Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Internal Combustion Engine Fault Detection through Audio Analysis

Michael McAndrew

May 3, 2021

Supervisor: Dr. Fergal Shevlin

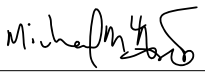
A dissertation submitted in partial fulfilment
of the requirements for the degree of
MCS (Computer Science)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed:  Date: 03/05/2021

Abstract

Internal combustion engines frequently develop faults. Experienced mechanics and technicians can often identify an engine fault by the sound it makes. These skills require significant training to master. If it is possible for humans to identify these sounds, it too could be possible for an automated system to diagnose faults. This dissertation presents a system that uses smartphone audio as a non-invasive method to identify engine faults by sound. A system to detect engine RPM, engine misfires and low cylinder compression is proposed by using methods from Digital Signal Processing and Music Information Retrieval. Time-frequency domain features are considered to identify features produced by engine events. Fault states are inferred from abnormal occurrence of these features. Statistical analysis is also used to determine engine cycle similarity when turning the engine over. A metric is then devised from this data to infer low cylinder compression. Experiments are devised to evaluate the system. Experiments use both iOS and Android smartphones, inside and outside the vehicle. It is found that the approach is effective at identifying misfires and low engine compression across many smartphone types and recording positions. The results indicate that the approach performed well even in sub-optimal recording conditions. The results of this dissertation will be used to develop further work in the area of internal combustion engine fault detection.

Acknowledgements

I would like to thank a number of people for whom this dissertation would not be possible without. Firstly, I would like to thank my supervisor, Dr. Fergal Shevlin, for allowing me to undertake the project and for his unwavering support, guidance and encouragement throughout the project.

I am grateful for the overwhelming support and help received by my friends for the past four years. I would also like to express my sincere gratitude to my family for their countless support in my academic endeavours. Without them, this dissertation would not have been possible.

Finally, I would like to include a special note of thanks to Laura Fitzpatrick for the endless support, motivation and encouragement she has given over the years, for her vital feedback and expertise in academic writing provided throughout the project.

Contents

List of Figures

List of Tables

1 Introduction

The detection of vehicle faults is essential to ensuring that the growing number of vehicles in use today are roadworthy, reliable and efficient. In Ireland alone, a total of 2.8 million private cars were registered in 2019. The fleet of vehicles is also ageing, with 1.4 million registered vehicles being four years old or more [1].

Identifying vehicle faults early is critical to ensure that serious or permanent damage to the vehicle does not occur. Modern vehicles use computer systems to monitor engine operation. While some of this data is available to vehicle owners and mechanics, it requires specialised equipment to access. To diagnose engine faults in vehicles without computer monitoring systems or where these systems are inaccessible, mechanics listen to the engine to perform fault detection.

Advances in mobile device computational power and sensor technology have changed how we perform sensing in many domains. Pervasive sensing using mobile devices is a rapidly growing field of research and application [2].

This dissertation investigates the use of mobile device microphones to detect engine faults. It is hypothesised that if experienced mechanics can identify engine faults through sound, the same faults could also be detected algorithmically by analysing audio captured by a mobile device.

Section 1.1 considers at a high level, the applications and use cases for the system. The general area of investigation for the duration of the project is defined in *Section 1.2*. Finally, *Section 1.3* outlines the structure for the remainder of the document.

1.1 Motivation

Internal combustion engine vehicles frequently develop faults. In Ireland in 2019, 51.2% of all vehicles failed the National Car Test (NCT) on initial inspection [1]. Of these vehicles, an average of 6.2% failed due to engine noise and exhaust issues, and 6.1% failed due to engine emissions failures [3].

Currently, engine faults are averted by performing preventative maintenance at service

intervals and by regularly inspecting the health of a vehicle's engine. The methods of engine health inspection often require previous experience with similar faults or expensive specialist tools.

Smartphone microphones are relatively low cost and readily available non-invasive sensors that could be used by mechanics, car owners and car buyers alike as a tool to determine the operating state of the internal combustion engine of a vehicle.

1.1.1 Vehicle Maintenance

Vehicle manufacturers create carefully documented maintenance schedules for their models for technicians to follow to ensure that reliability is maintained throughout a vehicle's lifetime.

