FUSION OF BRAKE PEDAL SENSORS IN BY-WIRE CARS: A FUZZY LOGIC APPROACH

Reza HoseinNezhad [(1)] **Alireza Bab-Hadiashar** [(1)] **Peter Harding** [(2)]

[(1)] *School of Eng. and Science, Swinburne Univ. of Technology, Victoria, Australia*
{rhoseinnezhad , abab-hadiashar}@swin.edu.au

[(2)] *Control Engineer, Pacifica Group Technologies, Victoria, Australia*
peter_harding@pgt.com.au

Abstract - By-wire braking is one of the important parts in a by-wire car from safety point of view. Because of safety critical issues, brake pedal is equipped with multiple sensors and their data must be properly fused. In addition to accuracy and robustness, mathematical complexity and computational load of the fusion method are also considered in this application. In this paper, a soft voting method is presented that uses fuzzy rule-base decision making. The fusion output is a weighted average of three sensory signals where the weights are calculated from a set of fuzzy rules. While there are only some piece-wise linear calculations in the proposed method, our experimental results show that the performance of the proposed voting method is desirable in presence of short circuits to ground or supply voltage, considerable SNR and sensor drifts. Voting error (in terms of MSE) is reduced by 82% by soft voting method, compared to simple majority (hard) voting.

Keywords: by-wire braking, voting, fuzzy rule-base, sensor data fusion.

1. INTRODUCTION

By-wire technology in automotive industry replaces the traditional mechanical and hydraulic systems with electronic control systems, electro-mechanical actuators and human-machine interfaces such as pedal and steering feel emulators. Hence the traditional components such as the steering column, intermediate shafts, pumps, hoses, fluids, belts and brake boosters and master cylinders are eliminated from the vehicle. Brakes are a safety critical system on the vehicle. Our by-wire brake pedal is shaped like a usual pedal but it does not transfer driver's brake effort through hydraulic systems. Fig. 1 shows the general structure of our brake pedal sensors and their connections to the local processors and brake actuators for the four wheels of a vehicle. Our pedal is equipped with three sensors that measure the brake demand (in terms of force and displacement) and transmit it as electric signals, via a communication network.

Two sensors measure the force and the third measures the pedal displacement. These values are filtered and sampled in a fault tolerant set of local processors and transferred to all four wheels via a fault tolerant communication bus. The sensor data are also sent to a central controller that generates high level braking commands, such as ABS (Anti-lock Braking System) commands.

In order to generate an estimation of driver's brake demand, sensory data should be fused both in the central controller and in wheel units. The result may

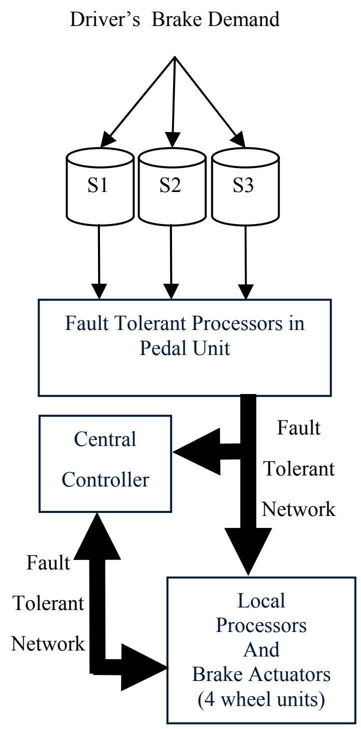

Fig. 1. General structure of our brake pedal sensors and their connections to wheel units and the central controller

be next fused with the data given by other sensors mounted on the vehicle (e.g. wheel speed sensors or temperature or tension sensors in brake calliper units) to generate final brake commands. These commands will be sent to callipers to activate brake actuators in wheel units. If for any reason the central controller is not working, then pedal sensor data will be fused in each of the wheel units independently and the result will cause an independent brake action on the wheel.

