Design and Implementation of an Acoustic-Based Car Engine Fault Diagnostic System in the Android Platform

Roy Francis Navea
Department of Electronics and Communications Engineering
De La Salle University – Science and Technology Complex
Biñan, Laguna, Philippines
roy.navea@dlsu.edu.ph

Edwin Sybingco
Department of Electronics and Communications Engineering
De La Salle University
Manila, Philippines
edwin.sybingco@dlsu.edu.ph

***Abstract*— This study aims to design and implement an acoustic-based car engine fault diagnostic system in the Android platform. The system is in the form of an Android application which can run on any phone with an Android operating system.**

A smart phone running in the Android operating system is used in order to analyze the sounds coming from the car engine. The smart phone is simply placed at an approximate distance of 1-foot away from the engine assembly. Fault diagnosis covers engine start problem detection, drive-belt analysis and tune-up detection based on valve clearance. Each fault is analyzed individually and sample engine acoustics are obtained per engine fault mentioned. The system is equipped with signal processing algorithms using Fourier transforms and spectral power densities that process the sounds obtained in order to come up with a diagnostic result and further recommend some possible solutions. The algorithm is based on the correlation coefficient of the spectral power densities (SPD), collected using two distinct clustering techniques (spectral power sum and spectral power hop), of the audio signals (test and reference engine sounds) which are fed into a fuzzy logic inference system for classification.

The design and implementation of the engine fault diagnostic system was made successful in a smart phone running in the Android platform. Fault signatures were made and were used as a reference for fault detection. Results show that the system was able to diagnose the engine faults on the target model as well as with other brands of cars.

Keywords—Android operating system, clustering, spectral power density, fuzzy logic, fault signature

I. INTRODUCTION

Sound change is usually the first symptom of a fault that leads to a malfunction. When a car engine is not in its good running condition, a possible threat of danger is inevitable to its passengers. Most of the sounds produced by automobiles come from its engine. The engine is one of the most important parts of a vehicle. Regular engine monitoring has to be done in order to maintain good running condition. Poor engine performance could be indicated by a sudden impulse or even a continuous acoustic activity which deviates from its normal pattern of behavior.

Commonly, car owners are not well-versed in diagnosing car engine problems. As one of the safety precautions, a vehicle would usually give an early warning sign to inform its driver about an anticipated engine problem through the traditional dashboard display. In some cases, there are problems which are not identified in the dashboard display such as engine rattles, timing-belt and drive-belt problems. When an engine problem gets worse, the car owner would go to an automobile expert center just to find out that the damage would cost him much as compared to when it was earlier detected and dealt properly.

Algorithms for specific engine sound recognition were done by [1] and a mobile-based application, running in Windows, for detecting engine faults was made by [2]. Microcontroller based on-board diagnostic (OBD) system was proposed by [3] to diagnose engines which are not yet equipped with OBD. A survey on diagnostic methods for automotive engines was presented by [4] to review the different work in the area of fault detection and diagnostics for automotive engines. However, there are still unexplored areas especially in the data processing technique used and the means by which engine diagnostics is accomplished.

Recently, a growing number of mobile devices are running using the Android operating system [5]. The number of software applications is increasing to meet the demands for optimum usability of these devices. For this study, an Android "app" was created running algorithms that can identify car engine conditions. It has a graphical user interface (GUI) that is capable of recording engine sounds, processing it and giving diagnostic results.

II. SYSTEM ARCHITECTURE AND ITS GUI

A. System Architecture

The proposed system is embedded in an Android-powered mobile phone equipped with a microphone that converts the sound energy to electrical energy. The analog signal captured by the transducer is converted to its digital form through an analog-to-digital converter which is commonly built-in in most electronic devices. Once it is already in its digital form, a digital signal processing technique is implemented in the core processor of the mobile phone together with other built-in peripherals such as the memory storage devices. In this way, the captured signal is processed and evaluated for a certain period of time. The graphical user interface (GUI) provides assistance to the user on how the engine fault diagnosis will go through by using the functional control buttons. Evaluation algorithms are designed to analyze the sound coming from the car's engine. The algorithm will determine and analyze the characteristics of the signal received thus provides interpretation as to

whether there has been a fault detected or none. The information coming from the DSP is converted back to its analog form through the D/A converter. The information coming out from the D/A converter is displayed on the GUI. The display monitor informs the user of the diagnostic results. It can also suggest possible solutions to solve the engine fault/s identified. Figure 1 shows the system architecture of the car engine fault diagnostic system in an Android powered mobile phone.

