A non-time series approach to vehicle related time series problems

Jonathan R. Wells[1], Kai Ming Ting[1] and Chandrasiri P. Naiwala[2]

[1]Gippsland School of Information Technology
Monash University, Australia
Email: {jonathan.wells,kaiming.ting}@monash.edu

[2]Toyota InfoTechnology Center Co., Ltd., Japan
Email: np-chandrasiri@jp.toyota-itc.com

Abstract

This paper shows that some time series problems can be better served as non-time series problems. We used two unsupervised learning anomaly detectors to analyse a vehicle related time series problem and showed that non-time series treatment produced a better outcome than a time series treatment. We also present the benefits of using unsupervised methods over semi-supervised or supervised learning methods, and rule-based methods.

1 Introduction and Motivation

Time series data treatments rely on the relationship between the points which are in a sequential time order; whereas, non-time series data treatments treat each point independently. Time series data conform to a natural ordering and the data indices often appear at a regular time step interval. Examples of time series applications are stock market, tracking an outbreak of a disease, a heart monitor or weather time series.

However, do all data, with a natural ordering based on time, need to be treated as a time series problem? In this paper, we look at a vehicle related time series problem to examine whether it can be solved as a non-time series problem.

We propose to use two anomaly detectors to solve this problem in a non-time series setting rather than in a time series setting. This paper shows that:

(i) A vehicle related time series problem is better treated as a non-time series problem and it can be effectively and efficiently solved using unsupervised learning anomaly detectors.

(ii) Rule-based approach, currently used in a non-time series setting, produces a set of rules in the form of a fixed linear model where some parameters modify the decision globally in the feature space. Any methods (e.g., McLaughlin et al., 2009; Knipling et al., 1993; Kiefer et al., 1999; Brunson et al., 2002) using this approach either cannot make or have difficulty in making these changes locally. The proposed approach can easily retrain a new model to cater for a new situation that has local changes only.

The key advantage of the proposed approach is that a time series problem becomes a simpler problem when treated as a non-time series problem. As a result, a simpler model can be used to solve this problem. The two anomaly detectors we employed, *i*Forest (Liu, Ting, and Z.-H. Zhou, 2008) and ORCA (Bay and Schwabacher, 2003), provide further advantage in providing flexibility in retraining a new model to suit different situations and users.

A vehicle related time series problem has all the characteristics of a time series problem with one unique property: a projection of any given data point can be determined, using the law of physics, by assuming that there are no further changes to the current actions between the driver and the approaching object. With this property, we can determine if there is a potential collision and issue an alert when require. This naturally leads to a non-time series analysis.

In this paper, features such as weather or road conditions are outside the scope because the main focus is to illustrate that a vehicle related time series problem could be solved as a non-time series problem.

We review related work in the next section. Section 3 describes the relationship between time series and non-time series treatments for a vehicle related time series problem. Section 4 provides a brief description of two anomaly detectors we employed to solve a vehicle related time series problem. Section 5 outlines the data sets and the feature selection process, and the type of models generated for evaluation. This is followed by the empirical evaluation in Section 6. We provide the conclusions in the last section.

2 Related Work

This section reviews two existing approaches to vehicle related time series problems. The first approach is represented by non-time series rule-based methods while the second approach is represented by time-series semi-supervised methods.

McLaughlin et al. (2009) evaluated three different collision avoidance systems to prevent rear-end crashes using a subset of the 100-car study (Neale et al., 2002; Dingus et al., 2006). The three different algorithms are rule-based methods using the time series data in a non-time series setting that treats each time step as an independent data point. Up to four sensor readings (range, speed, acceleration, and relative velocity) of the 'following' vehicle and up to two computed values (acceleration and velocity) for the 'leading' vehicle are used in each of the methods.

The first method, Knipling (Knipling et al., 1993), is a straight rule-based method with a constant allowing for the driver reaction time plus the braking time. The second method, CAMP Linear (Kiefer et al., 1999), adds to the model a set of coefficients which are derived from a regression analysis to accommodate different driver characteristics. The final

	Rule-based	Non-time series - unsupervised	Time series - semi-supervised
feature space	Use the original feature space		Feature space transformation
labels	No labels required		Labels are needed for some time series events
model	No model is built; cannot be retrained.	Model is built and it can be retrained when needed.	
complexity	Simplest because there are no models requiring to be built. Just plug in the values into a pre-defined model.	Simpler. Build a model using existing anomaly detectors and predictions are made from the model.	Complex as a new feature space needs to be constructed before building a model; and the model is non-linear and requires complex learning procedures.

Table 1: Differences between rule-based, non-time series unsupervised and time series semi-supervised methods

method, NHTSA (Brunson et al., 2002) which incorporates both the above features, has an additional feature that allows a driver to set different alert sensitivity settings.

The first two methods compute a warning range and compare this value to the actual sensor range value. An alert is issued when the actual range falls below the computed range. The third method computes a distance for the following vehicle to avoid a collision with the leading vehicle. This is combined with a threshold computed based on the velocity of the following vehicle. If the computed distance is less than the computed threshold then an alert is issued.

However, all these methods are 'hard-coded' linear model and do not allow for the changes in driver characteristics over time apart from the three different alert sensitivity levels in NHTSA.

Ning et al. (2010) presented a semi-supervised time series approach where each time series event is labelled 'crash' (if it actually happened) or 'safe' (if there are no crashes during the whole event). From a transformed feature space, a temporal difference learning (a form of reinforcement learning) is used to learn a 'danger level' function from training events, which exhibits low values if the vehicle approaches a crash point, and high values for safe events. A threshold can be used to trigger an alert if the output of the 'danger level' function is lower than the threshold. Their proposed non-linear method is found to perform better than a linear method, logistic regression and linear regression.

However, training a non-linear model is computationally expensive. Wang, Zhu, and Gong (2010) attempted to overcome the long training time by using faster parameter updating schemes instead. Otherwise, their approach is similar to Ning et al. (2010) in that both transformed the feature space in a similar manner and then modelled a danger level function from training events with two 'known' states.