Most vehicles are serviced annually or after a certain number of kilometres travelled, either by dealer technicians or by an independent mechanic. Minor servicing typically includes consumable component changes and a general inspection of the vehicle for any other components that have failed or are near failure. This inspection should include a health check of the vehicle's engine.

Currently, mechanics inspect engine health by:

- listening by ear to the engine to identify faults.
- using diagnostic tools.
- using engine test instruments.

Listening to an engine for the presence of fault sounds requires prior knowledge of the engine and the sound and behaviour of that fault. This can be challenging for apprentices and junior mechanics with less experience. Very often, faults subtly present themselves audibly. As a result, the fault sound can be missed during inspection.

Diagnostic tools communicate with the Engine Control Unit (ECU) to detect fault codes and monitor engine sensors. All EU vehicles from January 1, 2004, must be fitted with an On-Board Diagnostics (OBD-II) interface to allow generic diagnostic tools to retrieve real-time information about a vehicle and its errors [4]. However, while the OBD-II standard provides a practical method for obtaining real-time information about a vehicle (RPM, throttle position, drive train), not all errors in the standard need to be supported by manufacturers [5]. Many manufacturers provide extra diagnostic information that can only be decoded by their diagnostic tools at a premium cost or by third parties who have reverse engineered the manufacturer's protocols. Furthermore, OBD-II error codes can also appear from a faulty sensor rather than from an engine fault.

Engine test instruments such as stethoscopes can be placed on different parts of the engine

to isolate sounds from that region to help with sound-based fault diagnosis. Compression testers measure engine cylinder compression by removing the cylinder spark plug, inserting the tester and turning over the engine. These instruments either require considerable prior knowledge of fault sounds or are invasive tools that take time to install and use correctly.

1.1.2 Vehicle Ownership

Currently, if a vehicle owner notices that the engine is performing poorly or there is an unusual sound originating from the engine, they can either organise a vehicle inspection at their expense and inconvenience or ignore their suspicions. In some cases, there may be a serious fault with the vehicle. Furthermore, if a vehicle owner is an enthusiast who wishes to maintain their vehicle themselves, they may not have access to advanced vehicle diagnostic tools to find faults and may have to bring the car to a dealer or garage with the equipment needed to make the diagnosis.

Another use case of the system is purchasing used vehicles, particularly with high mileage and no official servicing history. It is recommended that a mechanic inspect a car before purchase. However, this is not always possible, especially with lower-priced vehicles. This results in used vehicle purchases being high risk even if the vehicle is functioning correctly and has no engine issues.

1.2 Objectives

This dissertation aims to investigate what information can be extracted from smartphone audio data captured of a vehicle in several operating states and to determine whether faults can be detected from this information. It aims to use audio analysis techniques from digital signal processing (*Section 2.3*) and music information retrieval (*Section 2.4*) to identify features referring to certain engine events and to use the information to infer a fault state. The primary objective is to investigate the presence of **engine misfires** and **low engine compression** in particular within the audio data discussed in greater detail in Section 2.2.1 and *Section 2.2.2* respectively.

Audio data is collected from several vehicles, using several smartphones to achieve these objectives. The engines are either running at low idle when detecting misfires or turning over but not running in the case of detecting compression. Algorithms are created from the data to identify engine events and detect faults. The algorithms are then evaluated to measure their effectiveness.

1.3 Dissertation Structure

This dissertation begins by providing an introduction to engine theory, engine fault detection and the techniques used from Digital Signal Processing and Music Information Retrieval to build an audio fault detection system (*Chapter 2*). This is followed by a detailed description of the approach and implementation of each fault detected (*Chapter 3*). The rationale and design decisions made are presented. The system is then evaluated and tested both with sample audio and statistical analysis (*Chapter 4*). Finally, the outcomes, challenges and future work for the project are discussed (*Chapter 5*).

2 Background

Identifying engine faults using audio analysis requires concepts and techniques from several fields. Internal combustion engine theory was used to contextualise the audio analysis. Existing engine fault detection methods were used to perform the data collection process. Knowledge of digital signal processing techniques and music information retrieval techniques were required to process and detect the engine faults.