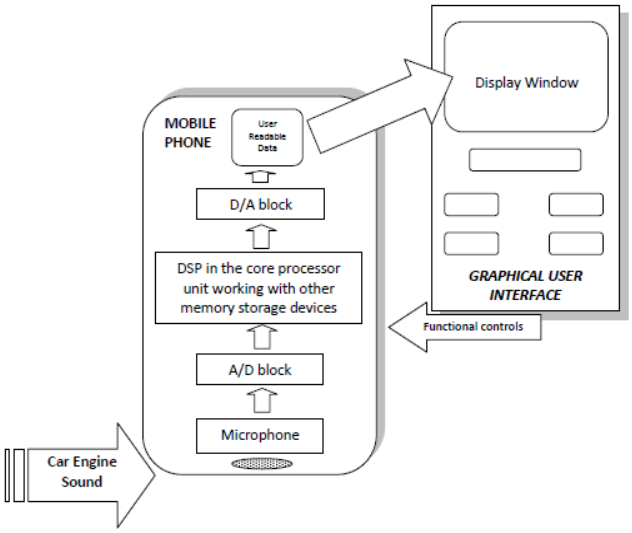
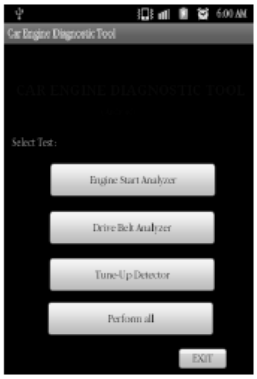


Figure 1. Block Diagram of the car Engine Fault Diagnostic System in an Android-Powered Mobile Phone

B. Graphical User Interface

The GUI is composed of five main buttons for user's control. The Main Window serves as the root for all other applications that the system can do. The main window, as shown in Figure 2.a, is composed of five functional buttons. The first four buttons refer to the three main functions of the system. Once a button is pressed, one of the four sub-windows, as shown in Figure 2.b, will appear accordingly. Each window provides a direction-for-use based on the test that shall take place. A functional button labeled "Start Test" allows the program to run as it performs the instruction set based on the programmed algorithm. Once done, the result of the test is displayed in the results area and in a pop-up message block. The user has the option as to whether to "save" or "discard" the test. A "Back to Main" button allows the user to go back to main window. The fifth button on the main window which is the "EXIT" allows the program to exit to the default screen of the mobile phone. This feature is optional since most Android mobile phones are equipped with a "back" button.



(a)

(b)

Figure 2. (a) GUI of the Main window, (b) GUI of the Sub-Windows

III. DATA COLLECTION, PROCESSING AND ANALYSIS

A. Data Collection

The study covered three major diagnostic events. Each event has its own way of data gathering procedure.

a) Engine Start Analyzer (ESA)

The data collection method for this test was made by recording the sound of an engine as it starts. Reference sounds are collected and processed first. The recording was made at various times and conditions.

b) Drive Belt Analyzer (DBA)

The data collection method for this test was made at various times and conditions by recording the sound of an engine commonly in the idle condition or running at around 1000 rpm.

c) Tune-up Detector (TUD)

The data collection method for this test was made by recording the sound of an engine in idle condition. The researcher chose a reference sound and recorded

it then followed by another reference sound for fault reference. Again, the recording was made at various times and conditions.

B. Data Processing and Analysis

The data processing and analysis part of the study is guided by the signal processing structure as shown in Figure 3.

