Forward collision warning system for motorcyclist using smart phone sensors based on time-to-collision and trajectory prediction

Qun Lim and Yi Lim
Department of Engineering Product Development, Singapore University of Technology and Design, Singapore, Singapore
Hafiz Muhammad
Department of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore, Singapore, and
Dylan Wei Ming Tan and U-Xuan Tan
Department of Engineering Product Development, Singapore University of Technology and Design, Singapore, Singapore

Abstract

Purpose – The purpose of this paper is to develop a proof-of-concept (POC) Forward Collision Warning (FWC) system for the motorcyclist, which determines a potential clash based on time-to-collision and trajectory of both the detected and ego vehicle (motorcycle).

Design/methodology/approach – This comes in three approaches. First, time-to-collision value is to be calculated based on low-cost camera video input. Second, the trajectory of the detected vehicle is predicted based on video data in the 2 D pixel coordinate. Third, the trajectory of the ego vehicle is predicted via the lean direction of the motorcycle from a low-cost inertial measurement unit sensor.

Findings – This encompasses a comprehensive Advanced FWC system which is an amalgamation of the three approaches mentioned above. First, to predict time-to-collision, nested Kalman filter and vehicle detection is used to convert image pixel matrix to relative distance, velocity and time-to-collision data. Next, for trajectory prediction of detected vehicles, a few algorithms were compared, and it was found that long short-term memory performs the best on the data set. The last finding is that to determine the leaning direction of the ego vehicle, it is better to use lean angle measurement compared to riding pattern classification.

Originality/value – The value of this paper is that it provides a POC FWC system that considers time-to-collision and trajectory of both detected and ego vehicle (motorcycle).

Keywords Forward collision warning, Time-to-collision, Advanced driving assistance system, Motorcycles

Paper type Research paper

1. Introduction

As technology advances, there is an increase in vehicle safety systems such as autonomous driving and Advanced Driving Assistance System (ADAS). Still, these did not lower the number of accident deaths. Unfortunately, Road Traffic Accidents (RTA) still increase from 1.25 to 1.35 million casualties globally between 2015 and 2018, according to the World Health Organization (WHO). This discrepancy is because the people who need this safety feature do not have the purchasing power for such technologies. This is evident by the fact that 93% of road fatalities come from the less developed nations. ADAS is expensive because it requires the use of many expensive sensors and to increase adoption, it is essential to create a low cost ADAS.

Unfortunately, Vulnerable Road Users (VRUs) such as cyclist and motorcyclist, which accounts for more than 50% of fatalities, do not have ready access to such sensors on their vehicles as ADAS are mainly made for cars. Hence, the next best alternative is to create an ADAS, which can tap onto smartphone hardware for VRUs. This paper will focus on the advanced Forward Collision Warning (FCW) system using only a monocular camera and an Inertial Measurement Unit (IMU), which both can be found on most smartphones.

The current issue and full text archive of this journal is available on Emerald Insight at: **https://www.emerald.com/insight/2399-9802.htm**

Journal of Intelligent and Connected Vehicles
4/3 (2021) 93–103
Emerald Publishing Limited [ISSN 2399-9802]
[DOI 10.1108/JICV-11-2020-0014]

Received 13 November 2020
Revised 29 June 2021
Accepted 24 August 2021

This paper intends to develop an Advanced Forward Collision Warning system that is capable of predicting if a collision will occur based on time-to-collision and considering the trajectory of the detected vehicle and the leaning direction of the ego vehicle. To develop this, the paper must tackle three challenges which are tackled in Sections 2, 3 and 4 accordingly.

The first challenge is the difficulty of obtaining time-to-collision for the FCW system based on using a single camera module as a camera. This is because cameras are designed primarily to capture images and not measure distance. This challenge will be tackled in Section 2. The second challenge is to predict the trajectory of the detected vehicle despite the ever-changing dynamics of the traffic situation. This challenge will be tackled in Section 3. The last challenge is to predict the leaning direction of the ego vehicle using an IMU sensor which is subjected to the high vibration frequency of the motorcycle, which can go up to 1Ghz. This challenge will be tackled in Section 4.

1.1 Scope of work

The scope of work is the proof of concept of the proposed advanced FCW (ADAS) system based on a single camera and low-cost IMU for motorcycles. The following aspects are presented to design and develop ADAS using low-cost sensors for motorcycles:

- Time-to-collision algorithm based on low-cost camera input, lightweight vehicle detection and nested Kalman filter.
- Trajectory prediction of vehicle coordinates algorithm to predict the future positions of the detected vehicle using the Long Short-Term Memory (LSTM) method.
- Low-cost IMU to determine motorcycle direction of lean.

1.2 Limitation

This section will present the assumptions and limitations of using the low-cost camera and IMU.

For calculation of relative distance and speed for time-to-collision, an ideal camera lens must be used such that the image follows the lens equation. However, in real life, the image captured has distortion and does not completely follow the lens equation, which will result in a deviation between the actual distance and distance calculated. Furthermore, time-to-collision is only calculated from the frontal axis, and potential collision from the lateral axis is not considered. Next, the accuracy of the distance detected also depends on the vehicle detection algorithm, and it runs on an assumed width of a vehicle which defers between vehicles.

For trajectory prediction, it is a challenge to predict a trend that is independent of past history. Hence, it is challenging to detect if a vehicle were to suddenly change direction in a completely abnormal way, such as very sudden swerving or braking.

Lastly, measuring lean direction for motorcyclists is subjected to the vibrations that a motorcycle produces.

1.3 Literature review

FCW encompasses many different sensors. These sensors help drivers to drive better through visual feedback, control and warnings to drive safer. This paper will focus on FCW, which uses a camera and IMU sensor only. In the paper by Zhao *et al.* (2014), a look-up-table (LUT) is used based on the shadow length of the vehicle in the image to deduce the distance. This means that the distance is calculated beforehand, and the LUT is not easily transferable as it is based on the position of the camera on the car. In the papers by Lim *et al.* (2018) and Salari and Ouyang (2013), it uses the width of the vehicle to obtain the distance and the time-to-collision data. However, the paper by Salari and Ouyang (2013), does not show how it filters the data as using this method is very noisy, and the article by Lim *et al.* (2018) shows a nested Kalman filter for displacement and velocity individually, but both of them can be combined into a single Kalman filter which requires less computational power. Similar to the paper by Lim *et al.* (2018), this paper will use computer vision and obtain distance based on width, but it uses the nested Kalman filter in a different way where it first uses a Kalman filter on relative distance and speed and the subsequent filter on time-to-collision based on a constant velocity model.