The main purpose of sensor fusion is to detect sensor faults (such as excessive noise, short circuits or sensor drifts) and to remove the effects of faulty sensor values from the brake demand. Multiple sensors require a voting operation to check their agreement before fusion.

Several well known voting algorithms have been widely used in commercial applications. The n-input *majority voter* produces a correct result where at least [(n+1)/2] voter inputs match each other. In cases of no majority, the voter generates an exception flag, which can be detected by the system supervisor to move the system toward a safe state (Avizienis, 1985). As an extended version of majority voter, *plurality voter* implements "m out of n" voting, where m is less than a strict majority (Bass, et al., 1997). *Median voter* is a mid-value selection algorithm. Assuming an odd number of redundant inputs, this algorithm successively eliminates pairs of outlying values until a single result remains (Brownrigg, 1984). The *weighted average voter*, on the other hand, calculates the weighted mean of its redundant input values (Tong and Kain, 1988). Parhami (1994) considered the performance of voters, in terms of execution time, and proposed efficient implementations of a variety of algorithms.

There is no agreement checking in weighted average and median voters (Latif-Shabgahi et al. 2003). Hence, they are not appropriate for safety critical applications such as braking. In the case of three inputs, majority and plurality voters are the same. Majority voters give *no result* in the output and instead a flag is set as explained before. In the by-wire braking system studied in this paper, "no result" is not acceptable as the output of brake pedal unit. Instead, three status bits are generated for each of the three pedal sensors. If the sensors do not agree, invalidity of the voter output will be deduced from the status bits. Another problem with a majority voter is its considerable output discontinuity in the event of long-time disagreements (Latif-Shabgahi et al. 2003).

Latif-Shabgahi et al. (2003) proposed to solve this problem by introducing a *smoothing voter* in which agreement checking threshold adaptively changes when the voter produces *no result.* While their proposed method results in a lower number of *no result* events in the output of the voter, such events are not completely prevented.

As an alternative solution for the problem, we propose to use the mean of agreeing sensors (if two or all the three sensors agree) as the output of a majority voter and use their median value if there is no agreement. In this method that we call *hard voting*, a status bit is set if the sensors agree and reset, if they don't. A problematic issue in this voting method is the *geometric distance threshold* value by which sensor agreement is checked. Due to sensor conversion errors, there is almost always a distance between two agreeing sensors of different types. The distance threshold should be large enough to prevent incorrect decisions about sensor agreements in presence of sensor conversion errors. A large value for distance threshold in hard voting method will result in *late fault detection* if the fault causes a gradual change in the sensory signal. Such faulty gradual changes in sensory signals usually happen because of drifts, short circuits[1] and sensor noise, which gradually increases with temperature.

In this paper we propose a new voting method, called *soft voting* (in contrast to its alternative, hard voting). By using fuzzy logic rule-base inference, the effect of a faulty sensor is gradually removed from the output of our proposed soft voter. Instead of status bits, a faultiness measure is defined for each sensor that gradually increases in the event of faults.

We will present our soft voting approach in Sec. 2. Then, comparative experimental results of hard and soft voting methods on real sensory data will be given in Sec. 3, followed by conclusive remarks in Sec. 4.

2. PROPOSED SOFT VOTING METHOD

Practically, if the data measured by all pedal sensors *agree* with each other then we simply calculate the driver's brake demand by averaging three sensor values. In presence of a fault or a substantial level of noise in sensor signals, they will not agree with each other. A voter detects these disagreements and uses them to identify faulty sensors and generate its output. A hard voter simply discards faulty sensor data and outputs the average of agreeing sensors.