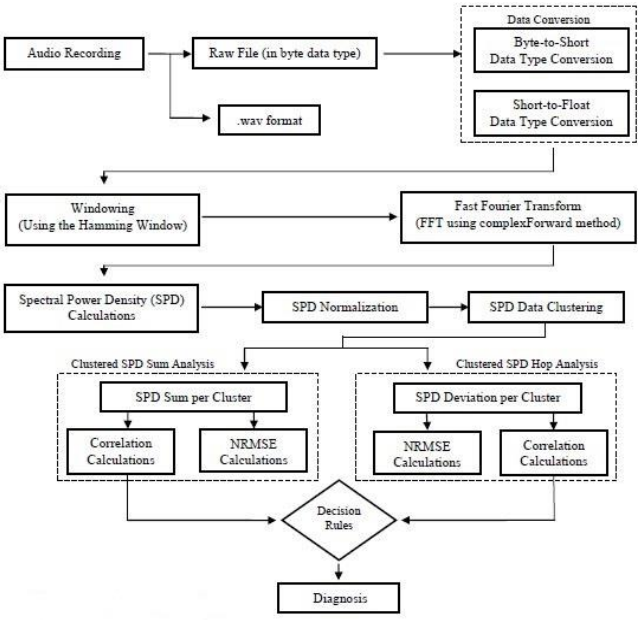


Figure 3. Data Processing and Analysis Diagram

a) Audio Data Recording

The audio recording algorithm in Java was followed after the model as presented by [6]. The audio captured from the engine is saved in the SD card of the mobile phone. Since the program is intended to record audio data, it is capable of saving the audio data in both .raw and .wav format. The .raw file requires other software in order to be played while the .wav file can be easily played by common mobile audio players. The system (Android application) developed in the study made use of the raw file stored in the SD card of the mobile phone.

The mobile recorder is placed at an approximate distance of 0.3 meters (approximately 1 ft) above the engine cover or above the belt assembly. This is to further reduce ambient noise by increasing the engine sound capture concentration. The recording time is 5 seconds for the engine start analyzer and 15 seconds for the other two tests.

b) Data Conversion

The raw audio file stored in the SD card is in byte data type. The byte audio data has a length of 4096. The byte data type is an 8-bit signed two's complement integer. It has a minimum of – 128 and a maximum value of 127 (inclusive). The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters [7]. Doing mathematical operations with bytes is not practical especially if the results are to be expressed in terms of numerical values which are most commonly used in programming such as integer, float and double.

In order to manipulate the byte data, it has to be converted to another data type. In this case, the float data type was chosen. The process goes like converting the byte first to short, then, to float. The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and maximum value of 32,767 (inclusive).

c) Windowing

Windowing is a technique used to shape the time portion of the measurement data to minimize edge effects that result in spectral leakage in the FFT spectrum. By using Window Functions correctly, the spectral resolution of the frequency-domain result will increase [8]. Before entering the FFT block, the converted audio data has to go through the windowing process. In this case, the Hamming window function was used. The Hamming window is defined as:

$$w(k) = 0.54 - .04\cos(\frac{2\pi k}{N}) \quad (1)$$

The resulting vector is multiplied element-by-element with the sample signal vector before applying the FFT [9].

d) Fast Fourier Transform Algorithm

The study implemented the complex FFT which is one of the Fast Fourier Transform methods in the FLoatFFT_1D class in the Jtransforms library by [10]. This class computes 1D Discrete Fourier Transform (DFT) of complex and real, single precision data. The size of the data can be an arbitrary number.

The complexForward method computes 1D forward DFT of a complex data. This method stores a complex number as two float values in sequence: the real and imaginary part. The size of the input array must be greater or equal to twice its length.

As for the system, the stored values in complex form were manipulated in order to calculate the magnitude of the FFT coefficients. With this, the data length was reduced from 2048 to 1024 since the array space allotted for the complex values are zeroed.

e) Spectral Power Density (SPD) Calculations and Normalization

The spectral power density (SPD) is the frequency response of a random or periodic signal. It tells where the average power is distributed as a function of frequency. The SPD is deterministic, and for certain types of random signals is not dependent on time. Calculating the SPD is useful especially for signal characterization. The SPD array of the information may be clustered into a number of segments of which in each segment, the SPD can be calculated [11].