The advantage of our proposed method over the rule-based methods are that i) we use an unsupervised learning method to train a model for prediction; and ii) the model can be retrained to allow for changes of driver characteristics over time. The advantages over the semi-supervised methods are that the problem is treated as non-time series and no labels are required which make the problem simpler. Table 1 summaries these advantages.

3 Treating time series as non-time series

The vehicle related time series problem (Neale et al., 2002; Dingus et al., 2006) that we investigated contains a number of driving sessions. Each session is a time series recording of the vehicle's states and any approaching objects detectable by the radars installed in the vehicle. A session starts from the time that a driver begins a journey until the vehicle has stopped due to either an unforeseen circumstance, such as a crash, or the driver has reached their destination. Each driving session is labelled: near crash or crash (if they occurred), or an incident-free event.

Although time series analysis had been used in the vehicle related time series problem (Ning et al., 2010; Wang, Zhu, and Gong, 2010), it can be treated as non-time series problem because the law of physics govern the vehicle and any impending objects at each time step during the driving session. This allows us to utilise the data in a non-time series setting because we can determine what the outcome would be at any data point, independent of other data points. Although we need to make an assumption that all conditions of that point remain constant, the calculated outcome is still valid to allow a vehicle warning system to issue an alert if there is an impending collision.

As such, every point in the time series is treated as an independent point using the law of physics; and it can be categorised into one of the following three labels: (i) 'unsafe' as a driver has minimal or no time to react to any impending collision between the vehicle and an approaching object, (ii) 'safe' as a driver has plenty of time to avert a crash, or (iii) 'alert' to avoid a crash if a driver is given a warning on time to take actions. We use a simple rule to define these three regions in a feature space in which a vehicle can be, in relation to an approaching object. The following two equations are used.

$$T = \frac{d}{v} \tag{1}$$

$$\mathcal{T} = \frac{v}{a} + c \tag{2}$$

where T is the time to impact between the vehicle and an approaching object; d and v are the distance and the relative velocity, respectively, between the vehicle and the object; $\frac{v}{a}$ is the vehicle deceleration time with the assumption that a certain deceleration rate a has been applied (due to timely braking of the vehicle). $\mathcal{T}$ is the minimum time required to avoid a crash. In order to avoid a crash, T must be greater than $\mathcal{T}$.

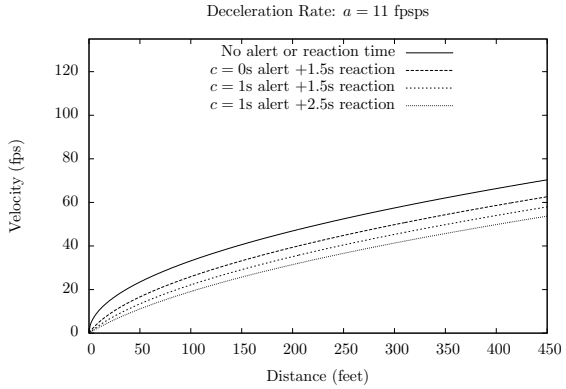
c in Equation 2 is the driver's response time to brake in order to avoid any potential collisions. It consists of the driver reaction time, R, plus an *extra* alert time W. The driver's response time is expressed as follows:

$$c = W + R \tag{3}$$

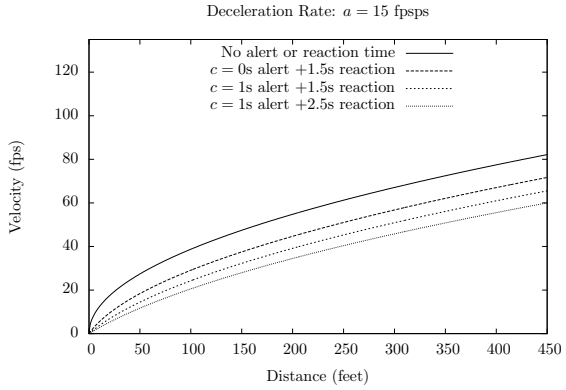
Solve for v using equations 1 and 2 gives:

$$v = \frac{a(\sqrt{c^2 + \frac{4d}{a}} - c)}{2} \tag{4}$$

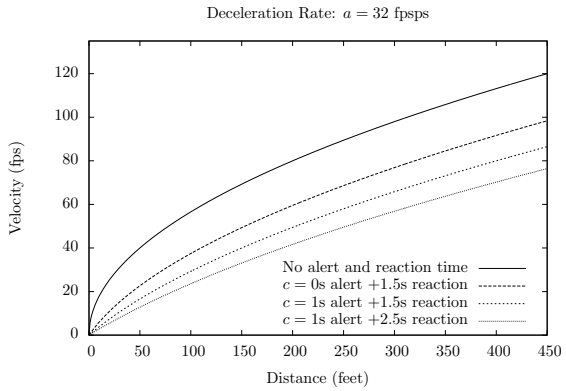
Using equation 4, we can plot different curves defining the boundaries between regions of potentially



(a)



(b)



(c)

Figure 1: Using equation 4, these charts show the different values for a, c, and d. The different a values represents the different deceleration rates for the driver ability to stop a vehicle. 11 fpsps is a conservative figure. 15 fpsps is the general figure. 32 fpsps is the figure for professional car racing drivers.

