Using Hidden Markov Models in Vehicular Crash Detection

Gautam B. Singh, *Senior Member, IEEE*, and Haiping Song

***Abstract*—This paper presents a system for automotive crash detection based on hidden Markov models (HMMs). The crash pulse library used for training comprises a number of head-on and oblique angular crash events involving rigid and offset deformable barriers. Stochastic distribution characteristics of crash signals are validated to ensure conformity with the modeling assumptions. This step is achieved by analyzing the quantile–quantile (Q–Q) plot of actual pulses against the assumed bivariate Gaussian distribution. HMM parameters are next induced by utilizing the expectation–maximization (EM) procedure. The search for an optimal crash pulse model proceeds using the "leave-one-out" technique with the exploration encompassing both fully connected and left–right HMM topologies. The optimal crash pulse architecture is identified as a seven-state left–right HMM with its parameters computed using real and computer-aided engineering (CAE)-generated data. The system described in the paper has the following advantages. First, it is *fast* and can accurately detect crashes within 6 ms. Second, its implementation is *simple* and uses only two sensors, which makes it less vulnerable to failures, considering the overall simplicity of interconnects. Finally, it represents a *general* and *modularized algorithm* that can be adapted to any vehicle line and readily extended to use additional sensors.**

***Index Terms*—Automotive crash detection, computer-aided engineering (CAE), continuous-value emission hidden Markov models (HMMs), crash pulse, discrete-value emission HMM, finite-element analysis (FEA).**

I. Introduction

With THE vast majority of vehicle consumers citing safety features as an important factor in their purchasing decision, crashworthiness of vehicles plays an important role in establishing quality and customer acceptance. In addition to meeting customer expectations, tangible benefits realized by a reduction in fatal and nonfatal injuries are estimated to be around $US 2 billion [1].

From a technological standpoint, an early detection of crash event allows safety engineers to better tailor airbag inflation rates, pressures, and belt-pretensioning systems. The automobile industry continues to make advancements by providing smarter restraint systems. The current crash-detection techniques are based on multistage sequential signal analysis algorithms and are generally tuned to detect specific types of crash events. These enhancements over the performance of the first-generation products were achieved through the incorporation of complex processing logic [2]. The fundamental paradigm of analyzing the signal parameters in the time domain continues to provide the necessary underpinnings to algorithmic enhancements.

Existing crash detection algorithms can be divided into two categories, namely, 1) those that utilize changes in vehicle speed and 2) those that utilize the vehicular crush information. The speed-based sensors rely on variables such as speed change ΔV, jerk, speed, displacement, energy, etc. The crush-based algorithms utilize data from one or more sensors mounted in the crush zone, which help predict crash severity. In both cases, crash-sensing algorithms are developed using a parameter tuning process where engineers analyze a library of crash pulses and develop rules and thresholds for airbag deployment. The resulting algorithms are not universal.

In recent years, crash-detection algorithms have begun utilizing machine learning and pattern-recognition techniques. For example, Barnard and Riesner's algorithm uses momentum and energy in semimetric spaces [3]. Singh and Song's earlier algorithm utilized a discrete Markov model to detect crashes [4]. Yin *et al.* proposed an algorithm based on artificial neural networks [5]. The goal has been to develop smart airbag systems that not only can predict crash severity but can also consider the occupant's size, weight, position, and velocity in a multistage bag deployment regime. Such a task is fairly complex but achievable through a *trainable* algorithm that utilizes inductive learning procedures such as that described in this paper. We demonstrate that a pattern recognition system trained on a large variety of crash acceleration classes constitutes an effective solution for designing smart airbag systems.

Hidden Markov models (HMMs) are being applied in many fields [6] such as control [7]–[9], communications [10], [11], handwriting and text recognition [12], [13], image processing, computer vision [11], [15], [16], and bioinformatics [17]–[19]. Most state-of-the-art speech systems today are based on HMMs [20], [21]. The reason that HMMs succeed in accurately modeling the real-world phenomenon is attributable to their flexible architecture and stochastic learning bases, rendering them generally immune to overtraining when an appropriate number of hidden states are used, and sufficient observation distributions and training data are used for learning.

An HMM models a doubly stochastic process: First, there is an underlying stochastic process that is not observable, and second, this hidden process influences another process that produces the sequence of observable symbols. When applied to the

Manuscript received July 5, 2005; revised November 18, 2005, November 27, 2006, July 28, 2007, November 6, 2007, February 5, 2008, April 10, 2008, and May 5, 2008. First published July 22, 2008; current version published March 17, 2009. The review of this paper was coordinated by Dr. M. S. Ahmed.

The authors are with the Department of Computer Science and Engineering, Oakland University, Rochester, MI 48309 USA (e-mail: singh@oakland.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2008.928904

crash-detection problem, the observed symbols are the observed accelerometer readings. The hidden states correspond to the underlying crushing, recoil, elastic, and plastic deformations caused by crash dynamics. Since these hidden changes should be similar for similar types of vehicles and crash situations, it should be possible to reliably correlate symbols observed and recorded by the accelerometer sensors in an aggregate sense.

This paper extends our earlier work on discrete HMMs where the underlying crash-detection model utilized quantized crash pulse data [4]. In this paper, the crash pulse data is not quantized and used for inducing a continuous-mode HMM. Our results with the continuous-mode HMM are far more superior than those obtained using a discrete HMM. The continuous-mode HMM detects a large variety of crash events within 6 ms of onset.

II. Crash Pulse Library

Crash libraries are generally developed by the OEM airbag module suppliers in close collaboration with automobile manufacturers. The data set used in this paper is from a complete crash library that was actually used by an automobile manufacturer. A crash library contains a collection of acceleration data measured by accelerometers. A crash library contains accelerometer values measured in real-world experiments where a specific vehicle is test-crashed under a set of controlled variables, including the vehicle speed, barrier type, and angle of impact. A crash library may also contain 3-D accelerometer samples obtained by simulating a crash using computer-aided engineering (CAE) models.

Acceleration Data: The vehicle acceleration values used for crash sensing are measured by an electronic sensor called the accelerometer. It measures instantaneous values for vehicle acceleration as it undergoes crash. The accelerometer produces a 3-D analog signal proportional to the acceleration of the vehicle in three dimensions: X—forward, rear to front of the vehicle; Y—right, driver to passenger side; and Z—down, roof to floor.

Fig. 1 shows the acceleration signatures in X- and Y-directions, or the crash pulse, for two cases. In all cases, the crash event occurs at time $t = 0$. Fig. 1(a) shows crash pulses for an event that does not require airbag deployment. The magnitude of acceleration sensed in these cases is bounded by about 10 g but can go up as high as 50 g. Crash pulses requiring airbag deployment are shown in Fig. 1(b). The magnitude of instantaneous accelerations along the X-axis in this case can reach up to 300–500 g.

Crash Pulse Groups: Generally, the crash pulses in a library may broadly be classified into three groups.