In order to standardize SPD value computation, the normalization process was done. The process goes like standardizing the values of the SPD to a range of values in which in this study is set from 0 to 1.

f) SPD Data Clustering

This method of clustering identifies the possible SPDs of the audio signal per cluster. For initial implementation, equally spaced clusters were made. The SPD array of the audio signal was divided into 64 clusters composed of 16 data points each. The SPD in each cluster was analyzed in terms of the total SPD and the pattern of which the SPDs behave. The total SPD per cluster is the summation of the 16 SPDs in the cluster and the SPDs behavior, or the "SPD hop pattern", was analyzed by calculating the standard deviation of the 16 data points in the cluster.

g) Clustered SPD Sum Analysis

The clustered SPD sum analysis works by calculating the total SPD of each cluster. The method doing this produces a new array composed of 64 SPD sums. The SPD sums are obtained for both the test audio and the reference audio sounds. The SPD sum array of the two signals was considered in order to calculate their correlation coefficient.

h) Clustered SPD Hop Analysis

For the clustered SPD hop analysis, it works by calculating the standard deviation of the SPDs in a cluster. The standard deviation is a measure of the dispersion of a set of data from its mean. The more spread apart the data, the higher the deviation. Higher deviation indicates volatility of the data points. This means that there is a measure of uncertainty or instability on the measured values.

The Java method doing this produces a new array composed of 64 SPD standard deviations. The SPD standard deviations are obtained for both the test audio and the reference audio sounds. The correlation coefficient and NRMSE are calculated afterwards.

i) Decision Rules

The decision making capability of the system is based on the implementation of fuzzy logic in the Android platform. The fuzzy logic is composed of membership functions and rules which are followed in concluding results based on a specific defuzzification method (centroid). The fuzzy logic in the system was patterned after the fuzzy logic implementation in Matlab®. Hence, the result in Matlab® should be the same as in Android.

j) Diagnosis

The audio signal obtained from the engine is compared against a reference engine audio file. Both audio files simultaneously pass through the signal processing block. The diagnosis of the system is based on the outcome of the decision rules. The crisp value determines the fault belongingness of the coefficients obtained by correlation.

IV. RESULTS

The system was tested with thirty-five (35) car engine sounds according to the test categories of the system. Initial recordings were done in different environment conditions and in different places. These recorded sounds were processed and used for system tests. The succeeding section shows the performance results of the developed application. The reference audio sounds were taken from Honda Civic cars with year model ranging from 1996 to 2000.

In Table 1, results show that the system is capable of total recognition at a rate of 56%. Fuel, timing and battery problems are 100% recognized. The fuzzy logic implementation in the Android platform is complete at a rate of 100%. The crisp output of the system is the same as with the Matlab output.

Table 1. Engine Start Analyzer for Honda Civic Cars

ESA Test Results for Honda Civic 1996 - 2000							
	Car Model	R1-SSM	R2-SHM	CRISP VALUE		Initial	System
				in Matlab	in Android	Diagnosis	Diagnosis
1	Civic 2000a	0.209	0.0852	37.5	37.5	N	T
2	Civic 2000b	0.0338	0.0597	12.5	12.5	F	F
3	Civic 1996a	0.2379	0.2331	87.5	87.5	N	N
4	Civic 1996b	0.2479	0.0672	37.5	37.5	N	T
5	Civic 1997a	0.0863	0.1721	62.5	62.5	B	B
6	Civic 1997b	0.0648	0.1206	37.5	37.5	T	T
7	Civic 1997c	0.0497	0.0776	37.5	37.5	N	T
8	Civic 1998a	0.0683	0.0165	12.5	12.5	N	F
9	Civic 1998b	0.0352	0.2109	12.5	12.5	F	F

R1-SSM = correlation for SSM; R2-SHM = correlation for SHM
N = normal; F = fuel problem; T = timing problem; B = battery problem

The results for DBA are shown in Table 2. A total recognition rate of 56% is achieved. Pulley bearing defect is recognized at 50%. Yet, the belt-slip fault was not detected. Normal engine sounds are recognized at 67%. The implementation of fuzzy logic in the Android platform outputs crisp values equal to the Matlab's output.