Next, another important feature is trajectory prediction of the vehicles in the vicinity. This is because if the system knows the trajectories of the vehicles based on their path history, it can extrapolate and determine if there will be a potential collision from these trajectories. While trajectory prediction is a heavily researched field in the realms of autonomous cars, it mainly uses expensive sensors like lidar for forecasting. As such, this paper will focus on using such algorithms using only camera and IMU data for prediction. There are many papers that use machine learning and deep neural network algorithms which are used as a reference for this paper's trajectory prediction. They are articles by Lim *et al.* (2019), Goli *et al.* (2018), Heravi and Khanmohammadi (2011), Park *et al.* (2018).

Apart from the neural network and machine learning, the Kalman filter is excellent at forecasting as it uses a state model for estimation. In the papers by Schulz *et al.* (2018), Lim *et al.* (2018), Kalman filter is used for forecasting as it can forecast 1-step ahead by comparing between the kinematic model and measurements value. Kalman filter can also predict h-steps ahead without the measurement update using its state transition matrix. Moving average is also another forecasting technique that is proven useful, as shown in the paper by Lauren and Harlili (2014), Alghamdi *et al.* (2019), Radziukynas and Klementavicius (2014), Sulandari and Yudhanto (2015). Moving average, Kalman filter, and LSTM will be compared further in Section 3.

Lastly, it is important to obtain information about the ego vehicle for a better warning system. There are various ways to quantify the ego vehicle direction, which are roll angle and riding pattern classification. First, roll angle measurements will determine if the ego vehicle is leaning toward the left or right. Next, riding pattern classification can be used to determine if a motorcyclist is changing lanes, making a turn or moving straight depending on the labels of the training set. IMU can be used to measure roll angle and riding pattern classification. To get accurate roll angle measurement, sensor fusion must be applied as IMU sensors are known to be subjected to noise and drift. In the paper by Hossam-E-Haider *et al.* (2017), sensor

fusion is applied to IMU sensors, and it shows and compares complementary and Kalman filter, which is important in understanding which filter is more applicable to this paper. Next, to perform riding pattern detection, it involves time series classification. ADAS uses many methods for time series classification, from lane change detection to dangerous driving classification. For example, in the paper by lane changing maneuvers are classified with LSTM and GRU, respectively. Further modifications can be done to the architecture, such as using 1 D CNN, CNN-LSTM, LSTM encoder decoder such as which can improve the results. This paper will use and compare some of these algorithms.

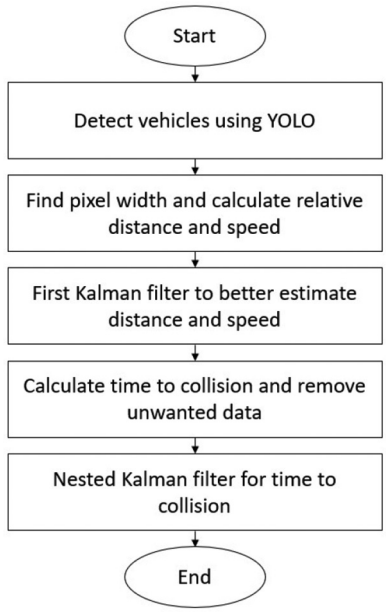
The next section will explain the basics of a FCW System and the reason why it is important to have a nested Kalman filter for prediction.

2. Nested Kalman filter based forward collision warning

A FCW system is a safety feature that warns the user that there is a frontal collision. It not only requires the ego vehicle to be able to detect frontal vehicles but also to quantify it through relative distance, velocity, acceleration and time-to-collision. This is a challenge because a monocular camera does not have the luxury of using sensor fusion to get a better estimation of distance. Hence the camera sensor will be used as a single sensor used to measure multiple measurements such as relative distance and speed. To measure distance and speed, this section will explain how a vehicle detection algorithm can be used to measure the distance from bounding box width and derive the relative distance, speed between the detected and ego vehicle. The output of the vehicle detection algorithm will be used as inputs for the first Kalman filter. Next, using the output of the first Kalman filter, the time-to-collision is calculated. This time-to-collision data is further filtered as zero, and negative speeds will result in very noisy data. After the data filtering, a second Kalman filter is used to remove the noise and unwanted data, hence the nested Kalman filter.

The algorithm will be explained in the following steps as seen in Figure 1 – (2.1) relative distance, speed estimation based on boundary box width, (2.2) Kalman filter to stabilized estimated relative distance and speed, (2.3) time-to-collision calculation and lastly, (2.4) nested Kalman filter for time-to-collision. The vehicle detection algorithm that is used will be using the You-Only-Look-Once detection algorithm because it has proven to be state of the art with extremely fast processing speed of up to 244 frames per second (fps) with reasonable accuracy of 78.6 mean average precision (maP) by Redmon *et al.* (2016). The output of the YOLO boundary box coordinates to obtain the boundary box width and height will be used for distance estimation, and the bounding box center coordinates will be used for trajectory prediction. For this paper, the assumption is that the difference between the bounding box left and right limit will correspond to the pixel width of the car. This means that vehicles that are not directly in front of the ego vehicle will have a larger error for distance estimated as the boundary box width is the variable that calculates the distance. For the nested Kalman filter FCW system. Both YOLO and Support Vector Machines (SVM) were used for this section, and YOLO was chosen because it was able to perform faster with 20 FPS.

Figure 1 Nested Kalman filter steps



2.1 Relative distance, speed estimation based on boundary box width

The distance between the ego vehicle and the detected vehicles can be calculated directly from an image with a known vehicle width or height. However, every car model will have a slightly different width and height and to get an accurate estimated distance which means that the algorithm must be able to detect different models of cars. To reduce complexity, the vehicle width of cars is assumed to be 1.8 meters, and width estimation is used for distance estimation as the variance of car width is smaller than that of car heights. For example, a sports car's height is low while a Sport Utility Vehicle (SUV) car height is much higher. The width of both sports car and SUV are relatively similar. The formula to calculate distance from vehicle width or height is as follows:

$$D_t = \frac{fW}{w_t} \tag{1}$$

where W is the physical width of the car in meters, w_t is the pixel width of the vehicle at time t, f is the focal length of the camera lens in meters. D_t is the relative distance from the ego vehicle to the detected vehicle in meters. From the distance in equation (1), the relative speed can be derived easily with Δt. While these formulas seem relatively easy to use, it is also under the assumption that the camera lens is an ideal lens that does not have spherical aberration, and a method to reduce this uncertainty will be to undistort the image for camera calibration. This can be addressed by using the OpenCV library function cv2.undistort() to calibrate the camera with respect to a chessboard.