1) The first group consists of the nondeployment cases, including low-speed and rough-road driving conditions. A vehicle-to-barrier crash of 6–9 mi/h is considered to be a low-speed crash.
2) The second group consists of medium-speed crashes of 12–25 mi/h. Crash types can be diverse and include head-on collisions and pole-impact signals. Generally, crashes in this group do not require the deployment of the airbag either. However, the data set usually contains at least one crash condition above the deployment threshold [2]. Typically, the pulses in this group are utilized to set up thresholds.
3) The third group includes higher speed crashes of various types, including head-on and angular (usually 30°) impacts with a barrier, crashes with a center pole, as well as crashes with offset poles.

Sensor Locations: Crash pulse libraries are a collection of crash pulses measured by accelerometers mounted at specific locations in the vehicle. Typical crash sensor locations include the radiator bracket, inside fender, frame rails, steering wheel, pillar rockers, etc. In general, two crash pulses or acceleration signatures are recorded for each crash experiment [22]. One of the crash pulses recorded is for the crush zone located in the engine compartment. The crush zone in a vehicle is designed to absorb a significant portion of the crash energy as it crushes during impact. The other crash pulse is recorded from the passenger compartment, where the goal is to minimize the mechanical deformations and maximize its integrity during a crash.

In our case, the crash pulse library, whether actual or simulated, consists of acceleration values in the X-direction only for two sensor locations: one each in the crush zone and the passenger compartment.

A. Crash Pulses: Real Crashes

While learning inductive models, the *quality* and *quantity* of the observations govern the overall performance of a classification system. Since the dimensionality of the problem is generally high, and the training data are finite and nonuniformly sampled, the modeling problem is almost always ill-posed [23]. Within the context of crash classification, engineers construct a crash pulse library composed of accelerometer readings for a carefully selected set of crash events that would hopefully capture the variability in pulse signatures.

Crash pulse data are captured by crashing vehicles by colliding them against each other, against rigid and deformable barriers, and against stationary poles and moving barriers [22]. Impact configurations are varied as well with front, rear, bumper override, and overlap collisions. The acceleration values are recorded and collectively constitute a crash pulse library for a vehicle line that the manufacturer uses to calibrate its sensors for that line.

An analog-to-digital converter transforms the analog acceleration signal values into a uniformly sampled digital time series. The accelerometer crash pulse is prefiltered at a rolloff frequency of 4 kHz and then sampled at a rate of 12 500 samples/s. A digital filter SAE J211 is utilized to obtain the useful frequency components of the acceleration signal [22]. As the acceleration experienced by the vehicle is dependent on the physical characteristics of the vehicle and the barrier, the crash pulses in a library are a unique signature of a specific vehicle.

Fig. 2 compares the characteristics of a "nondeploy" and a "deploy" crash pulse. The waveform characteristics before and after it has been filtered are illustrated. The Δv profile of the

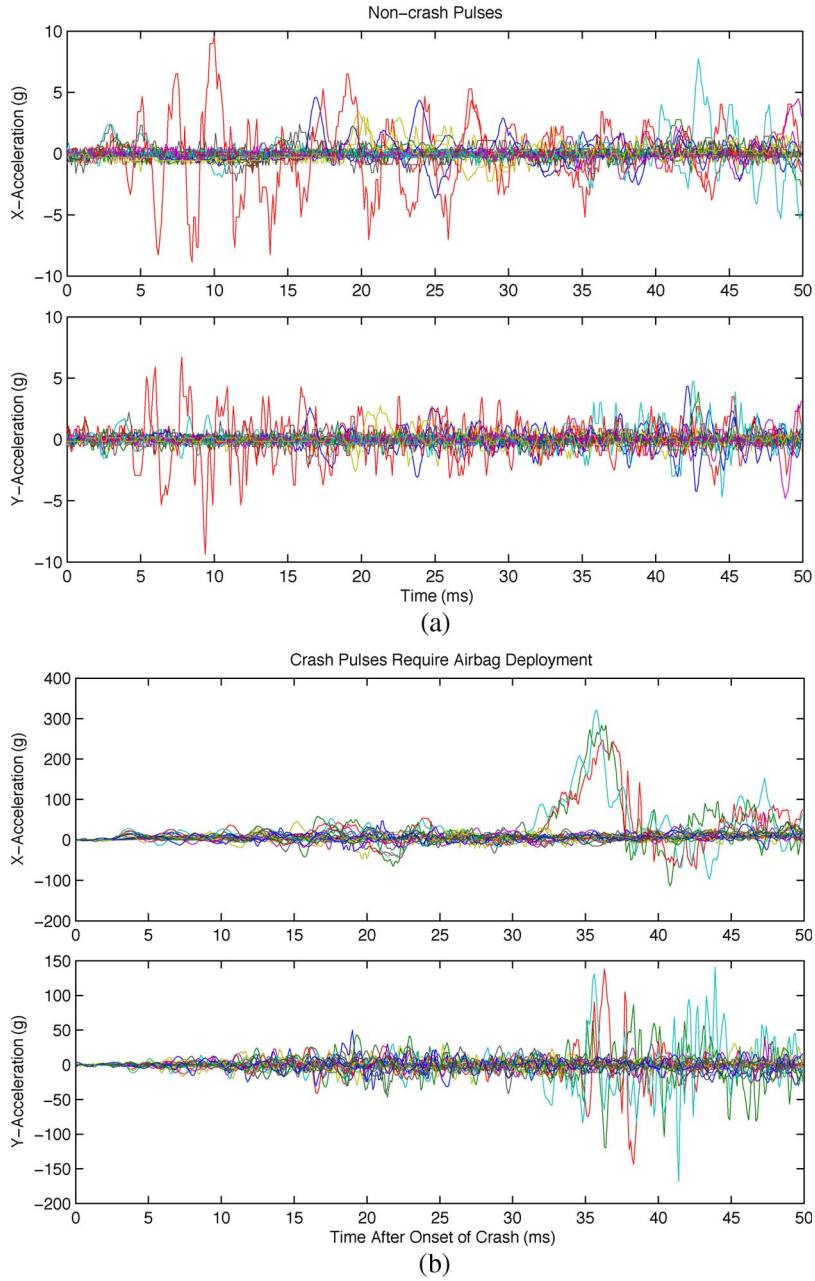


Fig. 1. (a) Acceleration in X- and Y-directions during a noncrash scenario. (b) Acceleration in X- and Y-directions for a crash that requires airbag deployment.

two pulses is also shown. The slightly convex shape of the Δv profile in Fig. 2(b) is indicative of the rapid drop in velocity to bring the vehicle to a stop in about the same time as the nondeploy case in Fig. 2(a).

B. Crash Pulses: Simulated Crashes

Due to the expense of crashing a vehicle for adding a crash pulse to the library, the number of pulses in the library tends to be limited due to economical concerns. The advancement of CAE provides indispensable tools in today's vehicle design. Today, in addition to the actual measurements of acceleration data, a crash library contains data obtained from running finite-element analysis (FEA). Simulated pulses from FEA can provide accurate lower frequency range information [24]. Fig. 3 compares the real crash data and the simulated data generated by FEA for signals observed in the noncrush and crush sensors[1] in a 35-mi/h frontal barrier impact crash test.

