model.

6 Results and Discussion

The results are obtained with the proposed fall-detection model using the database generated by SNUSense for training and testing. Using SVM tuning, we obtain the optimum SVM classifier is determined with radial basis kernel, gamma = 0.33, $\xi = 0.1$ and 25 support vectors. A total of 215 experimental observations of the fall event have been used for testing and the rest 54 have been used for testing the model. If P is the probability of occurrence, binary 1 represents the occurrence and binary 0 represents the non-occurrence of the fall event. The detection of the single fall event by our system is shown in Fig. 9.

The accuracy of the detection model has been evaluated using mean absolute percentage error (MAPE) [22]. MAPE is defined as

$$MAPE = \frac{1}{N}\sum_{i=0}^{N} \left|\frac{y_i - f(x_i)}{y_i}\right| * 100 \tag{10}$$

Where, N is the number of observations, y_i is the actual event that occurred at i^{th} observation (represented by 0 or 1), x_i is the input vector (time), and f is the forecast model.

The MAPE for our SNU database was found to be 1.80%, i.e. the model is able to accurately detect 53 out of 54 test occurrences. The detection of the fall event by the IoT system is depicted in Fig. 10.

Hence, it is concluded that our fall detection system has an accuracy of 98.2% and is capable of accurately detecting events of vehicle fall.

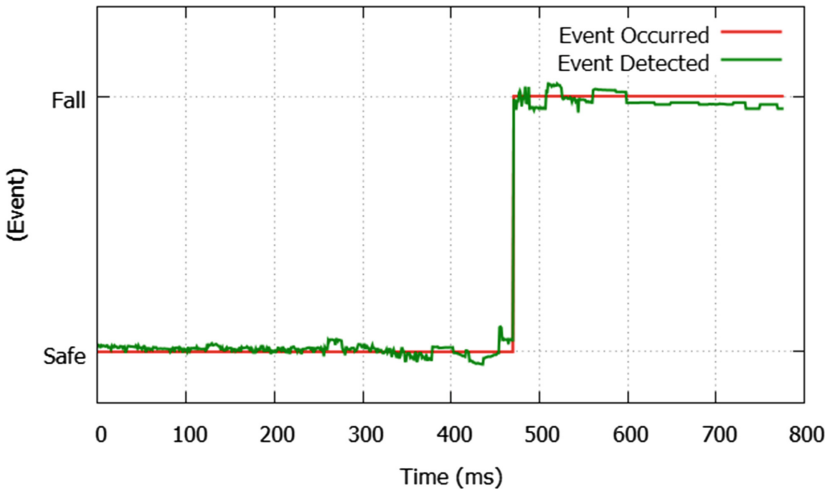


Fig. 9. Detection of a single fall event by proposed IoT system

7 Conclusion and Future Work

In this work, a reliable, affordable and precise IoT system has been presented, which can be retrofitted to any vehicle. It uses a smartphone and connected sensors to sense absolute linear acceleration, speed, and altitude of the vehicle. These parameters are used to develop an SVM based fall detection model, which has been proven to accurately detect events of vehicle fall events in this work.

As a future work, we propose to develop and evaluate an IoT accident reporting system which is capable of detecting as well as classifying the types of accident event.

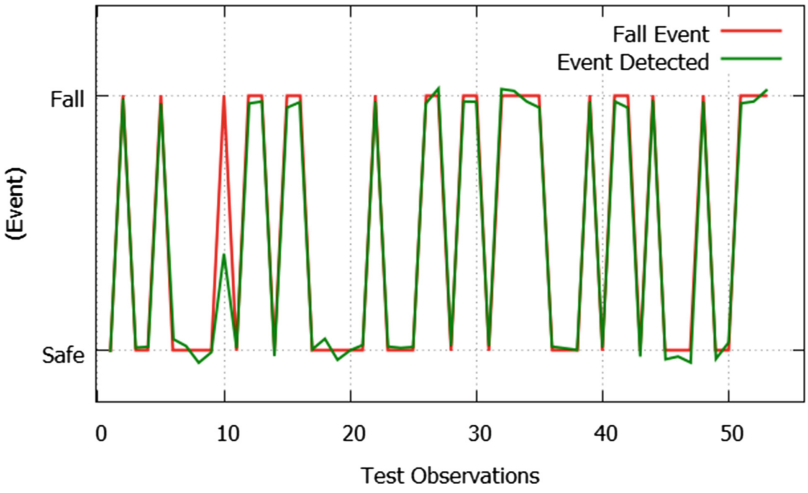


Fig. 10. Results of the event detection system for 54 test observations

The proposed system will be able to report the occurrence as well as the severity of the accident to the end users. For this purpose, we plan to use parameters such as the vehicle speed, acceleration, roll, pitch and altitude to categorize the road accident as mild, moderate or severe.