2.2 Kalman filter to stabilized estimated relative distance and speed

After estimating the distance from the boundary box width, the data must be stabilized due to the noise of the boundary box width data. It is important to filter and smooth this data because this noise and error generated will be brought forward in the calculation of time-to-collision. In this section, the Kalman filter is used to stabilize the estimated relative distance and speed. The Kalman filter algorithm is as follow (where x and $\dot{x}$ represents the relative distance and speed of the vehicle detected):

Firstly, initialize the Kalman filter with estimates of $\hat{\mathbf{x}}_0$ and $\mathbf{P_0}$.

$$\text{Where } \hat{\mathbf{x}}_0 = \begin{pmatrix} x_{t=0} \\ \dot{x}_{t=0} \end{pmatrix}, \ \mathbf{P}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Next, the Kalman filter will project the state and error covariance ahead in equations 2 and 3. Equation (2) is a state-space model where the priori estimate is based on the state transition matrix $\mathbf{F_{t-1}}$.

$$\hat{\mathbf{x}}_t^- = \mathbf{F}_{t-1}\hat{\mathbf{x}}_{t-1}^+ + \mathbf{B}_{t-1}\mathbf{u}_{t-1} \tag{2}$$

$$\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}^+\mathbf{F}_{t-1}^{\mathrm{T}} + \mathbf{Q}_{t-1} \tag{3}$$

where $\mathbf{F_{t-1}} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}$ and $\mathbf{u} = 0\mathrm{u} = 0$, as there is no control input based on the constant velocity model. Next, the Kalman gain is computed in equation (4) for the measurement update step in equation (5).

$$\mathrm{Kt} = \mathbf{P}_t^-\mathbf{H}_t^T \left(\mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t\right)^{-1} \tag{4}$$

$$\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_t^-) \tag{5}$$

$$\mathbf{z}_t = \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} x_t \\ x_t \end{pmatrix} \tag{6}$$

$\mathbf{z}_t$ refers to the measurement of the relative distance and speed detected. The last step of the Kalman filter is to update the error covariance in equation (7).

$$\mathbf{P}_t^+ = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t^- \tag{7}$$

Equation (7) is the last step and the Kalman filter repeats from equations (2) to (7) continuously to estimate future values of relative distance and speed of up to $t + 1$. Moreover, the Kalman filter is computationally efficient because it only needs to remember the last step, allowing quick computation.

2.3 Time-to-collision calculation

Time-to-collision (TTC) can be derived as follows:

$$TTC = \frac{D_t}{v_t} \tag{8}$$

Time-to-collision will be used as a measure of how much of a threat is a detected vehicle to the ego vehicle. The threshold of the danger warning is set to 4 s. The danger warning can be improved using simplified notification through sound alerts or blinking external light sources.

2.4 Nested Kalman filter for time-to-collision

After time-to-collision is calculated, the data must be further manipulated. This is because the time-to-collision data is heavily dependent on D_t and v_t. When v_t is negative, the time-to-collision is negative, which has to be removed. When v_t is zero, the time-to-collision will be infinity which will result in a spike. Next, time-to-collision data should only be used within a range of interest. There is not much value in recording time-to-collision data with the conditions mentioned above. Hence, when velocity is zero, negative or if the time-to-collision data is above a certain threshold, it will return the threshold value. This will be further demonstrated in the simulation section.

In the next section, trajectory prediction of vehicle pixel center coordinates using LSTM neural network will be compared with existing time series forecasting methods.

3. Trajectory prediction model

Trajectory prediction is very important in the advanced FCW system because by understanding the trajectory of the detected vehicle, the FCW algorithm will have more time to alert the ego vehicle if there will be a potential collision. The trajectory prediction input data is based on pixel center coordinates of the detected vehicle using the object detection algorithm in the previous section to obtain time-series data denoted by:

$$\mathbf{y} = (y_1, y_2, \ldots, y_t)$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_t)$$

With the inputs above, the objective is to map x_t, y_t, to x_{t+h}, y_{t+h} coordinates which are h steps ahead denoted by:

$$\mathbf{y_{prediction}} = (y_1 + h, y_{2+h}, \ldots, y_{t+h})$$

$$\mathbf{x_{prediction}} = (x_{1+h}, x_{2+h}, \ldots, x_{t+h})$$

To quantify the effectiveness between algorithms for trajectory prediction, it has to be compared with basic kinematics forecasting such as a constant velocity model. In this section, the comparison will be with simple moving average constant velocity model, Kalman filter forecasting, and LSTM model using root mean square error to quantify the performance.

3.1 Moving average forecasting

Moving average is defined as a succession of averages derived from successive segments of a series of values. It is simple to use, but it comes with an increased lag for increased smoothing based on the number of data points used for the moving average. Moving average (MA) can be computed with the following equation:

$$MA_n = \frac{\sum_{i=1}^{n} d_i}{n} \tag{9}$$

where d_i refers to the data points and n is the number of points for the moving average. Moving average for forecasting can be

done by using the predicted value based on past values as the actual value for the next prediction. However, this method converges quickly to the mean of the last few data points. Hence, instead of this forecasting method, a moving average will be used on the data, and the difference between x_t and x_{t-1} will be multiplied by h to forecast h steps ahead.

3.2 Kalman filter forecasting

Instead of using a simple moving average filter, a Kalman filter forecasting is used for comparison. In Section 2, the Kalman filter is used to estimate distance and speed based on the pixel width of the bounding box, which predicts a 1 step ahead estimation. For trajectory prediction, it is required to predict h steps estimation instead, and it uses pixel coordinates and pixel velocities as its input to the Kalman filter. To predict h steps ahead, the state-space model is recursively used instead of the measurement equation.