'unsafe' and 'safe' regions. Figure 1 shows the different curves for driver's ability to stop a standard vehicle and their response times in a given event. The figures show the effect of using different values for a and c (*A Policy on Geometric Design of Highways and Streets* 2004; McLaughlin et al., 2009). The region above the top curve is the 'unsafe' region. For different c values, the region below the 'c' curve is the 'safe' region; and the region between the 'c' curve and the top curve, is the alert region. A professional racing driver will have the ability to stop a vehicle much quicker than an average driver; therefore, has

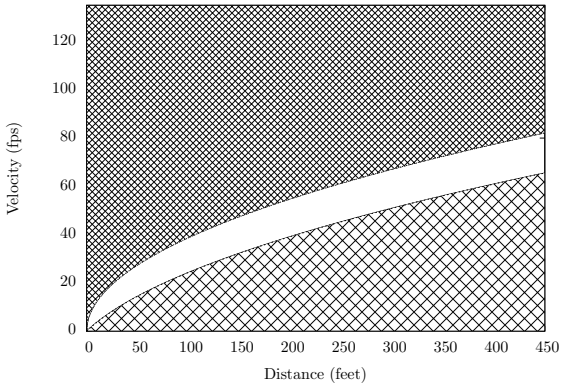


Figure 2: The different zones as determined by the simple rule. The hashed zone, at the bottom, signifies the safe region. The white zone signifies the alert region which also contains the driver reaction time. The hashed zone, at the top, signifies the unsafe region.

a greater 'safe' region than an average or beginner driver. This is reflected in the graphs.

The following constants are used to derive a simple rule. Deceleration rate a is set to 15 fpsps (feet per second per second) as outlined in *A Policy on Geometric Design of Highways and Streets* (2004) and the driver reaction time, R, is set to 1.5 seconds which represents the reaction time for 75% of the population (McLaughlin et al., 2009). An alert time, W, of 1 second is also set. Figure 2 shows the zones that were created by the simple rule. The hashed zone, at the top, is the area to be considered 'unsafe' according to the rule if there are no further changes to the driver's current action or the approaching object. The white zone is the area where an alert will be given and the hashed zone, at the bottom, shows the area where it is considered 'safe'.

The simple rule uses the unit measure of 'time' to determine whether a collision is imminent; whereas, the three non-time series methods (Knipling et al., 1993; Kiefer et al., 1999; Brunson et al., 2002), described in Section 2, use the unit measure of 'distance' to determine whether a collision is imminent. Both measures, time and distance, are derived using the same law of physics.

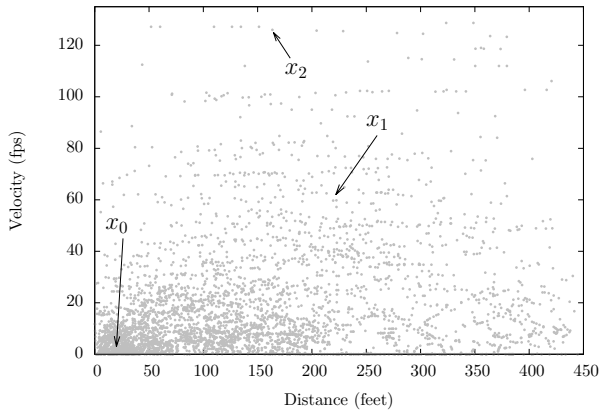
4 Unsupervised learning approach

In this research, we used two anomaly detectors called *i*Forest (Liu, Ting, and Z.-H. Zhou, 2008) and ORCA (Bay and Schwabacher, 2003). We provide a brief description of each of these detectors in the following two sections.

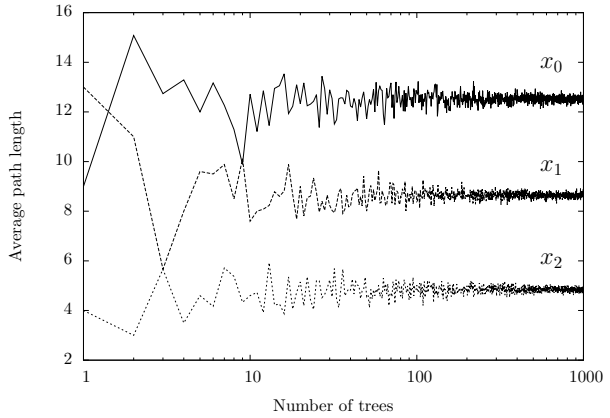
4.1 *i*Forest

Anomalies can be characterised by two quantitative properties: the number of anomaly points is small and these points have values that are significantly different to those of normal points (Liu, Ting, and Z.-H. Zhou, 2008).

Traditionally, anomaly detectors model the profile of the normal points and then identify points that do not conform to the normal profile as anomalies. However, these detectors are optimised for normal points and not for anomalies. Liu, Ting, and Z.-H. Zhou (2008) introduced a different kind of anomaly



(a) Showing 20% of the crash data reported in Section 5. x_0 is in the lower left region; x_1 is in the middle of the data cloud; x_2 is at the edge of the data cloud.



(b)

Figure 3: The result of isolating x_0, x_1 and x_2 using a number of iTrees. The average path length required to isolate x_0, x_1 and x_2 converged to 12.7, 8.5 and 4.8, respectively.

detector called *Isolation Forest* or iForest that isolates anomalies from normal points rather than modelling normal points. Given a data cloud, Liu, Ting, and Z.-H. Zhou (2008) show that anomalies can be isolated faster than normal points because anomalies are more susceptible to isolation than normal points.

The isolation mechanism using partitioning can be implemented using a random tree called *Isolation Tree* or iTree. iTree is a binary tree that is constructed as follows. An attribute is randomly selected at each node; then a random split point is chosen that divides the data space into two sub-regions. This is repeated until every point is isolated from the rest of the points. Points that are isolated quickly will appears at the top of the iTree. Therefore, anomalies are defined as those points that have shorter average path lengths than normal points for a given set of iTrees.

To provide an example, Figure 3 shows that the average path length required to isolate x_0, x_1 and x_2, from data showed in Figure 3(a). The average path length converged to 12.7, 8.5 and 4.8, respectively, as the number of iTrees increases; where x_0 is a point at the middle of a data cloud; x_1 is in the middle ring and x_2 is at the fringe.

iForest is an unsupervised learning method that constructs an ensemble of iTrees. The parameters required are: number of iTrees in the ensemble and the training subsample size used to build each iTree. Note that each iTree can be trained using a subsample size significantly smaller than the given data set. This allows iForest to be built fast and makes iForest one of the fastest anomaly detectors available. For ease of reference, the algorithms used to build an iForest are re-produced from Liu, Ting, and Z.-H. Zhou (2008) in the Appendix.