Table 2. Drive Belt Analyzer for Honda Civic Cars

DBA Test Results for Honda Civic 1996 - 2000							
	Car Model	R1-SSM	R2-SHM	CRISP VALUE		Initial	System
				in Matlab	in Android	Diagnosis	Diagnosis
1	Civic 2000a	0.3397	0.2724	15	15	N	N
2	Civic 2000b	0.2642	0.117	20.7	20.728	PB	BS
3	Civic 1996a	0.0024	0.2561	22.3	22.3482	N	BS
4	Civic 1996b	0.3629	0.2207	15	15	N	N
5	Civic 1997a	0.2834	0.1461	18.7	18.6987	N	N
6	Civic 1997b	0.0624	0.1073	25	25	N	BS
7	Civic 1997c	0.183	0.1981	5	5	PB	PB
8	Civic 1998a	0.2518	0.2189	12.5	12.5408	BS	N
9	Civic 1998b	0.1659	0.3396	16.4	16.4517	N	N

R1-SSM = correlation for SSM; R2-SHM = correlation for SHM
N = normal; PB = pulley bearing problem; BS = belt-slip problem

The results for TUD are shown in Table 3. A recognition rate of 67% is achieved and the valve clearance defect is completely recognized. Valve clearance problems are 100% recognized. The implementation of fuzzy logic in the Android platform outputs crisp values equal to the Matlab's output.

Table 3. Tune-up Detector for Honda Civic Cars

TUD Test Results for Honda Civic 1996 - 2000							
	Car Model	R1-SSM	R2-SHM	CRISP VALUE		Initial	System
				in Matlab	in Android	Diagnosis	Diagnosis
1	Civic 2000a	0.1641	0.3706	62.5	62.4753	N	N
2	Civic 2000b	0.0955	0.3104	50.4	50.4336	N	N
3	Civic 1996a	0.2053	0.1884	40.8	40.8567	N	VC
4	Civic 1996b	0.2343	0.2425	75	75	N	N
5	Civic 1997a	0.0211	0.2048	27.3	27.2975	N	VC
6	Civic 1997b	0.1612	0.2061	32.3	32.3266	VC	VC
7	Civic 1997c	0.1104	0.2011	25.8	25.7991	N	VC
8	Civic 1998a	0.2658	0.1781	61.3	61.3239	N	N
9	Civic 1998b	0.0287	0.2053	27.5	27.5167	VC	VC

R1-SSM = correlation for SSM; R2-SHM = correlation for SHM
N = normal; VC = valve clearance problem

This study also explored in using the designed Android application with other car models though the variability in the mechanical design of the engines would affect the sound that it produces.

The ESA seems to be of no good when it comes to other car models considered in this study. Only a few percentage of match is seen in Figure 4. Though the reference is taken from a Civic, results show that a Sentra could possibly have the same start response with the reference. Jazz and Getz got 33.33% a piece while City has 25%.

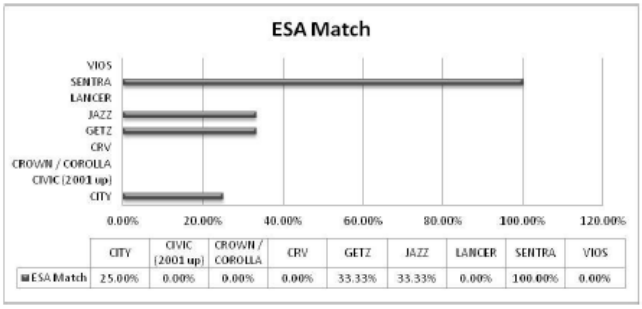


Figure 4. ESA Match results for Other Car Models

The DBA and TUD test results on other car models may seem to be worth using as the response of the system is significantly high as compared with the ESA. Higher percentages of recognition rates are observed in Figures 5 and 6 as they depict the DBA and TUD match results.