3.3 Long short-term memory

The last part of this section will introduce deep learning for trajectory prediction based on the LSTM model. LSTM models have proven themselves useful and effective in many time series applications. However, LSTM on vehicle pixel center coordinates has yet to be explored, and this section will explain further the LSTM model and the pre and post-processing required. The key principle of an LSTM is its cell state and the various gates: the input gate, output gate and forget gate. The following explains the operations of an LSTM cell:

$$\mathbf{i_t} = \sigma(W_i[\mathbf{h_{t-1}}, x_t] + \mathbf{b_i}) \tag{10}$$

$$\mathbf{ft} = \sigma\left(W_f[\mathbf{h_{t-1}}, x_t] + \mathbf{b_f}\right) \tag{11}$$

$$\mathbf{o_t} = \sigma(W_o[\mathbf{h_{t-1}}, x_t] + \mathbf{b_o}) \tag{12}$$

$$\mathbf{c_t} = \mathbf{f_t} * \mathbf{c_{t-1}} + \mathbf{i_t} * tanh(W_c[\mathbf{h_{t-1}}, x_t] + \mathbf{b_c}) \tag{13}$$

$$\mathbf{h_t} = \mathbf{o_t} * tanh(\mathbf{c_t}) \tag{14}$$

where

$\mathbf{i_t}$: represents input gate;
$\mathbf{f_t}$: represents forget gate;
$\mathbf{o_t}$: represents output gate;
σ : represents sigmoid function;
w_x : weight for respective gate(x) neurons;
$\mathbf{h_{t-1}}$: output of the previous LSTM block;
x_t : input of current timestamp; and
$\mathbf{c_t}$: cell state(memory).

The gates 10,11,12 determine how the LSTM forget and updates its information and the cell state and output are updated according to equations (13) and (14). To use LSTM for time series forecasting, it is important to rescale the data as LSTM is very sensitive to the scale of input data. Hence, pre- and post-processing are required. First, the input data has to be rescaled to the range of 0 to 1. Next, the data will be used as input for the LSTM, and the output of the LSTM will then be rescaled back accordingly. Tensorflow Abadi *et al.* (2016) is used for the LSTM time series forecasting training and testing.

This section has covered the various algorithms for trajectory prediction of vehicle center coordinates, and the experimental results will be shown in Section 5. In the next section, the IMU sensor will be used to augment the FCW system by providing information on the lean direction of the ego vehicle.

4. Leaning direction of ego vehicle

This section will explore two different approaches to find the leaning direction. The first approach involves measuring the leaning angle of the motorcycle with respect to the ground. The second approach involves using machine learning and neural network to perform riding pattern classification, and both methods will be compared in terms of how it can better augment the FCW by providing information of the ego vehicle.

4.1 Leaning angle of ego vehicle

The first method to detect the leaning angle of the ego vehicle and the assumption is that a lean to the right will correspond to the rightwards trajectory of the ego vehicle, and a lean to the left will correspond to the leftwards trajectory accordingly. To detect the leaning angle, a gyroscope and an accelerometer are required, which can be found in an IMU. The gyroscope is used to detect a change in lean angle, and the accelerometer is used to measure the gravity force, which will then be used to calculate for lean angle. These two sensors are combined with the use of a complementary filter or Kalman filter for leaning angle detection. To calculate the lean angle, the accelerometer sensor is used to detect the gravity, and when the ego vehicle is standing upright, the accelerometer will detect an acceleration of 1 g at $\varphi = 0°$. When φ is 90°, a will be 0 g. The equation for phi, φ, detected by the accelerometer is as follows:

$$\varphi = 90^{\circ} - arccos\left(\frac{a}{g}\right) \tag{15}$$

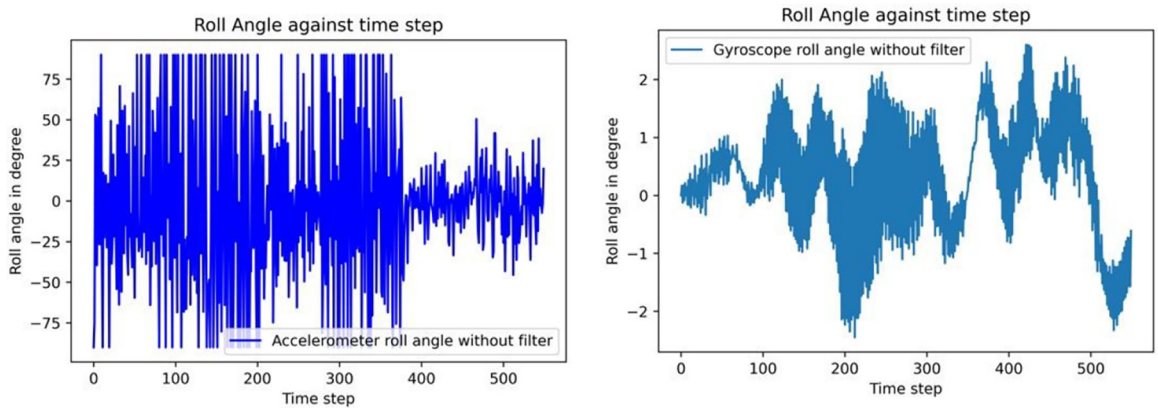
While both sensors can detect the lean angle individually, each sensor has its pros and cons, and a sensor fusion between these sensors will produce a more reliable result. An accelerometer sensor measures all forces acting on it, which includes vibration, centripetal and coriolis acceleration which makes it only reliable in static lean angle detection and better in the long term. On the other hand, the gyroscope is not susceptible to external force detected, and it is able to measure the angular rate of change precisely, but the integration over time of angle rate will result in drift with the angle not returning to zero. Hence, the idea of using sensor fusion is to use each sensor to make up for the other sensor's weaknesses, and this can be done with a complementary filter or a Kalman filter.

As seen in Figure 2, both accelerometer and gyroscope lean angle measurement are full of noise as compared to Figure 6 and hence not reliable. This subsection will explain how both complementary filter and Kalman filter is necessary for lean angle detection, and it will evaluate which filter is better.

4.1.1 Complementary filter sensor fusion

In order for both sensors to complement each other, the data from the accelerometer sensor will be passed through a low pass filter while the gyroscope data will be pass through a high pass filter, and the complementary filter can be described as follows:

Figure 2 Lean angle using accelerometer and gyroscope individually



$$angle = \alpha * (angle + gyrData * dt) + (1 - \alpha) * accData \quad (16)$$

From the equation above, it is observed that the parameters affecting the angle is only the α value, and it does not have any information on the output filtered data.