Block diagram of pedal sensor fusion is shown in

Fig. 2. In this diagram, the two force sensors are called S_1 and S_2 and the displacement sensor is called S_3. Initially, a simple low-pass filtering (to reduce the noise) and missing data handling (in form of a predictive filter) are performed on the raw sensory data. The three output signals of S_1, S_2 and S_3 are called f_1, f_2 and x, respectively. The displacement signal x is then converted to two estimated force measurements by some experimental models. These models give a piece-wise quadratic expression of the force sensor signals in terms of the displacement sensor signals. Our measurements show that the piece-wise quadratic models give an average and maximum estimation errors of 3% and 20%, respectively. In *Agreement Evaluation* blocks, three distance metrics for three pairs (two force signals or a pair of force and converted position signals) are computed as below:

$$\alpha_{12} = \left| f_1 - f_2 \right|; \alpha_{23} = \left| f_2 - \hat{f}_2 \right|; \alpha_{31} = \left| \hat{f}_1 - f_1 \right| \quad (1)$$

Finally, the sensory data f_1 and f_2 and their estimated values from x and also the three agreement

[1] The RC filters that are connected to the inputs of ADCs (Analog to Digital Converters) cause a gradual change in sensory signals when a short circuit happens.

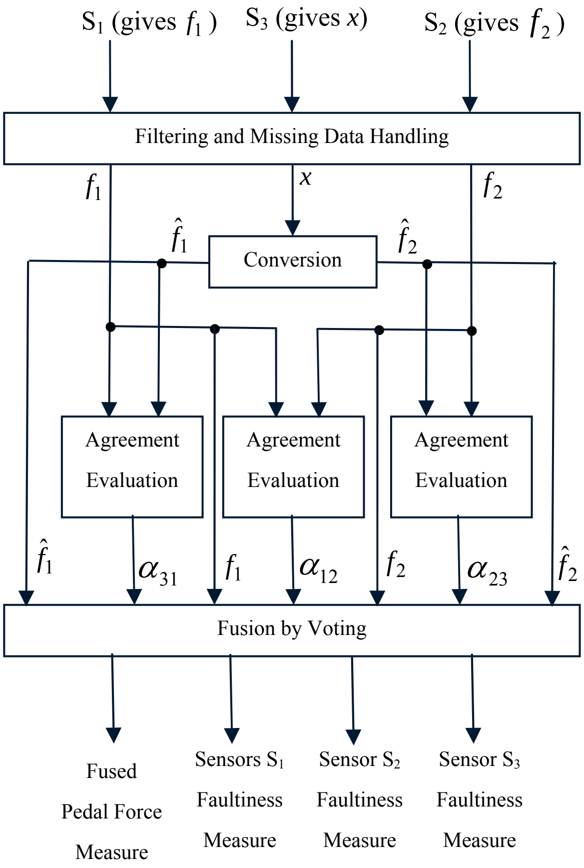

Fig. 2. Block diagram of pedal sensor fusion

evaluations α_{12} , α_{23} and α_{31} are passed on as inputs of a box that is responsible for fusion by voting. Soft voting output consists of a fused pedal force measurement and three *faultiness measures* (instead of status bits) for the three pedal sensors. Each faultiness measure is a quantitative evaluation of voter's belief in faultiness of a sensor in [0,1], with a value of 1 for total belief. A hard voter however, outputs a fused pedal force measurement and three status bits, showing the occurrence of a fault in the sensors.

Based on sensor agreements, following possible outputs are voted as the fused pedal force measure:

1. S_1 only: $O_1 = f_1$
2. S_2 only: $O_2 = f_2$
3. S_3 only: $O_3 = (\hat{f}_1 + \hat{f}_2)/2$
4. Average of S_1 and S_2: $O_4 = (f_1 + f_2)/2$
5. Average of S_2 and S_3: $O_5 = (f_2 + \hat{f}_2)/2$
6. Average of S_3 and S_1: $O_6 = (f_1 + \hat{f}_1)/2$
7. Average of all sensors: $O_7 = (f_1 + \hat{f}_1 + f_2 + \hat{f}_2)/4$

In a hard voting strategy, α_{ij} values are compared with some *agreement thresholds.* Based on the results of comparison, one of the above possible outputs is selected as the fused output and three status bits are set. For example if α_{12} and α_{31} are more than the agreement threshold (i.e. S_1 and S_2 do not agree with each other. So do the pair of S_1 and S_3) and α_{23} is less than the agreement threshold (i.e. S_2 and S_3 agree with each other), then the hard voter will deduce that S_1 is faulty. In this case, the fused output will be the average of S_2 and S_3 and the faultiness status bits will be 100 for S_1, S_2 and S_3 respectively.