4.2 ORCA

The second anomaly detector we used is a k-nearest neighbour based anomaly detector called ORCA (Bay and Schwabacher, 2003). The algorithm uses a pair of nested loops with a pruning rule. This pruning rule is designed to speed up the nearest neighbour search by removing data points which fall below a threshold. It uses a distance metric to find the k nearest neighbours and then it computes an anomaly score to evaluate each point. The anomaly score can be any nearest neighbour based function such as the distance to the kth nearest neighbour or the average distance of the k nearest neighbours. Readers are referred to Bay and Schwabacher (2003) for other details of the algorithm.

4.3 Applying anomaly detectors to vehicle related time series problems

In applying a batch mode anomaly detector to a vehicle related time series problem, we must first represent it as a non-time series problem where every point is independent of each other. We can then use an anomaly detector to identify rare events, such as crash or near crash, as anomalies; and common no-incident events as normal points. Here, we hypothesise that drivers will be in the no-incident events most of the time and will be in the crash or near crash events a few times and far between events. This hypothesis fits in with the definition of what anomalies are.

However, for this type of application, we can also use the anomaly detector in a slightly different way. Here, the emphasis is not detecting anomalies but to define boundaries between unsafe and safe regions as defined in Figure 2.

4.4 Semi-supervised versus unsupervised

The type of anomaly detectors we explored in this paper are different from those employed in the literature e.g., Wang, Zhu, and Gong (2010); Y. Zhou et al. (2007); Ning et al. (2010). The existing methods mainly employ supervised and semi-supervised learning methods which require labelled data. The type of anomaly detectors we explore are unsupervised learning methods which do not require labelled data. The labels defined in Section 3 (provided by the simple rule) are required for verification and computation of detection performance measure only, and they are not used to train iForest and ORCA.

We are unable to compare with semi-supervised learning approaches described in Ning et al. (2010) and Wang, Zhu, and Gong (2010) because neither their software nor the data sets used are available.

5 Data sets used and their characteristics

5.1 Data and attribute selection

We use two existing data sets from the Virginia Tech Transport Institute (VTTI)[1]. These data sets were collected from a study of approximately 100 vehicles

[1] http://www.vtti.vt.edu/

Radar	# of crashed events	# of near crashed events
Front	57 (84%)	644 (85%)
Rear	61 (90%)	688 (91%)
Either	66 (97%)	754 (99%)

Table 2: The values represent the number of events that the radars are active. 'Either' indicates that one of the radars is active for a given event. Each figure in bracket is the percentage of events from the total of 68 crash events or 760 near crash events.

driven by 241 primary and secondary drivers who had driven approximately 3.2 million vehicle kilometres in approximately 43,000 hours over a period of twelve months (Neale et al., 2002; Dingus et al., 2006).

The first data set consists of 68 crash events and the second data set consists of 760 near crash events. Each event contains a snapshot of approximately forty seconds; and it is broken down into thirty seconds before the crash (or near crash) and ten seconds after. The data is made up of three distinct sections: sensors, video and the manually added information. The sensor data consists of seven components: pedals (accelerator and brake), indicators (left and right), motion (lateral, longitudinal and yaw), lane tracking, radar (front and rear), light intensity and GPS. The video data consists of the recording from five cameras in the vehicle cabin showing the four different views outside the vehicle and a single view of the driver's hands and feet, steering wheel and the instrument panels. The manual data is added by an analyst studying the video footages and recording what the driver was doing at the time of the event, the conditions of the road, traffic, weather, time of day (day or night time) and analyst's analyses.

The focus of our research is on the sensor data. The radar data appears to be the best component for this research because it contains all of the necessary information required to determine whether there will be any impending crashes. The radar data consists of two streams of data: the front radar for tracking objects in front of the vehicle and the rear radar for tracking objects behind the vehicle. Each radar is capable of tracking up to seven objects to a distance of approximately one hundred metres. Each radar provides the following measurements relative to the vehicle: the range to the object (distance), the rate of change (relative velocity) and azimuth (angle).

We analysed the data sets to see if we have enough data. The result of the initial analysis on the radar data are summaries in Table 2. For this research, we can use over 97% of the available crash event data and over 99% for near crash event data. The remaining two crash and six near crash events were detected by other means such as an analyst studying the video. It should be pointed out that only one of the two radars need to be active in order to predict an impending collision provided that the approaching object is in the radar zone of detection.

5.1.1 Non-Time Series Treatment

The non-time series treatment employs two attributes: distance and relative velocity. It treats every point as independent.

5.1.2 Time Series Treatment

For the time series analysis, we constructed a new feature space from the sensors data using the method as outlined by Wang, Zhu, and Gong (2010). This involved computing the relationship between two consecutive data point and then constructing a 'sliding-window' on the computed results. Finally, a series of statistical analysis are conducted for each window which produce a set of statistical features.

100-Car Study	Wang, Zhu, and Gong (2010)
lane distance - left lane distance - right	driver's lateral lane position
vehicle composite speed	driver's longitudinal velocity
yaw rate	steering angle
longitudinal acceleration and brake = off	longitudinal acceleration due to throttle
longitudinal acceleration and brake = on	longitudinal acceleration due to brake
radar range - forward	minimum range - opposite direction
radar range - rear	minimum range - same direction
not available	throttle depression fraction
not available	braking depression fraction

Table 3: Corresponding attributes between the 100-car study (Neale et al., 2002; Dingus et al., 2006) and the study by Wang, Zhu, and Gong (2010).

The selection of attributes is based on the method described by Wang, Zhu, and Gong (2010). They selected nine attributes from a set of thirty eight attributes: lane position, longitudinal acceleration and deceleration, longitudinal velocity, steering angle, throttle and brake depression fraction, and the closest object approaching the front or rear of the vehicle. Table 3 lists the corresponding attributes between the 100-car study (Neale et al., 2002; Dingus et al., 2006) and the study by Wang, Zhu, and Gong (2010). All of the attributes except throttle and brake depression fractions are used. The lane position is separated into two attributes: left and right side lane marking details. This produced eight available attributes to analyse the two data sets.