4.1.2 Kalman filter sensor fusion

On the other hand, in Kalman filter, the output filtered data is used as an input for the next iteration as it is an iterative process which aims to find the statistically optical value. Unlike the Kalman filter in Section 2, lean angle and angular velocity are used as state vector instead of relative distance and speed. This subsection has explained how the lean angle is measured using Kalman filter and complementary filter. The next section will use machine learning and neural networks in classifying riding patterns to predict the leaning direction of the ego vehicle.

4.2 Riding pattern classification

The objective of the riding pattern classification is to detect riding maneuvers such as moving straight, turning and lane change to predict the trajectory of the ego vehicle.

Similar to the time series prediction in Section 3, riding pattern classification involves the input of a time series array, but the output will be a single class. For this paper, the classes of interest will be as follows: Moving straight (S); Left Turn (LT); Right Turn (RT); Left lane change (LLC); Right lane change (RLC); and Others (O). Similar to the time series prediction in section 3.3, the algorithm used for time series classification will be the LSTM. Instead of training the network on regression outputs of vehicle center coordinates in Section 3, the training outputs will be the six classes. In the next section, simulation and results will be shown based on collected data which will be used to test and evaluate the respective algorithms.

5. Simulation and results

For this paper, the data was collected from a smartphone (Huawei Mate 20 Pro) which is mounted vertically onto a Honda TA-200 motorcycle to collect IMU and video data, which are in sync. The IMU sensor is the model lsm6dsm which consists of both an accelerometer and gyroscope. For the video recording, the video is recorded at 29.97 frames per second, and the IMU data are recorded at 10 hz and 50 hz. For nested Kalman filter and trajectory prediction, the video data recorded will be used for testing and evaluation. For leaning direction prediction, both IMU data of 10 hz and 50 hz are used. The simulation results are based on an 11 second 330 frames video, and the respectively IMU data is recorded accordingly.

5.1 Nested Kalman filter simulation and results

For this subsection, the objective is to quantify stabilization, and root mean square error is used for comparison to the ground truth. While the video is recorded without the use of an external sensor to detect parameters such as distance, speed and time-to-collision, the ground truth is to be estimated. The ground truth is estimated by using the scipy.signal.filtfilt in python, and this is done as this filter is able to filter forward and backward with no time shift, and it is post-processing, so it cannot be done in real-time to replace Kalman filter.

5.1.1 Results on relative distance and speed

In Figure 3, it shows how the Kalman filter can reduce noise. From Table 1, it shows that the reduction in noise for relative distance seems minimal from 0.49 to 0.34.

Still, there is a large noise reduction for the relative velocity from 18.2 to 2.93, and this is important especially for the calculation of time-to-collision, which depends on both values.

5.1.2 Results on time-to-collision

Figure 4 (left) shows the time-to-collision without using any filter for relative speed and velocity. Hence, the first Kalman filter is very important, especially for the relative speed velocity as the root mean square error is reduced by a factor of 6.2. Figure 4 (Right) shows the time-to-collision graph after removing negative velocities, zero velocity, and time-to-collision above the threshold of 8 s as any value above 8 s does not contribute much to the warning system and only makes the data noisier. By comparing both figures, the first Kalman filter is necessary to make this time-to-collision warning useful. As seen in Table 2, the RMSE is reduced from 4.05 to 1.23 to 0.72 from unfiltered to using the first Kalman filter for relative distance and speed and lastly with the nested Kalman filter.

Figure 3 Relative distance and speed with Kalman filter

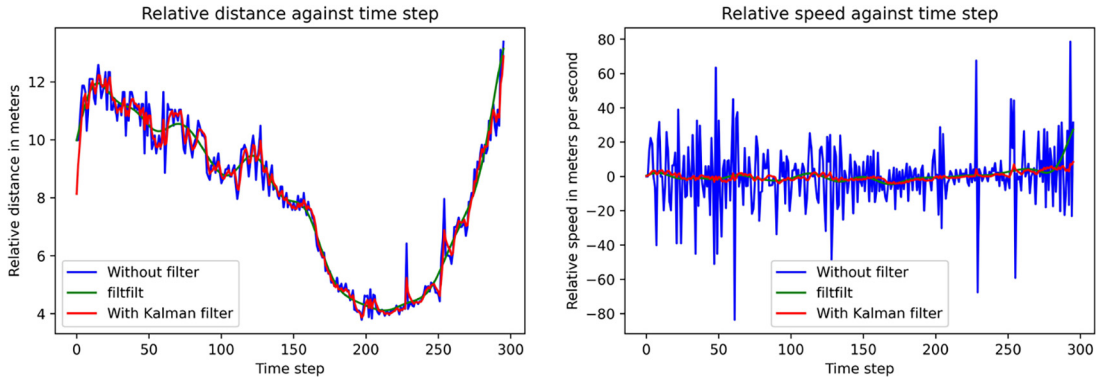


Table 1 RMSE of relative distance and speed

Root mean square error	Relative distance	Relative speed
Without filtered	0.49	18.3
Kalman filter	0.34	2.92

For time-to-collision analysis, false positives will also be used to quantify the improvement in time-to-collision. From above, it is observed that most instances of time-to-collision without Kalman filter create false positive. While this experiment does not include other sensors to detect actual distance and relative velocity, a true FCW will be assumed to be the instance whereby both time-to-collision (with nested Kalman filter) and time-to-collision (with first Kalman filter) agrees that the value is equal or less than 4 s. Through this method, the total false positive has been reduced from 26 to 2.

5.2 Trajectory prediction experimental results

The experiment is conducted using the same video as Section 2. The (x,y) center vehicle coordinates from YOLO are compiled into a list. This data will be used for the trajectory prediction experiment. This subsection will quantify trajectory prediction by comparing the root mean squared error (MSE) for moving average forecasting, Kalman filter forecasting and LSTM neural network. For all of the algorithms mentioned above, the experiment aims to predict $h = 20$ steps ahead.

Table 2 RMSE of time-to-collision

Root mean square error	Time-to-collision
Without filter	4.05
With first Kalman filter	1.23
Nested Kalman filter	0.72

From Table 3, it is evident that LSTM forecasting is the best as it is able to achieve the lowest root mean square error value. This is expected as LSTM applications have been on the rise due to their proven performances. Moreover, the deep neural network has been becoming more popular as it does not require feature extraction as required in traditional machine learning algorithms. The performance comparison between the algorithmns can be seen in Fig 5.