References

1. Golbal Status Report on Road Safety, World Health Organisation (WHO) (2018)
2. Iyoda, M., Trisdale, T., Sherony, R., Mikat, D., et al.: Event data recorder (EDR) developed by Toyota motor corporation. SAE Int. J. Trans. Safety **4**(1), 187–201 (2016). https://doi.org/10.4271/2016-01-1495
3. Consultancy Services for the Provision of a Road Accident Data Management System (RADMS) - Variation Order No. 4 – Operational Support and Training Completion Report - Quarter 3 from April 2018 to June 2018
4. Ferreira, J., Carvalho, E., Ferreira, B.V., de Souza, C., Suhara, Y., et al.: Driver behavior profiling: an investigation with different smartphone sensors and machine learning. PLoS One **12**(4), e0174959 (2017)
5. Aloul, F., Zualkernan, I., Abu-Salma, R., Al-Ali, H., Al-Merri, M.: iBump: smartphone application to detect car accidents. Comput. Electric. Eng. **43**, 66–75 (2015). ISSN 0045-7906
6. Liyanage, Y., Zois, D.-S., Chelmis, C.: Quickest freeway accident detection under unknown post-accident conditions. In: 6th IEEE Global Conference on Signal and Information Processing (GlobalSIP), Anaheim, CA, 26–29 November 2018 (2018)
7. Dar, B.K., Shah, M.A., Islam, S.U., Maple, C., Mussadiq, S., Khan, S.: Delay-aware accident detection and response system using fog computing. IEEE Access **7**, 70975–70985 (2019). https://doi.org/10.1109/ACCESS.2019.2910862
8. Xu, X., Yu, J., Chen, Y., Zhu, Y., Qian, S., Li, M.: Leveraging audio signals for early recognition of inattentive driving with smartphones. IEEE Trans. Mob. Comput. **17**(7), 1553–1567 (2018). https://doi.org/10.1109/tmc.2017.2772253
9. Park, H., Ahn, D., Park, T., Shin, K.G.: Automatic identification of driver's smartphone exploiting common vehicle-riding actions. IEEE Trans. Mob. Comput. **17**(2), 265–278 (2018). https://doi.org/10.1109/tmc.2017.2724033
10. Bhatti, F., et al.: A novel Internet of Things-enabled accident detection and reporting system for smart city environments. Sensors **19**(9), 2071 (2019). https://doi.org/10.3390/s19092071
11. Lohani, D., Acharya, D.: Real time in-vehicle air quality monitoring using mobile sensing. In: 2016 IEEE Annual India Conference (INDICON), Bangalore, pp. 1–6 (2016). https://doi.org/10.1109/indicon.2016.7839099
12. https://github.com/Sensorcon/Sensordrone
13. https://g-p.hk/gptoys-foxx-s911.html
14. https://firebase.google.com
15. Lee, Y., Yeh, H., Kim, K.-H., Choi, O.: A real-time fall detection system based on the acceleration sensor of smartphone. Int. J. Eng. Bus. Manage. (2018). https://doi.org/10.1177/1847979017750669
16. Kendall, J., Solomon, K.A.: Air bag deployment criteria, The Forensic Examiner (2014)
17. http://www.gpsinformation.org/dale/nmea.htm
18. Utkin, L.V., Chekh, A.I., Zhuk, Y.A.: Binary classification SVM-based algorithms with interval-valued training data using triangular and Epanechnikov kernels. Neural Netw. **80**, 53–66 (2016). ISSN 0893-6080
19. Mathur, A., Foody, G.M.: Multiclass and binary SVM classification: implications for training and classification users. IEEE Geosci. Remote Sens. Lett. **5**(2), 241–245 (2008)

20. Hmeidi, I., Hawashin, B., El-Qawasmeh, E.: Performance of KNN and SVM classifiers on full word Arabic articles. Adv. Eng. Inform. **22**(1), 106–111 (2008)
21. Madzarov, G., Gjorgjevikj, D., Chorbev, I.: A multi-class SVM classifier utilizing binary decision tree. Informatica **33**, 233–241 (2009)
22. de Myttenaere, A., Golden, B., Le Grand, B., Rossi, F.: Mean absolute percentage error for regression models. Neurocomputing **192**, 38–48 (2016). https://doi.org/10.1016/j.neucom.2015.12.114. ISSN 0925-2312