These eight attributes are then transformed from the original feature space into a new feature space, as in Wang, Zhu, and Gong (2010). For each of the eight attributes, four different attributes are constructed as follows: the original value (f), the first-order forward difference (Δf, f^2, and Δf^2). This produces a set of thirty two attributes for one time step. A sliding window with a length of 10 time steps is then applied. For each window, the following statistical information are calculated on each of the thirty two attributes: minimum, maximum, mean, and standard deviation. This produces the final set of 128 attributes for each window.

5.2 Data preparation

The crash data set contains 42,098 individual points across 68 events which has a total of 28,962 time steps[2]. This data set is checked for any abnormalities. A total of 138 points are removed because of negative distance measurements; and one point has an impossible velocity reading. Since we are only interested in points that are approaching the vehicle, 20,068 points that have objects moving away from the vehicles are also removed. This leaves 21,891 points to be used for the non-time series data treatment. We used all of the 28,962 time steps for the time series data treat-

[2]A radar can detect up to 7 objects simultaneously for a single time step. Each object, in the time step, is treated as a single point.

Data	Format	Labels			Not assigned	Total
		Safe	Alert	Unsafe		
Crash	Non-Time Series	16,684	3,122	2,085		21,891
	Time Series	9,278	2,487	1,526	15,059	28,350
Near Crash	Non-Time Series	418,633	46,178	47,138		511,949
	Time Series	158,587	25,492	25,188	126,169	335,436

Table 4: Number of instances in each of the labels as described in Section 5.3.

Model	Data Size	Number of Attributes		Notes
		Radar	New feature space	
1	21,891	2		Attribute used are distance and relative velocity between the object and vehicle.
2	51,891	2		Same as Model 1 plus a Gaussian distribution, with a mean at $x = 450, y = 0$ and variance of 1, consisting of 30,000 points.
A	28,350		128	The 128 attributes are derived from the 8 attributes in the original feature space.
B	28,350		10 - *i*Forest / 5 - ORCA	Number of attributes selected using Forward Greedy Search from the available 128 attributes.
C	28,350		6	The best six attributes selected by Wang, Zhu, and Gong (2010, shown in Table III).

Table 5: Training data description. Attributes in the original feature space are the attributes from the 100-car study. Attributes in the new feature space are derived from the attributes as described in Section 5.1.2.

ment[3] which includes objects moving away from the vehicle in order to maintain the time sequence.

For the near crash data set, there are 897,631 individual points across 760 events which has a total of 342,276 time steps. 13,396 are removed because of negative distance; 12 points are removed because of impossible velocity readings; and 372,274 points are removed for objects moving away from vehicle. This leaves a balance of 511,949 for the non-time series data treatment. All of the 342,276 time steps are used for the time series data treatment for the same reason given above for the crashed data.

5.3 Label assignments

This section describes how the labels are assigned for the purpose of verification and computation of detection performances.

5.3.1 Non-time Series

There are three labels: *unsafe*, *alert*, and *safe*. The individual labels are derived from using the simple rule (equation 4) with the following parameters. The boundary between unsafe and alert is defined as $c = 0$ and $a = 15$ fpsps (feet per second per second). The boundary between alert and safe is defined as $c = 2.5$ seconds and $a = 15$ fpsps as shown in Figure 2.

5.3.2 Time Series

A label is assigned to each time step using distance and relative velocity between the vehicle and the nearest approaching object using the same equation and parameters as in the non-time series treatment with one exception, i.e., no label is assigned if there are no objects being tracked for that particular time step.

Table 4 shows the final class assignments for both the crash and near crash data sets.

[3]Actually, this figure is reduced to 28,350 because of the sliding window effect. This is because each window has a length of 10 points in our setting, and any windows that does not have 10 points are discarded. This happens at the end of each event (68 * 9 = 612). Note that the near crashed data has the same effect.

6 Empirical Evaluations

The aim is to assess the utility of the time series treatment and the non-time series treatment. The evaluation assesses three different time series models in the new feature space: Model A is constructed using all of the 128 attributes; Model B is the best model using a subset of attributes chosen from a Forward Greedy Search; and Model C is constructed by using the best attributes as outlined by Wang, Zhu, and Gong (2010). Two non-time series models are constructed with the two attributes obtained from the radars. Model 1 is constructed using the distance and relative velocity between the vehicle and an approaching object. The current distribution of Model 1 is biased because the data sets used consists of the crash or near crash events only. This does not represents the true distribution because most events has majority of the points in the safe region. Model 2 is constructed by adding a Gaussian distribution, consisting of 30,000 synthetic points, to the lower right corner of the original data space. The details of all five models are summarised in Table 5.

*i*Forest and ORCA are employed to generate each of the above five models to assess the relative performance among the five models and between *i*Forest and ORCA.

6.1 *i*Forest's ranking capability

Figure 4 shows *i*Forest's ranking results of the crash non-time series data. Figure 4(a) shows the result of top rankings which includes the first 500 unsafe points above the top curve. This ranking includes 76 safe points below the bottom curve and 16 alert points between the top and bottom curves which are ranked higher than the top 500th ranked unsafe point. Each of the following figures, 4(b) to 4(d), contains the next set of top rankings which includes the next set of 500 unsafe points along with alert and safe points which were ranked higher than the corresponding 500th ranked unsafe points. The final figure, 4(e), shows the remaining 85 unsafe points along with the other points.