The agreement thresholds are important in voting process. They depend on α_{ij} values in a normal working condition, when no sensor is faulty. We measured absolute difference values α_{ij} in normal condition. Because of modelling errors in conversion of position values to equivalent force values, α_{23} and α_{31} change widely. The agreement thresholds should be greater than α_{ij} maximum values in normal condition, so that conversion errors don't cause the voter to incorrectly assume that two sensors disagree. For example, agreement threshold for S_1 and S_3 or for S_2 and S_3 should be higher than the maximum modelling error of 20%. These large thresholds cause long delays for fault detection in hard voting process. Particularly, if a sensor gradually deviates from its true values because of sensor drifts or noise or short circuits, then the large thresholds cause a long delay in detection of the fault by checking sensor disagreements.

We propose a new *soft voting* approach to solve the problem of latency in fault detection and prevent large discontinuities in the fusion output. The proposed soft voter gives a weighted average of all possible outputs of the hard voter, as expressed below:

$$\text{Fused Output} = \sum_{i=1}^{7} w_i \times O_i \Big/ \sum_{i=1}^{7} w_i \qquad (2)$$

Voting is performed by smoothly changing w_i weights, managed by a fuzzy rule-base. Like any fuzzy inference engine, at the first step α_{ij} inputs are fuzzified. We define three fuzzy sets of **L**arge, **M**edium and **S**mall agreements by their membership functions. These definitions are based on empirical maximum values of α_{ij}, derived from our measurements and calculations. Fig. 3 shows our definition of the fuzzy sets for α_{12}. Since S_1 and S_2 give nearly same values, same definitions can be applied for fuzzification of their agreement with the third sensor i.e. α_{31} and α_{23}. This definition is shown in Fig. 4.

In these definitions, mathematical shapes of the membership functions are similar and more

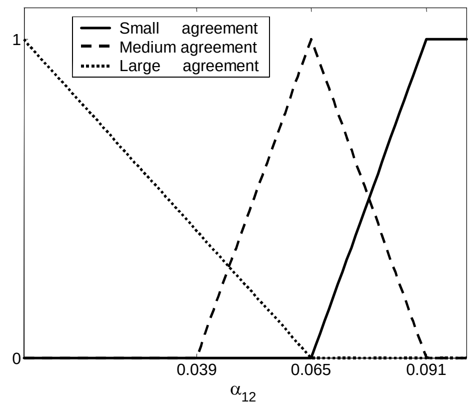

Fig. 3. Definition of three fuzzy sets for fuzzification of sensor agreements for S_1 and S_2

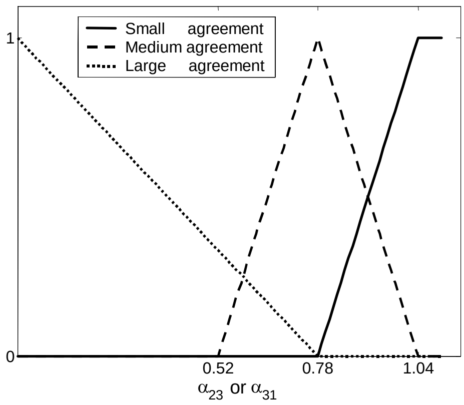

Fig. 4. Definition of three fuzzy sets for fuzzification of sensor agreements for S_1 and S_3 or for S_2 and S_3

importantly piece-wise linear and easy to compute. Breaking points of the functions in Fig. 3 and Fig. 4 are different because the range of variation of signals and their absolute differences (α_{ij} values) in normal condition are different. Furthermore, since a lower α_{ij} value means more agreement between S_i and S_j, the **L**arge fuzzy set is associated with lower α_{ij} values and vice versa. The resulting membership values are then used by a fuzzy rule-base for fuzzy inference.