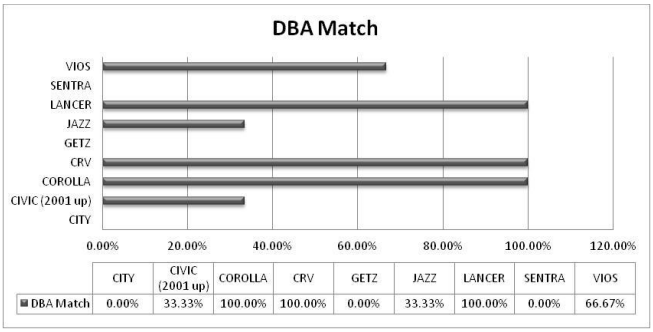


Figure 5. DBA Match results for Other Car Models

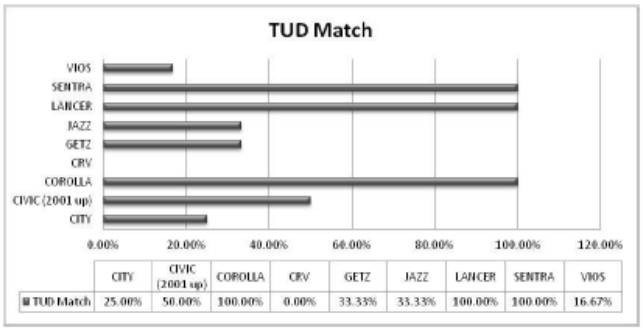


Figure 6. TUD Match Results for Other Car Models

V. CONCLUSION

This study was able to design and implement an acoustic-based car engine fault diagnostic system running using the Android platform. An algorithm was made to work in order to analyze the sound coming from the car's engine using the correlation coefficients obtained from two distinct clustering methods of the power density of the spectrum. The implementation of fuzzy logic in the Android platform performed well as the results show a complete match between the crisp values from the Android platform and the crisp values obtained from the fuzzy logic toolbox in Matlab®. A graphical user interface was designed in order to control the functionality of the system. The GUI is equipped with button controls and display menus for result visualization. The car engine diagnosis was performed at varying level of recognition rate. Thus, in most of the car models used in this study, the recognition rate is acceptably high. The system was tested first using the reference cars then, to other car models, to see the applicability of the system's usage.

REFERENCES

[1] M. Madain, A. Al-Mosaiden, M. Al-Khassaweneh, "Fault Diagnosis in Vehicle Engines using Sound Recognition Techniques," IEEE International Conference on Electro/Information Technology, pp. 1-4, 2010

[2] Y. Karunakar, A. Kuwadekar, K.A. Begain, "A Mobile Based Application for Detecting Fault Analysis in Car Engines using Triangular Window and Wavelet Transform," International Conference on Computational Intelligence and Communication networks, pp. 523-528, 2010

[3] N. Hasan, A. Arif, U. Pervez, M. Hassam, S. Shabeeh, U. Husnain, "Microcontroller based on-board diagnostic system for Non-OBD

Vehicles," UkSim 13th International Conference on Computer Modelling and Simulation, pp. 540-544, 2011

[4] J. Mohammadpour, M. Franchek, K. Grigoriadis, "A survey on Diagnostics Methods for Automotive Engines," American Control conference, pp. 985-990, 2011

[5] Android Developers. What is Android? from http://developer.android.com/guide/basics/what-is-android.html

[6] K. Varma. krvarma-android-samples. from http://code.google.com/p/krvarma-android- samples/source/browse/ trunk /AudioRecorder.2/src/com/varma/samples/audiorecorder/ RecorderActivity.java

[7] Oracle and/or Affiliates. Primitive Data Types. from: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

[8] NI Developer Zone. Windowing: Optimizing FFTs Using Window Functions. from: http://www.ni.com/white-paper/4844/en

[9] IDL Online Help. Using Windows. from: http://www.physics.nyu.edu/grierlab/idl_html_help/signal9.html

[10] P. WendyKier, Jtransforms. from: https://sites.google.com/site/ piotrwendykier/software/jtransforms

[11] Power Spectral Density from faculty.etsu.edu/blanton/lab_3_psd.doc

APPENDIX A: Fault Signatures