The FEA methods are based on mathematical techniques for analyzing stresses encountered in the car crash by modeling the physical structure into substructures called "finite elements." The finite elements and their stress interrelationships are converted into mathematical equations that are solved using computational techniques. In general, the crush zones are designed to absorb most of the impact energy. The pulses produced by FEA are filtered by a three-stage 4-kHz, low-pass digital filter.

[1]These locations are physically identified on a vehicle's body or chassis and labeled as such based on the extent to which a specific zone is crushed or not crushed in the event of a typical crash.

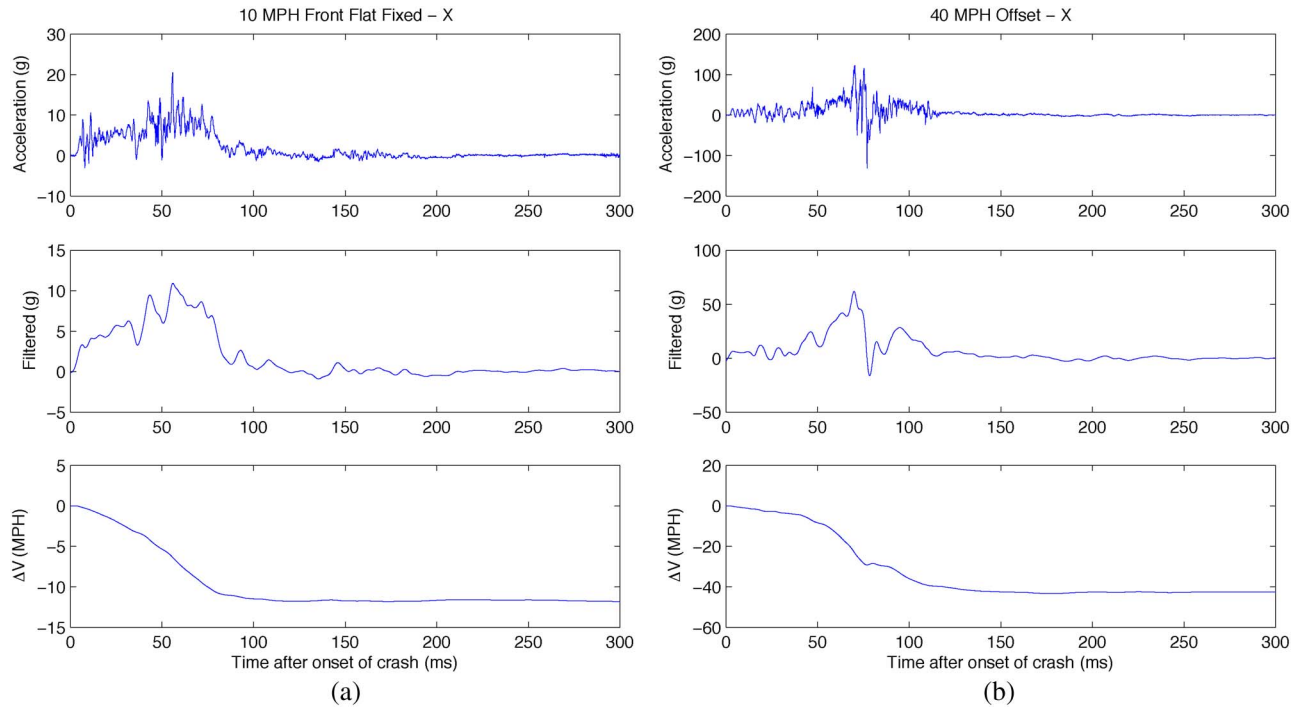


Fig. 2. Crash pulse characteristics. (a) "Nondeploy" 14-mi/h crash pulse after filtering and its Δv profile. (b) "Deploy" 40-mi/h crash pulse after filtering and its Δv profile. Note that Δv at the conclusion of the crash event is equal to the speed of the vehicle at the time of the crash.

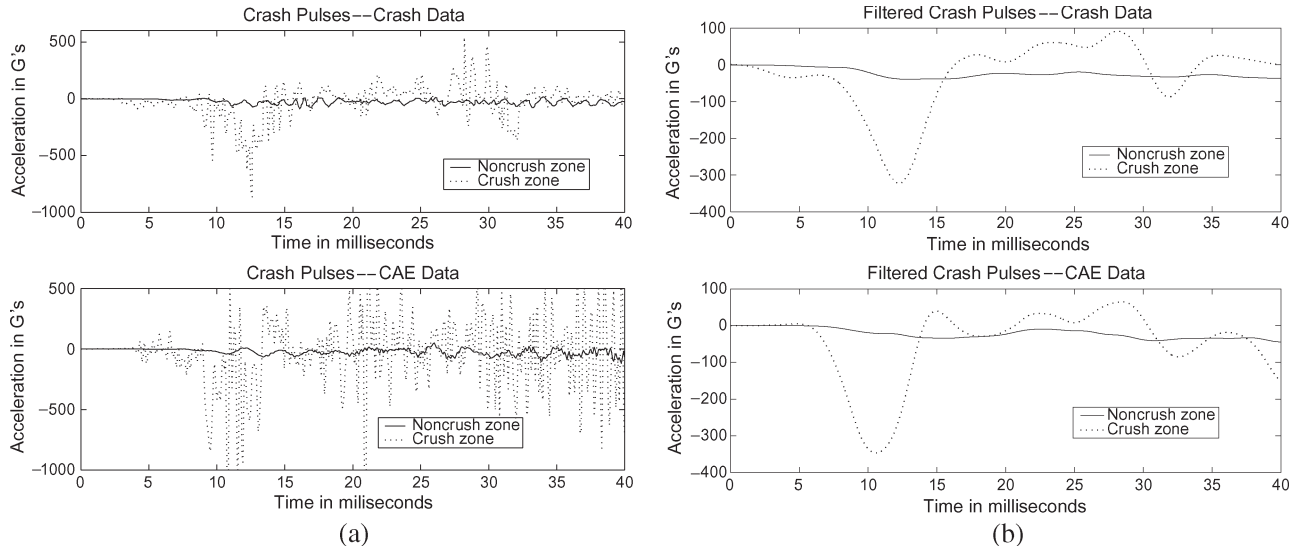


Fig. 3. Actual and filtered crash pulses. (a) Before filtering. (b) After filtering with a 4-kHz low-pass filter. Although the original crash pulses look distinct, filtered signals show substantial correspondence.

Fig. 3(a) and (b) depicts the waveforms before and after filtering for both real and CAE-generated pulses, respectively. A Δv analysis ($\Delta v = \int a\,dt$) and the power spectrum for the two types of pulses are shown in Fig. 4(a) and (b), respectively.

The comparison of the power spectrum indicates that the CAE-generated pulses exhibit good correspondence to the actual pulses. This is also evident in the filtered signal and Δv analysis.