Based on the same logic as in hard voting, the fuzzy rules determine the output (fused) value and faultiness of each sensor in different conditions of sensor agreement. For example one of the fuzzy rules is as follows:

IF

S_1 and S_2 agreement is **S**mall

AND S_2 and S_3 agreement is **L**arge

AND S_3 and S_1 agreement is **S**mall

THEN

The fused output is the average of S_2 and S_3

AND S_1 faultiness is **L**arge

AND S_2 faultiness is **S**mall

AND S_3 faultiness is **S**mall

This rule explains what is logically expected as voting result if S_1 does not agree with the other two sensors. Average of S_2 and S_3 (fusion output O_5) is the expected fusion output in this case and faultiness of S_1, S_2 and S_3 are **L**arge, **S**mall and **S**mall, respectively. The final fusion output is calculated as a weighted average of all possible expected outputs by Eq. (2). We calculate each weight ,w_i , as the sum of the products of membership values in the *antecedent* of every rule that determines O_i as the fusion output in its *consequent*. Thus, the resulting weights smoothly change from 0 to 1 (or 1 to 0) and the fused output is smoothly switched from one vote to the other, hence the name *soft voter*. Sensor faultiness values must also be defuzzified into real values. Definitions for the fuzzy sets, named Small, Medium and Large are depicted in Fig. 5. Faultiness measures are defuzzified into crisp outputs by *fuzzy centroid* method.