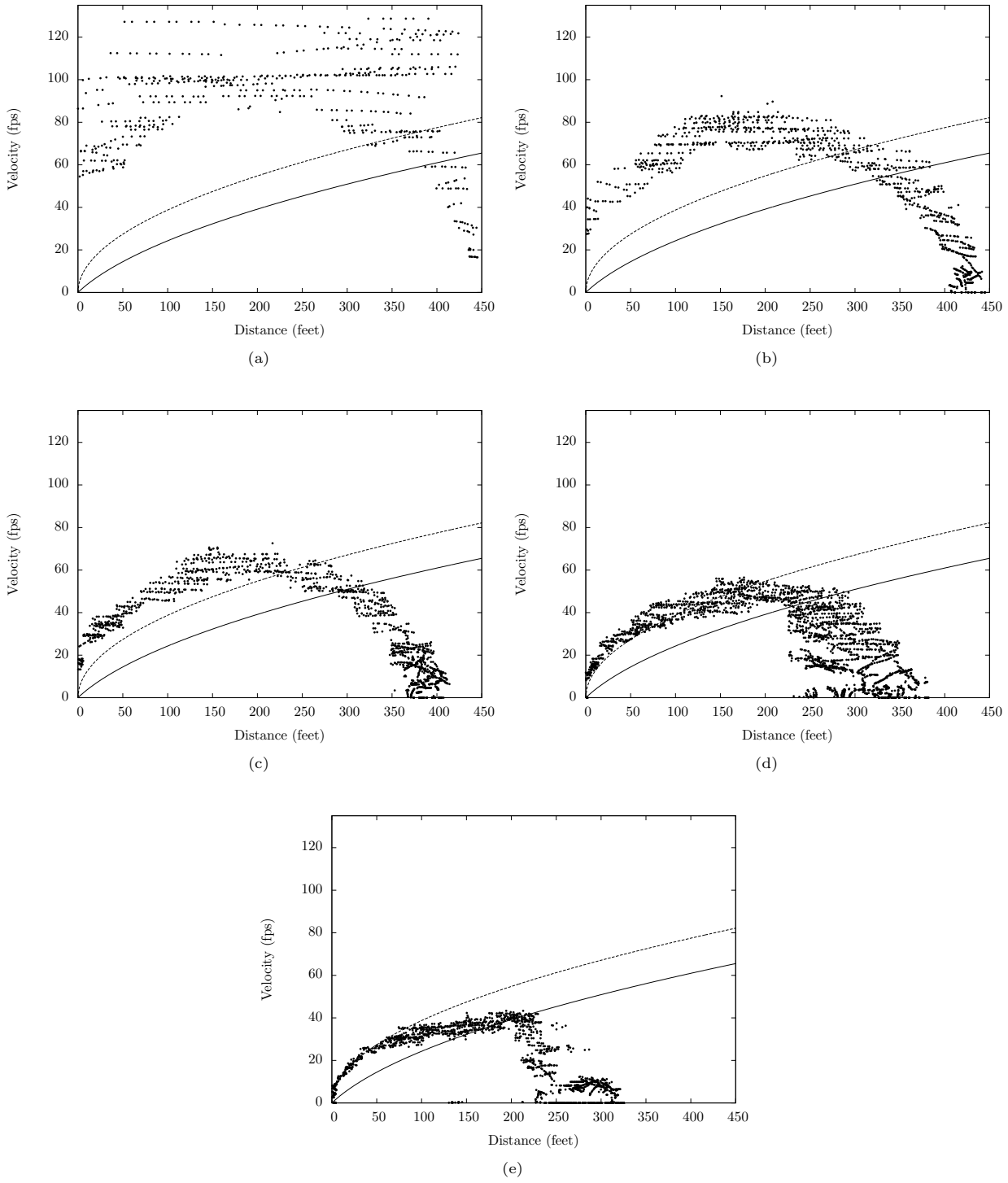
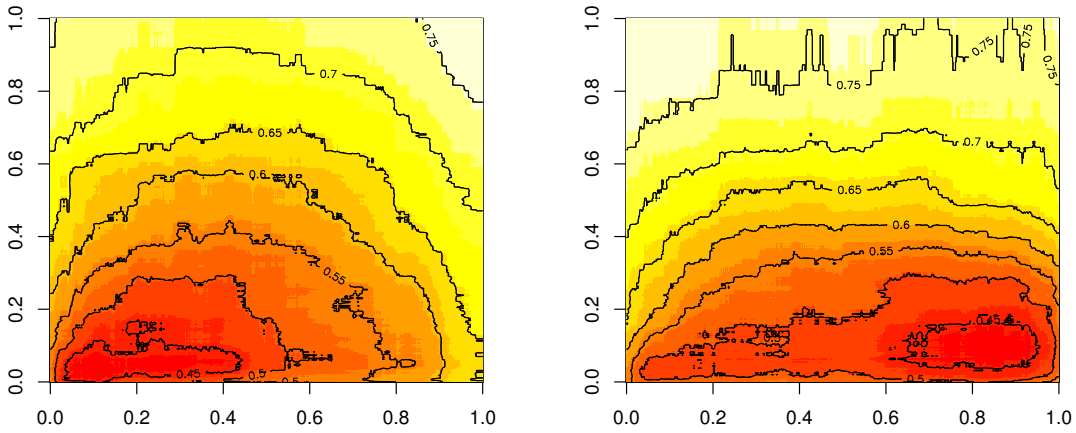


Figure 4: The ranking results of *i*Forest, trained using the crash non-time series data. Figure (a) shows the first 500 unsafe points plus the additional alert and safe points. Figure (b) through to figure (d) are the next subsequent sets of 500 unsafe points respectively. Figure (e) shows the last 85 unsafe points.



(a) Model 1 (b) Model 2

Figure 5: Contour maps of *i*Forest for Models 1 and 2. They are produced using the anomaly scores output from *i*Forest. The anomaly scores range from 0 to 1. The anomaly score function can be found in the Appendix.