5.3 Leaning direction prediction simulation and results

5.3.1 Results on leaning angle detection

For leaning angle detection, the 50 hz IMU data was used. For the complementary filter, the accelerometer and gyroscope data

Table 3 RMSE of trajectory prediction algorithm

RMSE	x pixel coordinates	y pixel coordinates
Moving average forecasting	357	174.4
Kalman filter forecasting	857.7	42.6
LSTM forecasting	120.5	15.8

Figure 4 Time-to-collision without using any filter (Left) and with nested Kalman filter (Right)

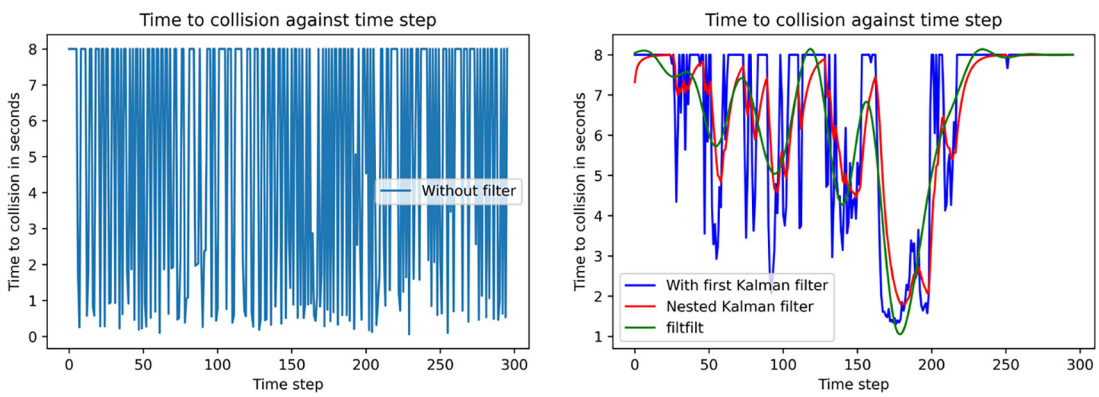


Figure 5 Trajectory prediction summary

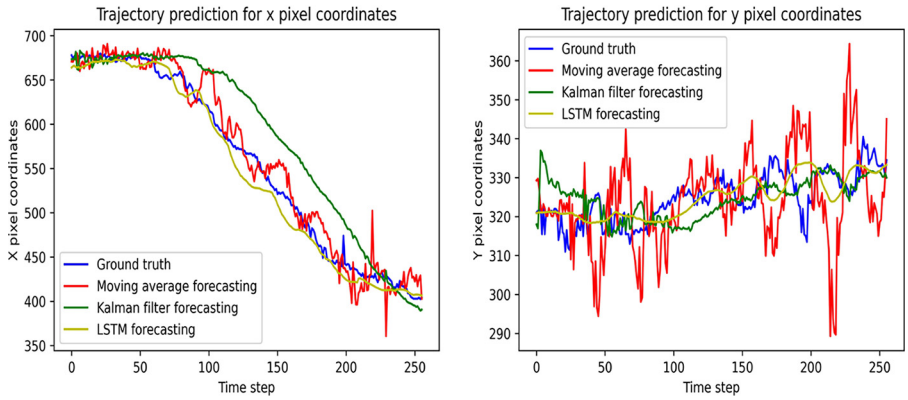
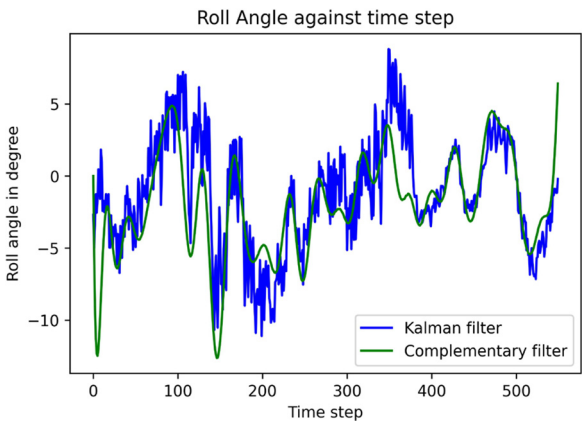


Figure 6 Lean angle using complementary and Kalman filter

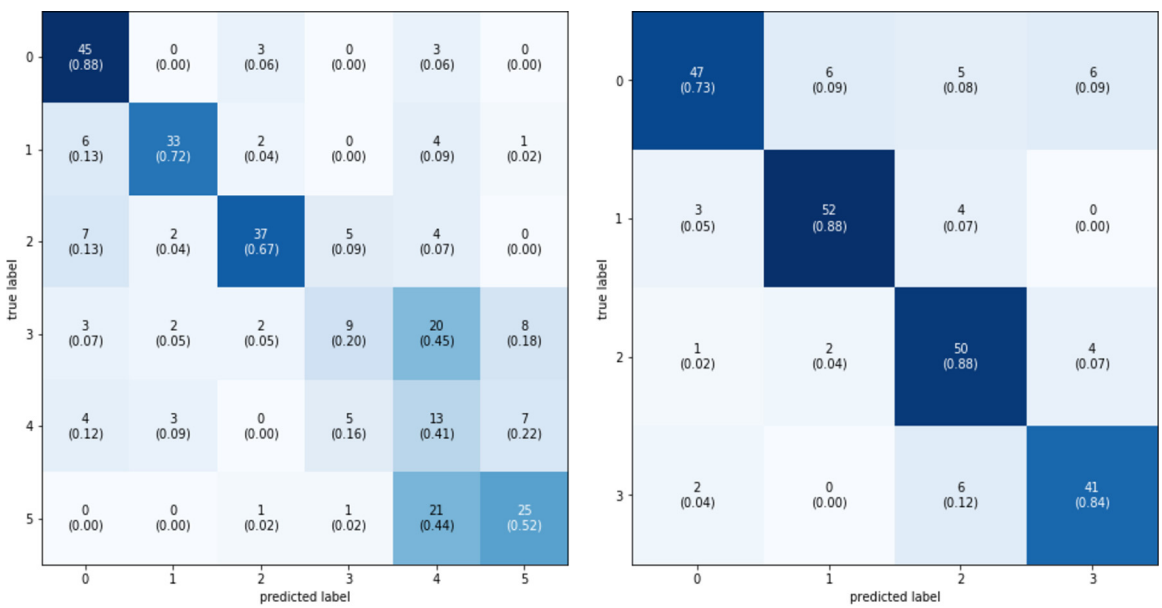


are first passed through a low and high pass filter at 3 hz cutoff frequency, respectively. Next, the $\alpha = 0.1$ is used. For Kalman filter, the noise for the accelerometer and gyroscope are first measured and the $Q = 0.06$ is used.