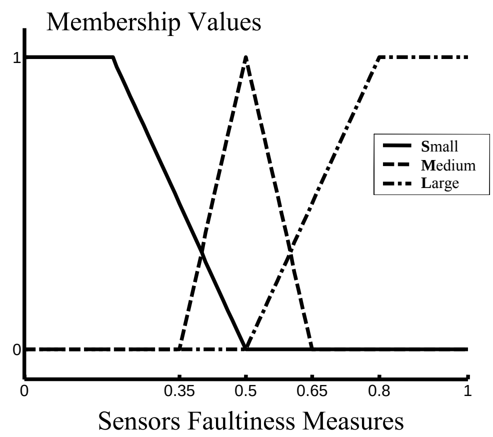

Fig. 5. Fuzzy sets definition for defuzzification of sensor faultiness measures

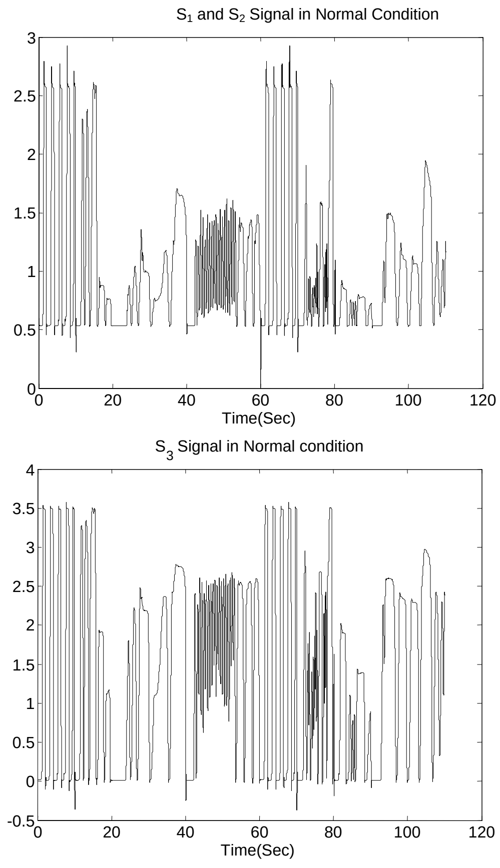

Fig. 6. Recorded sensory signals in normal (no fault) condition

3. EXPERIMENTAL RESULTS

We tried different kinds of brake efforts in different conditions such as a continuous hard brake, short-time hard brakes, short-time soft brakes, a continuous soft brake and so on. Total length of each experiment is 110 seconds. Fig. 6 shows the recorded signals of the three sensors. S_1 and S_2 signals(pedal force measurements) match each other and one of them is shown in Fig. 6. We injected several kinds of faults into S_1 during the time interval [80,110] and used both hard and soft voting methods to fuse the sensor data. Fig. 7 shows the results when S_1 signal is short-

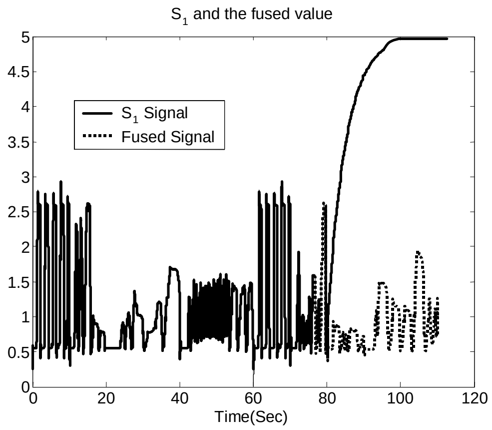

Fig. 7. Soft voting result when S_1 is short circuit and gradually rises toward supply voltage

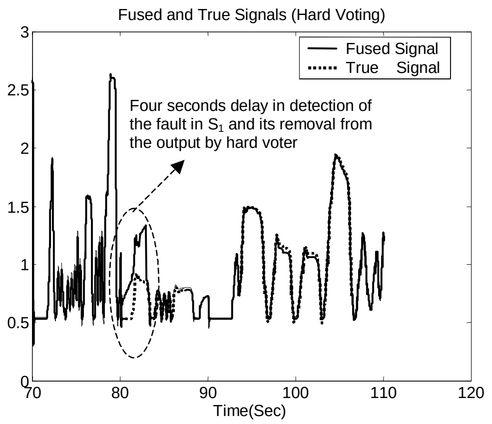

Fig. 8. Hard voting result when S_1 is short circuit and gradually moves toward supply voltage

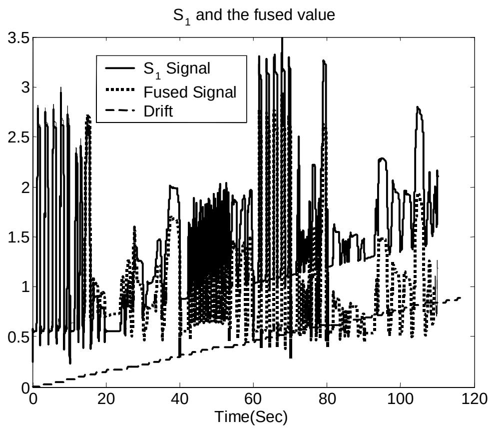

Fig. 9. Soft voting result when there is a linear drift in S_1

circuited to supply. Because of the RC circuitry connected to the input of ADCs (Analog to Digital Converters) S_1 signal does not suddenly jump to the supply voltage and rises gradually. Soft voting detects the fault and removes S_1 signal from voting process in a timely manner. We also tried hard voting to detect the fault. Fig. 8 shows the fused signal and its expected true values in the time interval, starting 10 seconds before the short circuit event. It is observed that the short circuit is detected by hard voting after 4 seconds which is quite dangerous and unacceptable in braking application.