For training our HMM, we utilize accelerometer data obtained from sensors in both the "crush" and "noncrush" zones. Since all crashes in the library are either head-on or slightly oblique, only the data along X-axes (longitudinal/forward) are used for training our HMM. The positive direction of the X-axes is from rear to front of the vehicle. Overall, the crash pulse library is composed of 21 actual crash pulses and 15 crash pulses generated by simulation of CAE models.

III. HMM of Vehicular Crash Pulses

HMMs are a class of models for representing the probability distribution of an observed time series. They are essentially stochastic finite state machines that output a symbol or a real value each time they leave a state. By specifying the state transition probabilities between states and the output generation

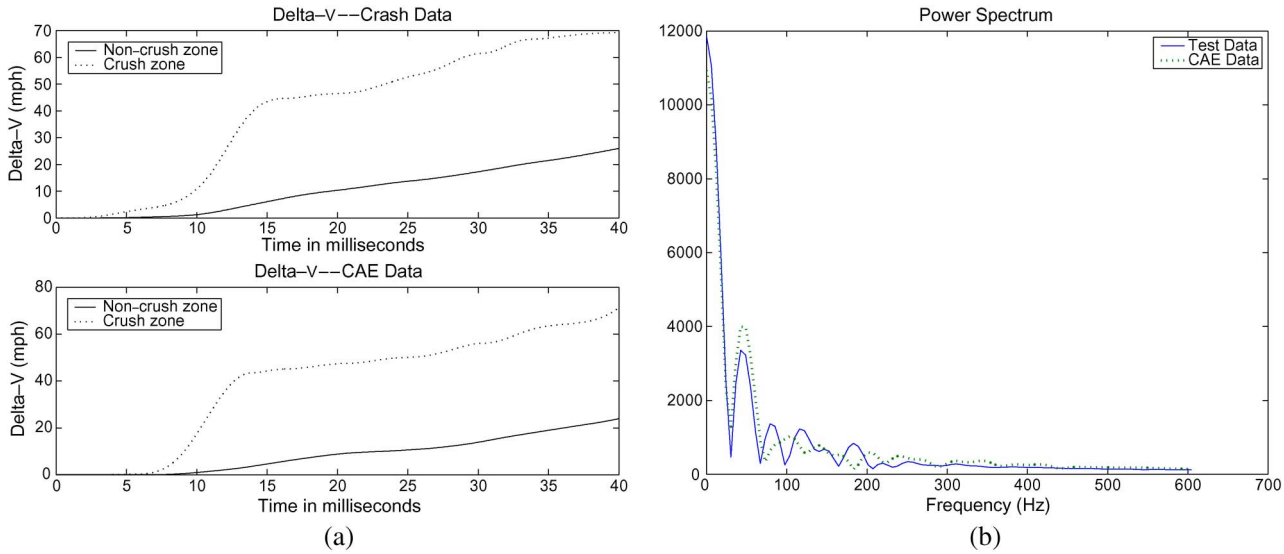


Fig. 4. Δv analysis and power spectrum of real and CAE-generated crash pulses. Again, the real and simulated pulses show close correspondence.

probabilities for each state, we can attempt to capture the underlying structure of the system that generates the output.

A. Multivariate Normality Test

The assumption that signals belong to a single Gaussian distribution must be validated as a prerequisite for using an HMM for representing crash pulses. The method for assessing multivariate normality utilizes the Mahalanobis distance and is obtained by plotting this distance against chi-square percentiles [25] in a quantile–quantile (Q–Q) plot. The Mahalanobis distance is utilized for determining the similarity of an unknown sample set to a known set [14], [16]. This distance metric takes into account the correlations of the data set and is scale invariant.

A Q–Q plot is a comparison of two distributions, either or both of which may be empirical or theoretical. When we compare the probability of distributions of two random variables, the Q–Q plot will result in a straight line if the two variables come from a common distribution. The advantages of the Q–Q plot are that the sample sizes do not need to be equal and that many distribution aspects, such as shifts in location and changes in symmetry, can simultaneously be tested. The Q–Q plot is particularly good for discriminating in the tail areas of the distributions since quantiles more rapidly change in those ranges, in turn requiring coverage between consecutive quantiles.

Assume that there are n independent observations $X_1, \ldots, X_n$, with each observation $X_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$ being a p-dimensional random vector sampled from a multivariate normal distribution with mean vector μ and covariance matrix $\mathbf{\Sigma}$. The squared Mahalanobis distance between X_i and μ is then given by

$$D_i^2 = (X_i - \mu)^T \mathbf{\Sigma}^{-1} (X_i - \mu). \quad (1)$$

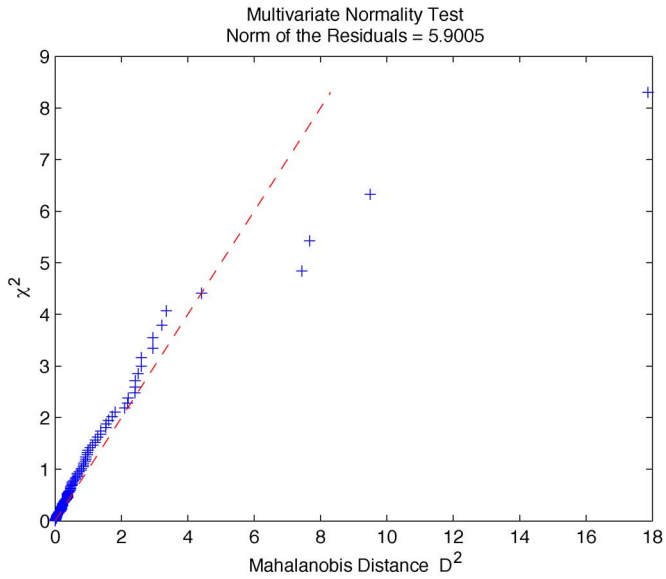
When both n and $(n-p)$ are greater than 25, the D_i^2 are distributed as independent chi-squared variate with p degrees of freedom df [26]. Because this is the case for the crash pulse data, we construct a chi-square plot for each sample, as shown in the list that follows.

1) D_i^2 are ordered from smallest to largest as $D_{(1)}^2$, $D_{(2)}^2, \ldots, D_{(n)}^2$.
2) The pair $(D_{(j)}^2, \chi_p^2((j-0.5)/n))$ is plotted, where χ^2 is the $100(j-0.5)/n$ percentile of the chi-square distribution with $df = p$ degrees of freedom.

Fig. 5(a) is one of the Q–Q plots for a crash pulse. The plot for each group produces a straight line with a norm of residual of 5.9 and, thus, conclusively establishes that the crash pulses approximately follow a multivariate normal distribution. The *norm of residual*, which is the largest singular value of the residual, can be used as a measure of the goodness of a linear regression. Fig. 5(b) shows a table of the norm of residuals for 21 real crash pulses, 15 of which are representative of crash events requiring airbag deployment and six being nondeploy crash events. Similar results were also obtained with CAE pulses and, thus, also established their correspondence with a multivariate normal distribution.