	Detector	Crash					Near Crash				
		Time Series			Non-Time Series		Time Series			Non-Time Series	
		Model A	Model B	Model C	Model 1	Model 2	Model A	Model B	Model C	Model 1	Model 2
AUC	*i*Forest	0.7244	0.8997	0.6631	0.9531	0.9955	0.6873	0.8576	0.5259	0.9696	0.9963
	ORCA	0.7579	0.8883	0.6724	0.9389	0.9372	0.6339	0.8384	0.4980	0.9917	0.8359
Time	*i*Forest	12	502	2	1	2	100	21	22	22	21
	ORCA	478	55,446	10,049	26	7720	6,314	9,135	222,416	606	164,991

Table 6: The AUC and timing results using two different anomaly detectors. For ORCA, each best k value is searched from 5, 10, 20, 40, 60, 80, 150, 250, 300, 500, 1000, 2000, 3000, and 4000 while using the crash data. The best k values are 10, 10, 4000, 80, 4000 for Model A, Model B, Model C, Model 1 and Model 2, respectively. The same corresponding k values, for each of the models, are used for the near crash data. For *i*Forest, there are no parameter search. The timing results are given in seconds.

	*i*Forest				ORCA			
Iter #	Attribute #	Attribute Name	Stats	AUC	Attribute #	Attribute Name	Stats	AUC
1	109	min dist front	min Δf^2	0.8724	106	min dist front	mean Δf	0.8043
2	109,99	min dist front	std dev f	0.8878	106,109	min dist front	min Δf^2	0.8705
3	109,99,95	Decel	std dev f^2	0.8943	106,109,93	Decel	min Δf^2	0.8859
4	109,99,95,32	vehicle velocity	max f	0.8928	106,109,93,125	min dist rear	min Δf^2	0.8870
5	109,99,95,32,58	yaw	mean Δf	0.8950	**106,109,93,125,61**	**yaw**	**min Δf^2**	**0.8883**
6	109,99,...,58,106	min dist front	mean Δf	0.8958	106,109,...,61,45	vehicle velocity	min Δf^2	0.8883
7	109,99,...,106,126	min dist rear	mean Δf^2	0.8970	106,109,...,45,29	right lane marker	min Δf^2	0.8880
8	109,99,...,126,45	vehicle velocity	min Δf^2	0.8987	106,109,...,29,13	line lane marker	min Δf^2	0.8879
9	109,99,...,45,104	min dist front	max Δf	0.8970	106,109,...,13,127	min dist rear	std dev Δf^2	0.8874
10	**109,99,...,104,71**	**acceleration**	**std dev f^2**	**0.8997**	106,109,...,127,119	min dist rear	std dev f^2	0.8871

Table 7: The AUC results of Model B for the Forward Greedy Search using *i*Forest and ORCA.

The current data distribution has the highest concentration of points to the lower left corner, and the anomalies are around the perimeter to the top and right of the data cloud. Based on this data distribution, the results shown in Figure 4 demonstrated that *i*Forest has correctly ranked all these data points— the most normal points are at the center of the data cloud which are ranked at the bottom of the list and the most outlying points are ranked at the top.

The contour maps of *i*Forest for Models 1 and 2 are shown in Figure 5(a) and Figure 5(b), respectively. Note that the centre of the data cloud shifted from the bottom-left corner, in Figure 5(a), to the bottom-right corner, in Figure 5(b). This is a result of introducing a Gaussian distribution of synthetic points described in Section 6. Also note that Model 2, shown in Figure 5(b), can now better model the two boundaries shown in Figure 4.

6.2 AUC and runtime comparisons

We measure the detection performance of anomaly detectors in terms of area under ROC curve (AUC). In order to calculate AUC, the three labels need to be converted to a two-label problem. We are only interested in determining the anomalies — unsafe points. Alert is merged with safe to become the second label — safe. A perfect AUC score will have all of the unsafe points ranked at the top of the list.

Table 6 shows the AUC results for both *i*Forest and ORCA. The time series results clearly show that Model B, using features selected from the Forward Greedy Search, outperformed Models A and C.

Table 7 shows the results for Model B using the first ten attributes selected by the Forward Greedy Search in the first ten iterations. *i*Forest produces the best Model B result using 10 attributes and ORCA uses 5 attributes.

However, Model B in time series treatment still performs worse than Models 1 or 2 in non-time series treatment in terms of AUC, as shown in Table 6. This is the same for both crash and near crash data sets, regardless of *i*Forest or ORCA is used.

The results also show that *i*Forest runs significantly faster than ORCA, up to five orders of magnitude faster.

These results reveal that it is unnecessary to perform the tasks in time series which requires more features, additional computation and feature space transforming; whereas, treating each point independently as in a non-time series problem works well.

6.3 Summary

This paper investigates whether anomaly detectors can be used to alert a driver of an impending crash. We show that anomaly detectors can correctly rank outlying points of the given data distribution. Because the current data sets are collected solely from crash or near crash events only, the data distribution is bias and does not represent the true distribution. However, we have demonstrated that *i*Forest can be easily trained with a different data distribution (ie. adding a Gaussian distribution to the existing data) to correct the data distribution bias and provide a better ranking result for this type of application. The current result also implies that an anomaly detector could potentially become the core of a vehicle warning system that has the flexibility to allow car manufacturers as well as car drivers to tailor the system to suit individual needs. This is because such an anomaly detector can be easily retrained to adapt to new requirements of individual users. A vehicle warning system based on a rule-based approach (Knipling et al., 1993; Kiefer et al., 1999; Brunson et al., 2002) lacks this kind of flexibility; and the one based on a semi-supervised time series model is unnecessary complicated and may not work as well.

7 Conclusions

In this paper, we presented a study to examine whether a time series problem can be solved more effectively as a non-time series problem. In a vehicle related time series problem, we found that it can be solved as a non-time series problem with significantly improved AUC result compared to that achieved in time series. We also highlighted the disadvantages of existing methods: the rule-based approach is 'hard-coded', and the semi-supervised approach requires labels and significantly more features and a feature transformation, making the problem unnecessarily complex with no additional benefits.

We have demonstrated that anomaly detectors can be used in a way to create boundaries between safe and unsafe regions. These boundaries can be locally altered from time to time when required to adapt to the individual driver requirements which are hard or not possible to do with rule-based methods.