As seen in Figure 6, both filters are able to filter out most of the noise decently. While the complimentary filter seems smoother than the Kalman filter in Figure 6 above, it, however, has more spikes. Moving forward, Kalman filter is selected to be used for algorithm integration as it is able to find the optimal statistical value for the next data point with feedback from the latest measured data, but the complementary filter does not.

This subsection shows the experimental result on the lean angle is measured using Kalman filter and complementary filter. The next section will show the experimental result for riding pattern classification.

Figure 7 LSTM confusion matrix



Notes: (Left): 0 – "Others", 1 – "Left Turn", 2 – "Right Turn", 3 – "Left lane change", 4 – "Right lane change", 5 – "Moving straight", LSTMconfusion matrix(Right): 0 "Others", 1 – "Left Turn", 2 – "Right Turn", 3 – "Moving straight"

Figure 8 Algorithms integrated example

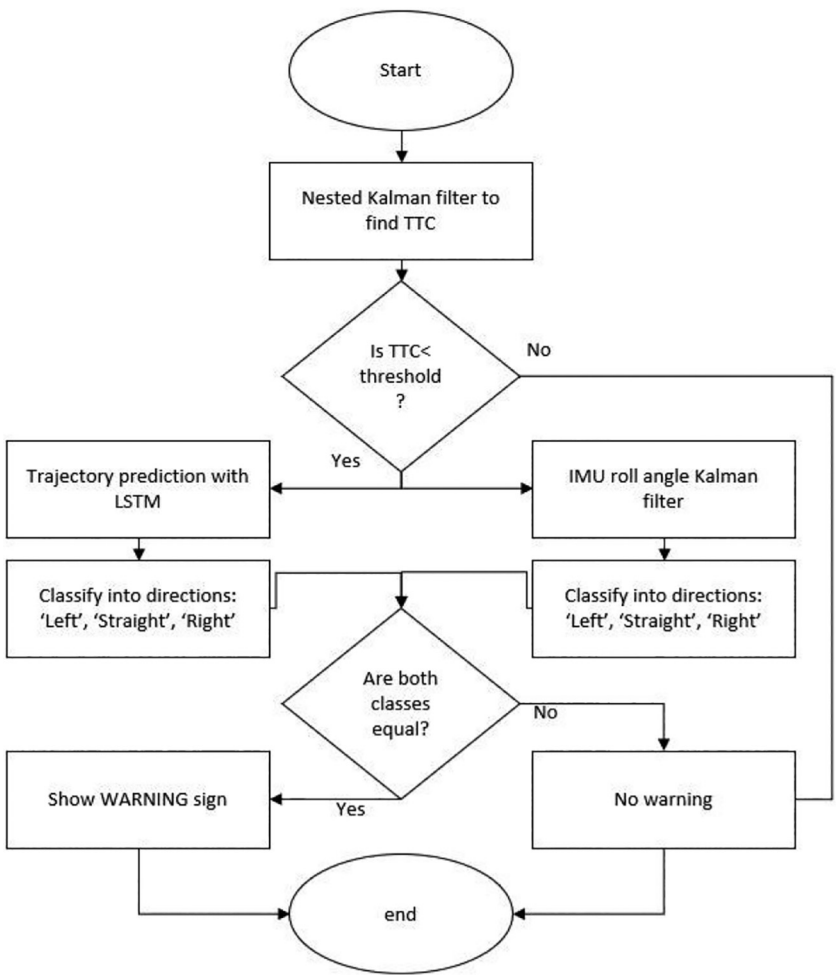
5.3.2 Results on riding pattern classification

For riding pattern classification, the 10 hz IMU data was used as a higher sampling rate will result in increased data inputs for LSTM, which makes it more complex. After processing with time step 40 and time shift of 20 and under-sampling, the data is trained for riding pattern classification. Initially, the network was trained with the six classes, which are Moving straight (S); Left Turn (LT); Right Turn (RT); Left lane change (LLC); Right lane change (RLC); and Others (O). The LSTM model will be compared with a Support Vector Machine (SVM) time series classification model using the toolkit by Tavenard *et al.* (2020).

However, upon training and testing on both LSTM and SVM, it was realized that the raw data labeled for "Right lane change," "Left lane change" and "Moving straight" are not very distinguishable as seen in Figure 7. This is because the sensors are unable to pick up any distinguishable pattern between moving straight and lane change as motorcycles only require a slight lean to change lanes. Another hypothesis is motorcycles move straight when changing lanes, but the straight movement does not correspond to the road lanes which allows them to change lane.

Next, to improve performance of the classification, the classes "Left lane change" and "Right lane change" were dropped from the data set and this new data set is which resulted in a rise in performance in terms of accuracy and the confusion matrix. As seen in Figure 7, there is significantly less confusion between the classes now that "Left lane change" and "Right lane change" are removed. The improved in performance is also reflected in the accuracy of the model as LSTM and SVM were able to achieve 83.0% and 80.3%, respectively. While 83.0% classification accuracy is not the

Figure 9 Algorithms integrated flowchart



best, the performance can be increase with a bigger data set and longer training time. With this algorithm, riding pattern classification can be used to predict the leaning direction as a "Left turn" and "Right turn" will indicate a left and right lean.

5.3.3 Comparison

In this subsection, using Kalman filter to find leaning angle and using LSTM for riding pattern classification will be compared to see which approaches can better augment the FCW system with leaning direction prediction. Because both approaches are fundamentally different in terms of input, output and time step, the metric of comparison will be how the algorithm is more applicable to the FCW system which is in terms of reaction time. Reaction time for the algorithm is very important for FCW system, which is why it will be used as a metric for comparison. In the Kalman filter, there is barely a lag to produce the output of leaning angle estimate as the algorithm only uses the previous data to estimate the roll angle. Hence, for Kalman filter, the reaction time will be 1-time step which is 0.1 s if a 10 hz data is used for leaning angle estimation. However, for LSTM, it requires an array of inputs with a pre-defined time step length, and the time step length used is 40-time steps for a good prediction. This means that for a data sampling frequency of 10 hz, it requires 4 s of data input before it can classify a riding pattern. 4 s is too long a time frame for an effective FCW system. For the training of the LSTM above, shorter time step length was used for training as well, but it did not produce any promising results. Hence, in terms of reaction time for the algorithm, Kalman filter leaning estimation will be the better option.