Pedal sensors data may also be drifted by temperature variations in motor warm-up or cool-down periods. Fig. 9 shows a linear drift of 1000 mV injected into S_1 and the result of soft voting by which the drift is detected and removed. On the other hand, hard voting method does not detect the drift, because the threshold of agreement evaluation is larger than the 1000 mV drift. Hard voting result is presented in Fig. 10. Faultiness measures resulted from soft voting in presence of the linear drift in S_1 are also shown in Fig. 11.

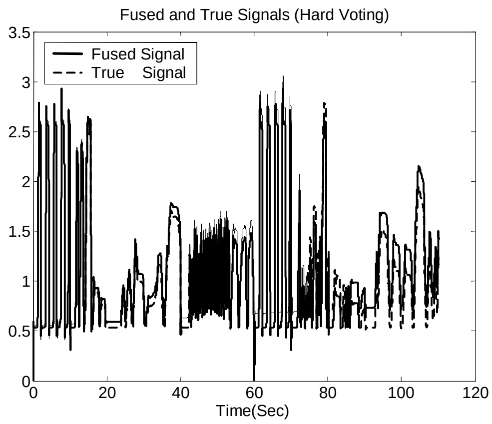

Fig. 10. Hard voting result when there is a linear drift in S_1

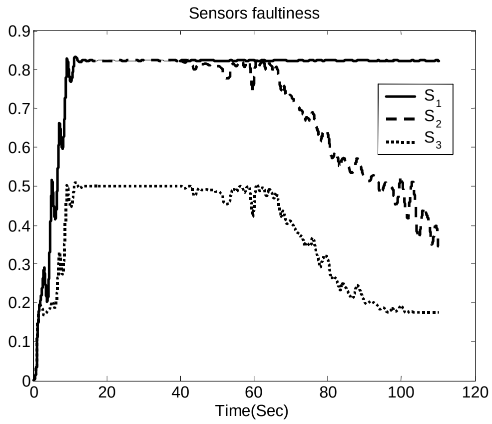

Fig. 11. Faultiness measures resulted by soft voting result in presence of a linear drift in S_1

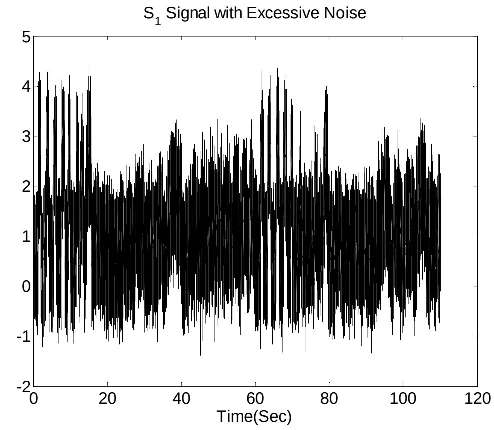

Fig. 12. S_1 signal in presence of excessive noise

It is observed that faultiness for S_1 is always large and faultiness for S_2 and S_3 are initially large but decrease while the drift in S_1 grows. A substantial level of noise was injected into S_1 signal as depicted