Crash detection is a real-time process, and thus, a simple model is preferred to enable us in making crash classification within hard real-time deadlines. The fully connected and left–right HMMs are the most commonly used models. A left–right HMM has only forward transitions, implying that the state transitions may never occur that take the HMM back to a state that has left in the past. The left–right HMMs have successfully been used in many time series applications. The fully connected HMMs are not subject to any such restriction, and it is possible for the HMM to transition from any state to another state during any time epoch.

As will be discussed in Section III-D, we use seven states to model the acceleration readings from the onset of the crash until 6 ms after the crash. The seven states represent the movement of the vehicle within 6 ms after the crash, i.e., moving forward, bouncing backward, moving forward again, bouncing backward again, and so on. At any time index t, it can either move forward to the next state or stay at the current state. A comprehensive



(a)

Pulse ID	*Norm(R)*	*pulse ID*	*Norm(R)*
1	3.8968	2	3.2983
3	3.1000	4	2.5038
5	7.6292	6	3.0973
7	2.8959	8	2.6238
9	5.9005	10	3.8160
11	4.2835	12	2.8139
13	3.4304	14	4.5001
15	3.4239	16	4.3512
17	2.6295	18	5.9986
19	8.3172	20	2.2502
21	3.0081		

(b)

Fig. 5. (a) Test for multivariate normality. (b) Norm of residuals for linear regression of the Q–Q plot.

review of HMMs is provided in [10]. In this section, we will briefly review the learning evaluation of HMMs with real-valued emission.

B. Components in an HMM

In this paper, the sequence of observations is denoted as $O = o_1, o_2, \ldots, o_T$. The corresponding hidden state sequence is denoted as $Q = q_1, q_2, \ldots, q_T$. Supposing that the observation has p dimensions, an HMM with N states has four components.

1) Initial state distribution $\Pi = \{\pi_i\}_{(N\times 1)}$, which is an $N \times 1$ vector. π_i is the probability of the event that the first hidden state is S_i. It can be written as $\pi_i = P(q_1 = S_i)$.
2) Transition matrix $A = \{a_{ij}\}_{(N\times N)}$, which is an $(N \times N)$ matrix. a_{ij} is given by $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, where q_t is the hidden state at time t, and S_i is the ith state.
3) Mean matrix $\mu = \{\mu_i\}_{p\times N}$, which is a $(p \times N)$ matrix. μ_i is the mean value of the ith state.
4) Covariance matrix $\Sigma_{(p\times p)}$, which is a $(p \times p)$ matrix.

In this paper, we denote an HMM as $\lambda = \{\Pi, A, \mu, \Sigma\}$.

C. Training HMMs With Real-Valued Emission

The Baum–Welch algorithm is used to train an HMM. The Baum–Welch algorithm can be divided into two steps, i.e., the E-step, where the likelihood $P(O|\lambda)$ is calculated, and the M-step, where the likelihood by updating the model λ is maximized.

1) E-Step: The likelihood of a crash pulse $P(O|\lambda)$ is defined as follows:

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda)P(Q|\lambda) = \sum_{\text{all } Q} \pi_{q_1} b_{q_1}(o_1), a_{q_1 q_2} b_{q_2}(o_2), \ldots, a_{q_{T-1} q_T} b_{q_T}(o_T)$$

where $b_j(o_t)$ is the probability of observation at time t given the hidden state is S_j, i.e., $b_j(o_t) = P(o_t|q_t = S_j)$, and Σ_j is the covariance matrix for state S_j, i.e.,

$$\Sigma_j = \frac{\sum_t P(q_t = S_j)(O_t - \mu_j)\prime(O_t - \mu_j)}{\sum_t P(q_t = S_j)}. \tag{2}$$

For a multivariate normal distribution, the emission probability can be written as

$$b_j(x) = \frac{1}{(2\pi)^{\frac{p}{2}}\sqrt{|\Sigma_j|}} \exp\left(-\frac{(x-\mu_j)' * \Sigma_j^{-1} * (x-\mu_j)}{2}\right). \tag{3}$$

The expectation probability is recursively computed using the following variables.

1) $\alpha_t(i)$: The forward variable is defined as the probability that the state after t observations is S_i, i.e.,

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = S_i|\lambda) \tag{4}$$

and can recursively be calculated using

$$\alpha_{t+1}(j) = \begin{cases} \pi_j b_j(o_1), & t = 1; \quad 1 \le j \le N \\ \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}), & 1 \le t \le T-1; \\ & 1 \le j \le N. \end{cases} \tag{5}$$

2) $\beta_t(i)$: The backward variable is defined as the probability that the sequence of observations from $(t+1)$ to T is observed after the system has been in state S_i at t, i.e.,

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T|q_t = S_i, \lambda). \tag{6}$$

Moreover, it can recursively be calculated using

$$\beta_t(i) = \begin{cases} 1, & t = T; \quad 1 \le i \le N \\ \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), & T-1 \ge t \ge 1; \\ & 1 \le i \le N. \end{cases} \tag{7}$$

Evaluation of an HMM is much simpler than training an HMM. To calculate the likelihood $P(O|\lambda)$ by substituting (4) into (2), we have

$$\begin{aligned} P(O|\lambda) &= \sum_{\text{all } Q} P(O|Q,\lambda)P(Q|\lambda) \\ &= \sum \pi_{q_1} b_{q_1}(O_1), a_{q_1 q_2} b_{q_2}(O_2), \ldots, a_{q_{T-1} q_T} b_{q_T}(O_T) \\ &= \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i). \end{aligned} \tag{8}$$

2) M-Step: In the M-Step, we update the model $\lambda = \{\Pi, A, \mu, \Sigma\}$ in such a way that Baum *et al.* have proved would guarantee the increase of likelihood, i.e.,

$$\overline{\pi}_j = \gamma_1(j), \qquad j = 1, \ldots, N \tag{9}$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \qquad \begin{matrix} i = 1, \ldots, N \\ j = 1, \ldots, N \end{matrix} \tag{10}$$

$$\overline{\mu_j} = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot o_t}{\sum_{t=1}^{T} \gamma_t(j)}, \qquad j = 1, \ldots, N \tag{11}$$

$$\overline{\Sigma_j} = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot (O_t - \mu_j)\prime(O_t - \mu_j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{12}$$

where $\xi_t(i,j)$ is the probability of being in state S_i at time t and state S_j at time $t+1$, given the model and the observation sequence, i.e.,

$$\begin{aligned} \xi_t(i,j) &= P(q_t = S_I, q_{t+1} = S_j | O, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \end{aligned} \tag{13}$$

and $\gamma_t(i)$ is the probability of being in state S_i at time t, given the observation sequence O and the model λ, i.e.,

$$\begin{aligned} \gamma_t(i) &= P(q_t = S_i | O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} = \sum_{j=1}^{N} \xi_t(i,j). \end{aligned} \tag{14}$$

D. Inducing the Optimal Crash Pulse Model

The process of crash detection is based on accumulation of readings sampled from the onboard acceleration sensors and stored in a buffer. Since crash sensing is a real-time application, simpler algorithms are preferred if the detection accuracy requirements can be satisfied. We have developed experiments to optimize the HMM topology, the number of states, and the size of the buffer used to store the past accelerometer readings.