5.4 Advanced forward collision warning simulation

The algorithms discussed in Sections 2, 3 and 4 can be run independently from each other, and to integrate them together, additional steps must be done. The additional steps are segregating the image into different regions and having a threshold for the roll angle measurement. The image will be segmented into 3 parts which indicates left, straight or right and these regions will be used to classify where does the trajectory of the vehicle detected fall under. Similarly, the roll angle will have a threshold that classify if a bike is leaning to the left, straight or right.

As seen in Figure 8, there are a few other elements in the picture as compared to normal vehicle detection. At the top, it shows words in green that describe the direction the ego vehicle is moving. From the image, it says that the bike is moving straight. This indicates the direction of the ego vehicle. Below it, it shows a "WARNING" message in blue, and the warning signal will only trigger under the correct conditions. In the image, it has two vertical blue lines, which indicate the left, straight and right regions of the image. These regions are based on x pixel threshold, and it will indicate if a detected vehicle trajectory is within one of these regions. Toward the bottom of the image, there is a green bounding box with the numbers 2.34 on the top left of the green box. The green box indicates the bounding box

From the YOLO model and the number represents the time-to-collision in seconds. Beside the green box, there is a red dot that indicates the center pixel coordinates of the vehicle detected $h = 20$ steps ahead. From this image, it shows that while the trajectory is toward the left of the green box, the trajectory is still within the straight region of the image. As the bike motion is classified as straight, the trajectory direction is classified as straight, and the time-to-collision value is less than the threshold, the blue "WARNING" signal is shown. This signal can be in other forms, such as a blinking light or beeping sound for the user to be notified.

Figure 9 shows the flowchart for the Advanced FCW system, which integrates the time-to-collision nested Kalman filter, trajectory prediction with LSTM and IMU roll angle using Kalman filter. As seen in the chart, the Advanced FCW starts

Figure 10 Screenshot of the Advanced Forward Collision Warning system

off with the time-to-collision calculation as it is the most important factor in quantifying a danger. It is only after the time-to-collision value is below a certain threshold whereby the algorithm starts to predict the trajectory of the vehicle center pixel. This is arranged in such an order because trajectory prediction does not need to be run at every frame to reduce computational power. Next, because the IMU roll angle Kalman filter is run using a different sensor with different sampling rates, it runs individually from the time-to-collision calculation, and the position of roll angle with Kalman filter is just for illustration purposes. To use roll angle in the warning system, it must be synced with the sampling rate of the video, and interpolation and estimation can be used to sync them together. Next, after the trajectory prediction and IMU roll angle measurement, both data will be subjected to certain conditions which classify the direction of the ego and detected vehicle. When both classes are the same, and when the time-to-collision is below the set threshold, a warning signal will be triggered to alert the user of potential danger.

Figure 10 shows the screenshot at various parts of the Advanced FCW system across the 330 frames. It is observed that the car moves across the screen from right to left, and finally, a warning signal was given at the bottom right screenshot as the conditions follow Figure 9.

6. Conclusion

The goal of this paper is to develop an advanced FCW system for VRUs such as motorcyclists. This paper presents the various aspects of an Advanced FCW system which can be implemented independently or together. As seen in the results, the number of false positives for time-to-collision warning has been significantly reduced, which is very important to a motorcyclist on the road. Furthermore, as this advanced FCW system only requires a monocular camera and an IMU sensor, it can be implemented on a smartphone and thus increasing adoption and practicality rather than spending lots of money on additional hardware.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M. (2016), "Tensorflow: a system for large-scale machine learning", 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 265-283.

Alghamdi, T., Elgazzar, K., Bayoumi, M., Sharaf, T. and Shah, S. (2019), "Forecasting traffic congestion using ARIMA modeling", *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pp. 1227-1232.

Goli, S.A., Far, B.H. and Fapojuwo, A.O. (2018), "Vehicle trajectory prediction with gaussian process regression in connected vehicle environment?", *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 550-555.

Heravi, E.J. and Khanmohammadi, S. (2011), "Long term trajectory prediction of moving objects using gaussian process", *2011 First International Conference on Robot, Vision and Signal Processing*, pp. 228-232.

Hossam-E-Haider, M., Islam, T., Islam, M.S. and Shajid-Ul-Mahmud, M. (2017), "Comparison of complementary and Kalman filter based data fusion for attitude heading reference system", *AIP Conference Proceedings*, vol. 1919, doi: 10.1063/1.5018520.

Lauren, S. and Harlili, S.D. (2014), "Stock trend prediction using simple moving average supported by news classification", *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, pp. 135-139.

Lim, Q., He, Y. and Tan, U. (2018), "Real-time forward collision warning system using nested Kalman filter for monocular camera", *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 868-873.

Lim, Q., Johari, K. and Tan, U. (2019), "Gaussian process auto regression for vehicle center coordinates trajectory prediction", *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pp. 25-30.

Park, S.H., Kim, B., Kang, C.M., Chung, C.C. and Choi, J.W. (2018), "Sequence-to-sequence prediction of vehicle trajectory via LSTM Encoder-Decoder architecture", *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1672-1678.

Radziukynas, V. and Klementavicius, A. (2014), "Short-term wind speed forecasting with ARIMA model", *2014 55th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, pp. 145-149.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016), "You only look once: unified, Real-Time object detection", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788.

Salari, E. and Ouyang, D. (2013), "Camera-based forward collision and lane departure warning systems using SVM", *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1278-1281.

Schulz, J., Hubmann, C., Löchner, J. and Burschka, D. (2018), "Multiple model unscented Kalman filtering in dynamic Bayesian networks for intention estimation and trajectory prediction", *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1467-1474.

Sulandari, W. and Yudhanto, Y. (2015), "Forecasting trend data using a hybrid simple moving average-weighted fuzzy time series model", *2015 International Conference on Science in Information Technology (ICSITech)*, pp. 303-308.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K. and Woods, E. (2020), "Tslearn, a machine learning toolkit for time series data", *Journal of Machine Learning Research*, Vol. 21 No. 118, pp. 1-6, available at: http://jmlr.org/papers/v21/20-091.html

Zhao, J., Xie, B. and Huang, X. (2014), "Real-time lane departure and front collision warning system on an FPGA", *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-5.

Corresponding author

Qun Lim can be contacted at: qun_lim@mymail.sutd.edu.sg