Although both *i*Forest and ORCA show comparable AUC results, *i*Forest is preferred because it runs significantly faster than ORCA, up to five orders of magnitude faster in our experiments.

Compare to the rule-based and semi-supervised approaches and ORCA, we conclude that *i*Forest is the best algorithm to be incorporated into a vehicle warning system to provide alerts to drivers for any impending collisions because of its fast execution, simplicity and flexibility.

References

A Policy on Geometric Design of Highways and Streets (2004). 5th. American Association of State Highway and Transportation Officials (AASHTO).

Bay, Stephen D. and Mark Schwabacher (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD-03)*. ACM, pp. 29–38.

Brunson, S., E. Kyle, N. Phamdo, and G. Preziotti (2002). *Alert algorithm development program — NHTSA rear-end collision alert algorithm - Final report.* Tech. rep. DOT-HS-809-526. Washington, DC: National Highway Traffic Safety Administration.

Dingus, T. A. et al. (2006). *The 100 Car Naturalistic Driving Study, Phase II — Results of the 100 Car Field Experiment.* Tech. rep. DOT HS 810 593. Virginia Tech Transportation Institute.

Kiefer, R.J., D. LeBlanc, M. Palmer, J. Salinger, R. Deering, and M. Shulman (1999). *Forward Collision Warning Systems: Development and Validation of Functional Definitions and Evaluation Procedures for Collision Warning/Avoidance Systems.* Tech. rep. DOT-HS-808-964. Washington, DC: National Highway Traffic Safety Administration.

Knipling, R., M. Mironer, D. Hendricks, L. Tijerina, J. Everson, J. Allen, and C. Wilson (1993). *Assessment of IVHS countermeasures for collision avoidance: Rear-end crashes.* Tech. rep. DOT-HS-807-995. Washington, DC: National Highway Traffic Safety Administration.

Knuth, D. E. (1998). *Art of Computer Programming, Volume 3: Sorting and Searching.* 2nd Ed. Addison-Wesley Professional.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). Isolation Forest. In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 08).* IEEE Computer Society, pp. 413–422.

McLaughlin, Shane B., Jonathan M. Hankey, Thomas A. Dingus, and Sheila G. Klauer (2009). *Development of an FCW Algorithm Evaluation Methodology With Evaluation of Three Alert Algorithms.* Tech. rep. DOT HS 811 145. Virginia Tech Transportation Institute.

Neale, V.L., S.G. Klauer, R.R. Knipling, T.A. Dingus, G.T. Holbrook, and A. Petersen (2002). *The 100 Car Naturalistic Driving Study, Phase 1 — Experimental Design.* Tech. rep. DOT HS 809 536. Virginia Tech Transportation Institute.

Ning, H., W. Xu, Y. Zhou, Y. Gong, and T. S. Huang (2010). A general framework to detect unsafe system states from multisensor data stream. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3, pp. 4–15.

Wang, Jinjun, Shenghuo Zhu, and Yihong Gong (2010). Driving Safety Monitoring Using Semisupervised Learning on Time Series Data. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3, pp. 728–737.

Zhou, Y., W. Xu, H. Ning, Y. Gong, and T. Huang (2007). Detecting unsafe driving patterns using discriminative learning. In: *Proceedings of 2007 IEEE International Conference Multimedia Expo (ICME 2007)*, pp. 1431–1434.

Appendix – *i*Forest Algorithms

Algorithms 1 and 2 show the steps required to train an *i*Forest. The tree height for each *i*Tree is set automatically to $l = ceiling(\log_2 \psi)$ which is an approximation of the average tree height (Knuth, 1998), where ψ is the sub-sampling size. Each *i*Tree is grown up to the average height because we are only interested in data points that have shorter-than-average path length.

Algorithm 1 : *i*Forest(X, t, ψ)

Inputs: X - input data, t - number of trees, ψ - sub-sampling size
Output: a set of t *iTrees*

Initialise $Forest$
set height limit $l = ceiling(\log_2 \psi)$
for $i = 1$ to t **do**
  $X' \leftarrow sample(X, \psi)$
  $Forest \leftarrow Forest \cup iTree(X', 0, l)$
end for
return $Forest$

Algorithm 3 shows how to compute the path length for a given data point. $c(\psi)$ is defined as follows.

$$c(\psi) = 2H(\psi - 1) - (\frac{2(\psi - 1)}{\psi}). \qquad (5)$$

where $H(i)$ is a harmonic number and it is estimated by $\ln(i) + 0.5772156649$ (Eulers constant).

Algorithm 2 : iTree(X, e, l)

Input: X - input data, e - current tree height, l - height limit
Output: an iTree

if $e \geq l$ or $|X| \leq 1$ **then**
  return $exNode\{Size \leftarrow |X|\}$
else
  let Q be a list of attributes in X
  randomly select an attribute $q \in Q$
  randomly select a split point p from max and min values of attribute q in X
  $X_l \leftarrow filter(X, q < p)$
  $X_r \leftarrow filter(X, q \geq p)$
  return $inNode\{Left \leftarrow$ iTree$(X_l, e + 1, l)$,
    $Right \leftarrow$ iTree$(X_r, e + 1, l)$,
    $SplitAtt \leftarrow q$,
    $SplitValue \leftarrow p\}$
end if

Algorithm 3 : PathLength(x, T, e)

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x

if T is an external node **then**
  return $e + c(T.size)$ {$c(.)$ is defined in Equation 5}
end if
$a \leftarrow T.splitAtt$
if $x_a \geq T.splitValue$ **then**
  return PathLength$(x, T.right, e + 1)$
else $\{x_a < T.splitValue\}$
  return PathLength$(x, T.left, e + 1)$
end if

To find out the anomaly score for a data point, the path length is first obtained and then the anomaly score is computed by using equation 6.

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}} \qquad (6)$$

where $h(x)$ is the path length from a single *i*Tree; $E(h(x))$ is the expected path length of *i*Forest.