Consistent estimators based on real crash data are used to optimize HMM parameters. Ziv and Merhav proposed an estimator of the number of states of an HMM based on entropy and data compression [27]. Liu and Narayan proposed an estimator by the method of mixtures [28]. Here, we use an estimator based on the likelihood of crash pulses. The rationale for using this estimator is that the likelihood is a by-product of HMM parameter estimation, which will ultimately be used to discriminate crash pulses from noncrash pulses. From the statistical learning theory, the likelihood can be written as $P(O|\lambda)$, where O is the observation (data) under the model λ. The likelihood is defined as the probability of the data (i.e., observation), given the underlying model. The probability $P(O|\lambda)$ can be calculated using (8).

Inductive modeling procedures typically begin by dividing the available data into two sets, namely, a training set and a testing/evaluation set. The general rule of thumb is to use two thirds of the data for training and leave the remaining one third for evaluation and follow the procedure outlined here to induct and validate the model.

1) Divide the available data into a training set and a testing/evaluation set.
2) Split the training data set into n folds; each fold contains approximately the same number of crash pulses.
3) Set the parameters for the model being validated (in our case, this is the HMM topology, number of states, and accelerometer buffer size).
4) Use $(n-1)$ folds to train a model and the remaining folds for testing.
5) Repeat steps 2–4 for each of the architectures—this is analogous to searching the architecture space for an optimal model.
6) Select the best model and train it using the entire training set.
7) Benchmark the final model using the evaluation set. These are the data that the model has never seen before, which provide a more accurate estimate of the performance on the testing/evaluation data selected in step 1.

Steps 2–4 are devised to establish the optimal HMM configuration. This technique is also referred to as the *leave-one-out* strategy. This strategy is applied by keeping the topology fixed and optimizing the detection system architecture with different parameter settings for a number of states and pulse storage buffer sizes. In the same manner, the HMM topology is varied to establish the optimal number of states and interconnectivity (fully connected or left–right connected). Subsequently, after selecting the best topology and training parameters, all the training data are utilized to train the final model.

The process of establishing and validating the crash model in our study is somewhat modified to facilitate learning a comprehensive set of crash characteristics from a complete set of crash pulses. Accordingly, two optimal HMMs are constructed. In the first instance, an optimal HMM is induced using the entire set of "actual" crash pulses with the entire set of "simulated" pulses serving as the testing set. In the second instance, the entire set of "simulated" pulses is used for training with the "actual" crash pulses serving as the testing set. Thus, two optimal crash HMMs, i.e., one with real pulses and the other with CAE pulses, are induced and validated using the other data set.

Due to the lack of information on the noncrash events, our training process is geared toward training crash HMMs only;

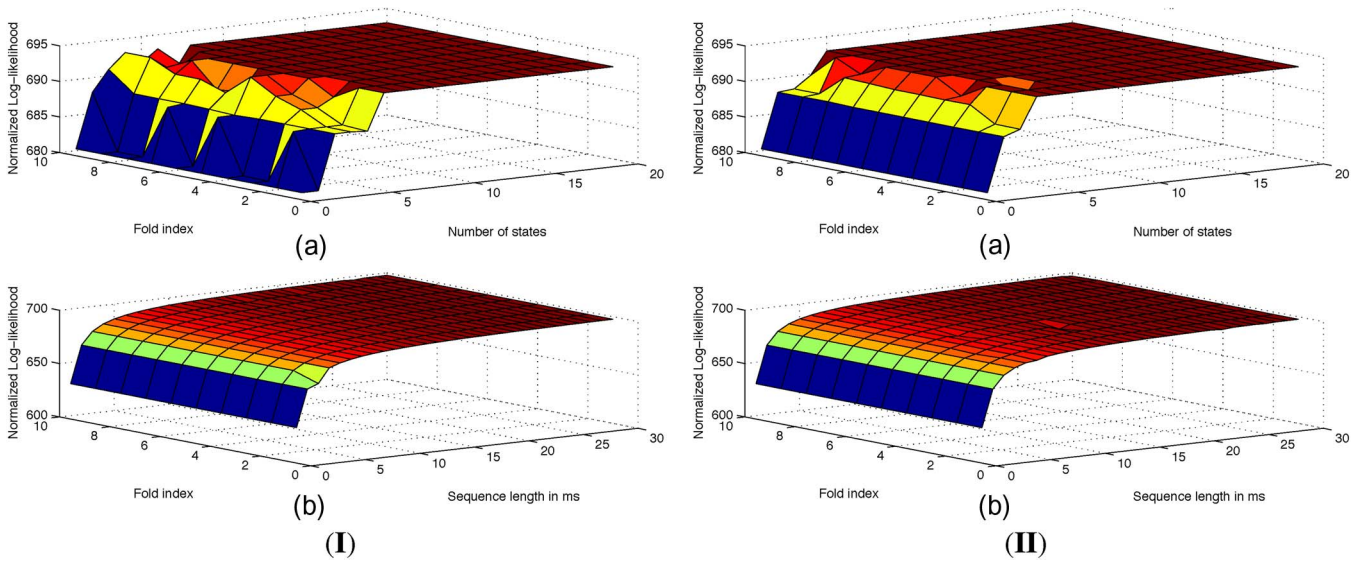


Fig. 6. "Leave-one-out" cross validation for (I) fully connected HMMs and (II) left–right HMMs. The Z-axis represents the normalized log likelihood of the ten pulses, and the Y-axis represents the fold index. (a) X-axis represents the number of states from 1 to 20. (b) X-axis represents the sequence length in milliseconds.

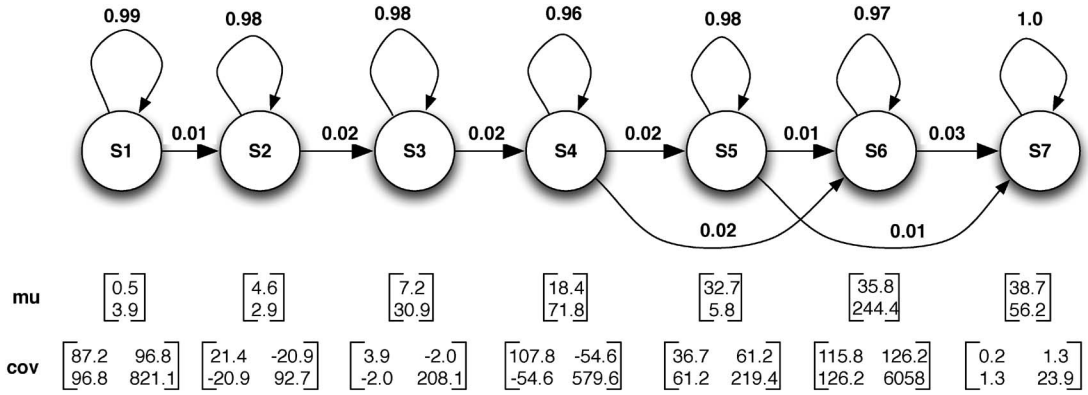


Fig. 7. Seven-state left–right HMM is used to model crash events.

we do not train any noncrash HMMs, as would sometimes be useful when measuring the log likelihood of a crash versus a noncrash. Thus, although the entire crash pulse library is composed of both the real and simulation data, the use of one type of data for induction in one case ensures that all representatives pulses in a set are utilized in the computation of model parameters as both sets are generally developed to be comprehensive to encompass all major crash events.