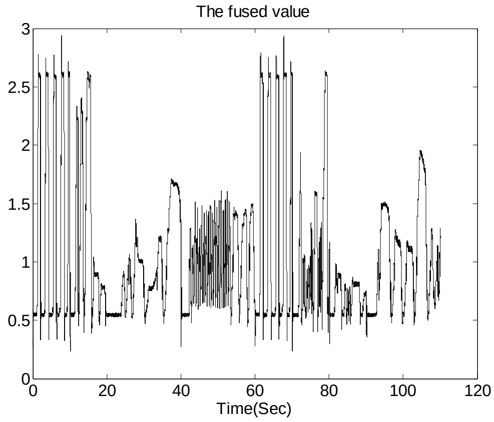

Fig. 13. Soft voting result when in presence of excessive noise in S_1

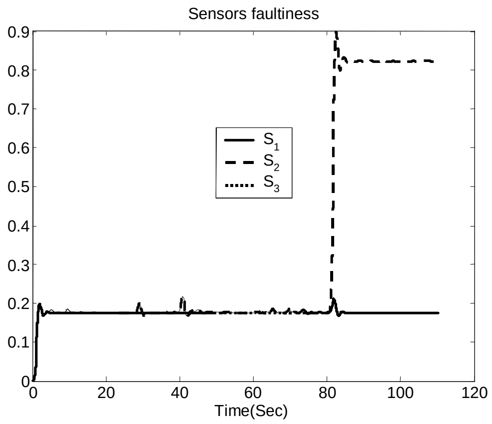

Fig. 14. Sensor faultiness measures in presence of a short circuit both for S1 and S2 sensors

Table 1. MSE error for pedal sensor fusion by soft and hard voting in presence of various faults

Injected Fault	Hard Voter	Soft Voter
Gradually Short to Ground	0.1932	0.0367
Gradually Short to Supply	0.1033	0.0272
Suddenly Short to Ground	0.2123	0.0298
Suddenly Short to Supply	0.2099	0.0245
Noise (Substantial SNR)	0.1277	0.0434
Drift	0.2108	0.0269
Total MSE	1.0572	0.1885

in Fig. 12 and soft voting could also effectively detect and remove it from sensor fusion output as shown in Fig. 13.

In order to compare the performance of majority (hard) voting method with our proposed soft voting method quantitatively, we have computed MSE (Mean Square Error) for soft and hard voting methods in presence of various faults. Table 1 shows the result of error computation. Totally, MSE is reduced by 82% in soft voting compared to hard voting and that is because of its early fault detection and removal capability. Finally we remind that our proposed method is a *voting* method, i.e. we do not expect it to detect a fault in majority of sensors (two or all sensors here). For example if a short circuit happens for both S_1 and S_2, then the hard or soft voter will deduce that S_3 is faulty because it does not agree with the other two sensors.

Fig. 14 shows the faultiness measures in such a case, while the short circuit fault happens during the time interval [80,110].

4. CONCLUSIONS

In this paper, we introduced a new method for fusion of brake pedal sensors data in a by-wire braking system, called *soft voting*. Because of the sensor conversion errors, sensor agreement thresholds in a majority (hard) voter are so large that an unacceptable delay in fault detection occurs. Our proposed soft voting method applies a fuzzy rule-base to perform voting. The fuzzy rules here are designed in such a way that the voter output is smoothly switched from one majority voted value to another in case of a sensor fault. Our proposed soft voter also gives faultiness measures for each of the sensors.

The novel point in our approach is that we calculate averaging weights as a normalised sum of products of membership values. Simplicity and timeliness attributes of our proposed method make it appropriate for real-time and safety critical applications such as braking, where computational load, convergence and stability are important issues. Experimental results show that our proposed method is successful in fault detection for many cases where hard voting approach either results in late detection or fails completely. Experiments also show that the soft voting total error (in terms of MSE) is reduced by around 82% compared to hard voting.

REFERENCES

Avizienis, A. (1985). The N-Version approach to fault-tolerant software. *IEEE Trans. Software Engng*, **1**, 1491-1501

Bass, J.M., G. Latif-Shabgahi and S. Bennett (1997). N Experimental comparison of voting algorithms in cases of disagreement. *Proceedings of 23rd Euromicro Conference*, Budapest, Hungary 516–523.

Latif-Shabgahi, G., S. Bennett and J. M. Bass (2003). Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems. *Microprocessors and Microsystems*, **27**, 303–313

Parhami, B. (1994). Voting Algorithms. *IEEE Transaction on Reliability*, **43 (4)**, 617–629.

Tong, Z. and R. Kain (1988). Vote assignments in weighted voting mechanisms. *Proceedings of the Seventh Symposium on Reliable Distributed Systems*, Columbus, Ohio, USA, 138–143.