Fig. 6(I) shows the $LL(N, F)$ and $LL(L, F)$ for crash pulses for the fully connected HMMs, where LL is the normalized log likelihood, N is the number of states of the HMM, F is the fold index, and L is the pulse length considered. The likelihood is normalized with respect to a random sequence of the same length. Thus, we can conclude that when $N \geq 7$ and $L > 5$ ms, all the folds show great consistency for a fully connected model.

Fig. 6(II) shows the $LL(N, F)$ and $LL(L, F)$ of crash pulses for the left–right HMMs. From this, we conclude that an increase beyond $N \geq 7$ and $L > 5$ ms (an increase in the number of states to more than seven or the buffer size with storage larger than 5 ms) does not result in further performance improvements. Furthermore, Fig. 6 demonstrates that with a sufficient number of states and sequence length, all training pulses yield similar $LL(N, L)$ characteristics. This further validates our earlier observations that HMMs are noise tolerant.

For models with similar likelihoods, the simplest model is preferred. Thus, a seven-state left–right HMM with a 6-ms buffer size is chosen to implement a crash-detection system. This induced HMM is shown in Fig. 7. The $\mu = [\mu_{\text{noncrush}}, \mu_{\text{crush}}]^T$ vector associated with each state is the average acceleration for the *noncrush* and *crush* zones. Classification of crash pulses using this model is presented next.

IV. Results

Left–right HMMs with seven states are trained by two sets of crash pulses. These two sets are given as follows: 1) crash pulses captured through actual accelerometer measurements in a crash or 2) the set of pulses generated by the simulation of CAE models. These two sets are independently treated. When the crash pulse model is induced using the set of pulses from actual crash data, model validation is performed using the set of CAE pulses. Similarly, when the crash pulse model is induced using the CAE pulses, model validation is performed using actual crash pulses.

TABLE I
CLASSIFICATION OF "REAL" CRASH PULSES USING AN HMM INDUCED FROM CAE-GENERATED SIMULATED CRASH PULSE DATA

Pulse Type	*Real Crash*	*Pulse Tested by HMM-CAE*	*Detection Time (ms)*
Real	Yes	35 mph flat frontal rigid barrier	1.0
Real	Yes	35 mph flat frontal rigid barrier	2.8
Real	Yes	35 mph flat frontal fixed barrier	2.8
Real	Yes	35 mph 90° frontal fixed barrier	2.2
Real	Yes	35 mph 90° frontal fixed barrier	1.0
Real	Yes	35 mph 90° frontal fixed barrier	1.0
Real	Yes	30 mph 30° frontal oblique fixed barrier	3.4
Real	Yes	30 mph 30° frontal oblique fixed barrier	2.8
Real	Yes	30 mph 30° left oblique fixed barrier	2.2
Real	Yes	40 mph offset deformable barrier	1.6
Real	Yes	35 mph flat frontal fixed barrier	1.0
Real	Yes	22 mph 30° oblique frontal car to car	1.0
Real	Yes	42 mph 30° frontal oblique fixed barrier	3.4
Real	Yes	24 mph car to car	1.6
Real	No	90° frontal center pole	Non-crash
Real	No	frontal offset fixed barrier	Non-crash
Real	No	flat frontal fixed barrier	Non-crash
Real	No	90° frontal fixed barrier	Non-crash
Real	No	undercarriage	Non-crash
Real	No	90° frontal offset fixed barrier	Non-crash

In addition to the crash pulses used for training that require airbag deployment, crash pulses that do not need airbag deployment are included in each set. Six actual crash pulses not requiring bag deployment, and three CAE nondeployment pulses are added to each set, respectively. *Detection time* is defined as the duration of time that elapses between the point of impact and the time that the system can classify the acceleration pulse as a being a *crash*. Again, the collision time is set to $t = 0$.

These results are shown in Table I. All CAE-generated crash and noncrash pulses are correctly classified by the HMMs within 6 ms.

Similarly, when the crash pulse model is induced using the set of pulses from CAE simulations of crashes, model validation is performed with the actual crash pulses. These results are shown in Table II. All crash and noncrash pulses are correctly classified by the HMMs within 6 ms. It might be noted that in some cases, classification is correctly achieved in 1 ms. This corresponds to making a decision based on about 12–13 samples corresponding to the sample interval of 0.08 ms. However, the decision is based on the entire 6-ms buffer of which is filled with near-zero acceleration values corresponding to the constant speed of the vehicle prior to the crash. The HMM learns to recognize this transition and correctly classifies the pulse correspondingly.

Results shown in Tables I and II demonstrate that HMM-based models are quite effective in early and accurate classification of crash pulses. Furthermore, the HMM trained by CAE pulses outperforms the HMM trained by true pulses when the average detection time is used as a measure of performance. Finally, for a given pulse, regardless of whether it is a true pulse or a pulse generated from CAE models, there were no false negatives.

TABLE II
CLASSIFICATION OF CAE-GENERATED SIMULATED CRASH PULSES USING AN HMM INDUCED FROM "REAL" CRASH PULSE DATA

Pulse Type	*Real Crash*	*Pulse Tested by HMM-Real*	*Detection Time (ms)*
CAE	Yes	20 mph frontal	5.2
CAE	Yes	23 mph frontal	4.6
CAE	Yes	25 mph frontal	4.6
CAE	Yes	28 mph frontal	4.6
CAE	Yes	22 mph angle	4.6
CAE	Yes	22 mph angle	4.0
CAE	Yes	35 mph frontal	2.2
CAE	Yes	93 mph frontal	6.4
CAE	Yes	24 mph vehicle to vehicle	5.2
CAE	Yes	46 mph vehicle to vehicle	4.6
CAE	No	Frontal	Non-crash
CAE	No	Frontal	Non-crash
CAE	No	Pole	Non-crash

V. CONCLUSION

Crash data were modeled as a 3-D time series composed of a time stamp, an acceleration reading from the tunnel (noncrush zone), and an acceleration reading from the crush zone. This paper has presented a methodology for building a crash detection system using continuous-mode HMMs and has established the proposed strategy to be a robust methodology for early detection of automotive crashes. A data-validating strategy for establishing that the crash pulse train is a sample from a multivariate normal distribution was presented. The process of optimizing a real-valued HMM parameter was also presented; a left–right HMM with seven states and a sample buffer with 6-ms crash pulse storage was shown to be a simple, yet robust, architecture for automotive crash detection. In our experiment, no incidence of false-positive or false-negative detection was observed, and thus, 100% accuracy was reported for all the pulses. It was also demonstrated that CAE pulses from FEA can provide valuable information in crash evaluation. Although the number of cases examined was limited and not adequate for generalization, CAE pulses may be more suitable for training crash pulse models with the actual crash pulses used to evaluate the model's performance.

ACKNOWLEDGMENT

The authors would like to thank Dr. C. C. Chou from Ford Motor Company for the many valuable discussions.

REFERENCES

[1] L. Evans, "Airbag benefits, airbag costs," presented at the Soc. Automotive Eng. World Congr. Exhib., Air Bags Belt Restraints, Detroit, MI, 2004, SAE Paper 2004-01-0840.

[2] C.-Y. Chan, "On the detection of vehicular crashes—System characteristics and architecture," *IEEE Trans. Veh. Technol.*, vol. 51, no. 1, pp. 180–193, Jan. 2002.

[3] R. Barnard and M. Riesner, "Vehicular air-bag control based on energy, momentum, and semimetric spaces," *IEEE Trans. Veh. Technol.*, vol. 49, no. 5, pp. 1641–1649, Sep. 2000.

[4] G. Singh and H. Song, "Intelligent algorithms for early detection of automotive crashes," presented at the Soc. Automotive Eng. World Congr. Exhib., Air Bags (Part A and B), Detroit, MI, 2002, SAE Paper 2002-01-0190.

[5] W. Yin, J. Zhang, and S. Huang, "Algorithm for airbag actuation based on artificial neural network," *J. Tsinghua Univ.*, vol. 39, no. 11, pp. 73–75, 1999.
[6] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden Markov model: Analysis and applications," *Mach. Learn.*, vol. 32, no. 1, pp. 41–62, Jul. 1998.
[7] T. Ivanova, V. Mottle, and I. Muchnik, "Estimating the parameters of hidden Markov models of noise like signals with abruptly changing probabilistic properties," *Autom. Remote Control*, vol. 55, no. 9, p. 1299, 1994.
[8] J. Yang, Y. Xu, and C. Chen, "Human action learning via hidden Markov model," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 27, no. 1, pp. 34–44, Jan. 1997.
[9] J. Yang, Y. Xu, and C. Chen, "Hidden Markov model approach to skill learning and its application to telerobotics," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 621–631, Oct. 1994.
[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[11] M. Brown, J. Foote, G. Jone, K. Spärck Jones, and S. Young, "Open-vocabulary speech indexing for voice and video mail retrieval," in *Proc. ACM-Multimedia Conf.*, 1996, pp. 307–316.
[12] E. Mittendorf and P. Schauble, "Document and passage retrieval based on hidden Markov models," in *Proc. 17th ACM-SIGIR Int. Conf. Inf. Retrieval*, 1994, pp. 318–327.
[13] S. Oh, J. Ha, and J. Kim, "Context dependent search in interconnected hidden Markov model for unconstrained handwriting recognition," *Pattern Recognit.*, vol. 28, no. 11, pp. 1693–1704, Nov. 1995.
[14] R. Khattree and D. N. Naik, *Applied Multivariate Statistics With SAS Software*. Cary, NC: SAS Inst. Inc., 1995, pp. 9–12.
[15] J. Chen and A. Kundu, "Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden Markov model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 208–214, Feb. 1994.
[16] V. Krishnamurthy, "Hidden Markov model signal processing in presence of unknown deterministic interferences," *IEEE Trans. Autom. Control*, vol. 38, no. 1, pp. 146–152, Jan. 1993.
[17] A. Krogh, M. Brown, S. Mian, K. Sjolander, and D. Haussler, "Hidden Markov models in computational biology—Applications to protein modeling," *J. Mol. Biol.*, vol. 235, no. 5, pp. 1501–1531, Feb. 1994.
[18] A. Krogh, M. Brown, S. Mian, and D. Haussler, "A hidden Markov model that finds genes in *E. coli* DNA," *Nucleic Acids Res.*, vol. 22, pp. 4768–4778, 1994.
[19] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure, "Hidden Markov models of biological primary sequence information," *Proc. Nat. Acad. Sci. USA*, vol. 91, no. 3, pp. 1059–1063, Feb. 1994.
[20] J. Bilmes, "What HMMs can do," Univ. Washington, Seattle, WA, Tech. Rep. UWEETR-2002-0003, Jan. 2003.
[21] U. Glavitsch and P. Schauble, "A system for retrieving speech documents," in *Proc. ACM-SIGIR 15th. Int. Conf. Inf. Retrieval*, 1992, pp. 168–176.
[22] M. Huang, *Vehicle Crash Mechanics*, 1st ed. Boca Raton, FL: CRC, Jun. 19, 2002, ch. 1.
[23] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, no. 26, pp. 314–319, Sep. 1985.
[24] F.-J. Stützler, C. C. Chou, J. Le, and P. Chen, "Development of CAE-based crash sensing algorithm and system calibration," presented at the Soc. Automotive Eng. World Congr. Exhib., Airbags Safety Test Methodology, Detroit, MI, 2003, SAE Paper 2004-01-0509.
[25] J. Stevens, *Applied Multivariate Statistics for the Social Sciences*, 2nd ed. Mahwah, NJ: Lawrence Erlbaum, 1992, pp. 245–251.
[26] M. S. Srivastava and E. M. Carter, *An Introduction to Applied Multivariate Statistics*. New York: Elsevier, 1983, pp. 25–35.
[27] J. Ziv and N. Merhav, "Estimating the number of states of a finite-state source," *IEEE Trans. Inf. Theory*, vol. 38, no. 1, pp. 61–65, Jan. 1992.
[28] C.-C. Liu and P. Narayan, "Order estimation and sequential universal data compression of a hidden Markov source by the method of mixtures," *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 1167–1180, Jul. 1994.

Gautam B. Singh (SM'07) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1983 and the Ph.D. degree in computer engineering from Wayne State University, Detroit, MI, in 1993.

He is currently an Associate Professor of computer science and engineering with Oakland University, Rochester, MI. Prior to joining Oakland University, he was a Senior Scientist with the National Center for Genome Resource, Santa Fe, NM, and held a faulty position with Wayne State University. He conducts research in machine learning and data mining and consults with Chrysler Corporation.

Dr. Singh is a member of Tau Beta Pi and the Association for Computing Machinery.

Haiping Song received the B.S., M.S., and Ph.D. degrees in electrical engineering from East China Normal University, Shanghai, China, in 1992, 1995, and 1998, respectively, and the M.S. degree in computer science from Oakland University, Rochester, MI, in 2002. She is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, Oakland University.

Her research interests include using machine learning and data-mining techniques and applications of intelligent systems technology to vehicle safety and in airbag system design.