This section serves as an introduction to each of these areas to provide context to the approach used in *Section 3* and to further explore the reasoning for the research decisions made in this dissertation.

2.1 Internal Combustion Engines

This section highlights engine theory concepts discussed in detail in *Chapter 1* of [6] that are required to understand the implementation of the fault detection system.

Internal combustion engines contain pistons that move linearly inside a chamber known as a cylinder. When a piston is at the top of the piston, it is referred to as *Top Dead Centre* (TDC). Likewise, when a piston is at the bottom of the cylinder, it is known as *Bottom Dead Centre* (BDC). When a piston moves from TDC to BDC or vice versa, this motion is referred to as a stroke. Each piston is then coupled by a connecting rod to a crankshaft that converts the linear motion of the pistons to rotation. The crankshaft turns the wheels of the vehicle via the gearbox.

Internal combustion engines require three elements to operate: fuel, air and ignition. A mixture of fuel and air is drawn into each cylinder. The rate at which the mixture is drawn in is controlled by an intake valve. The burnt gases produced from ignition exit to the exhaust system by an exhaust valve. A four-stroke process is then repeated for the engine to operate:

1. **Intake stroke** - The intake valve opens and the piston moves from TDC to BDC, drawing a mixture of fuel and air into the cylinder.

2. **Compression stroke** - The intake valve closes and the piston begins to move to TDC, compressing the fuel and air mixture within the chamber.
3. **Power stroke** - Compressed fuel and air mixture are ignited by a spark plug. This applies a force to the piston, moving it to BDC.
4. **Exhaust stroke** - The exhaust valve opens and the piston moves to TDC, pushing the burnt gases out of the engine to the engine exhaust pipe.

Ignition spark is created by spark plugs. On modern vehicles, an induction coil known as a coil pack is used generate the large voltage required to create the required spark.

The intake and exhaust valves are controlled by a rotating camshaft. As the camshaft rotates, it opens and closes the valves in sequence. The camshaft moves in sync with the crankshaft to ensure that the valves open and close at the correct time.

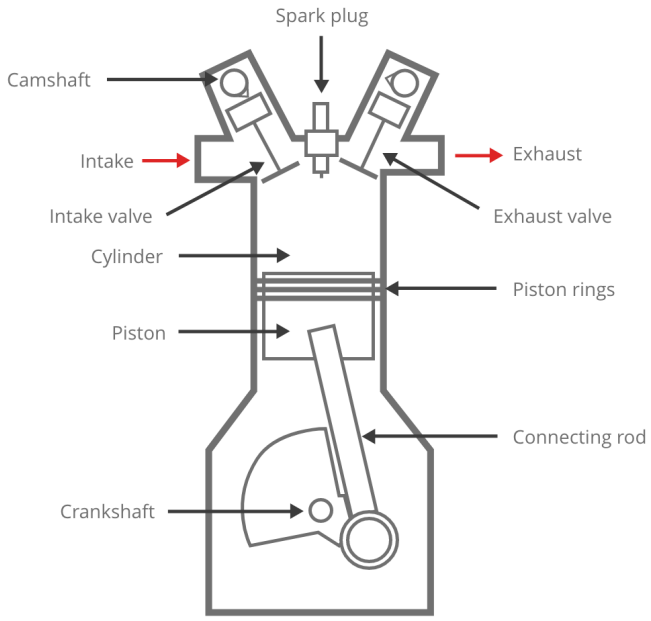


Figure 2.1: Cross-section view of four-stroke engine

Most modern vehicles have several cylinders in various configurations. For the context of this dissertation, only the inline four-cylinder engine configuration will be discussed. As the name suggests, inline four-cylinder engines have four cylinders in line with each other placed over a crankshaft. The pistons move in pairs. Most frequently, cylinders 1 and 4 and 2 and 3 (where cylinder 1 is the cylinder closest to the front of the engine) move up and down together and are offset by one stroke. This results in one pair of pistons always moving towards TDC while the other pair moves towards BDC. Nearly all modern inline four-cylinder

engines use a firing order of 1-3-4-2. The firing order is the order in which cylinders are in their power stroke.

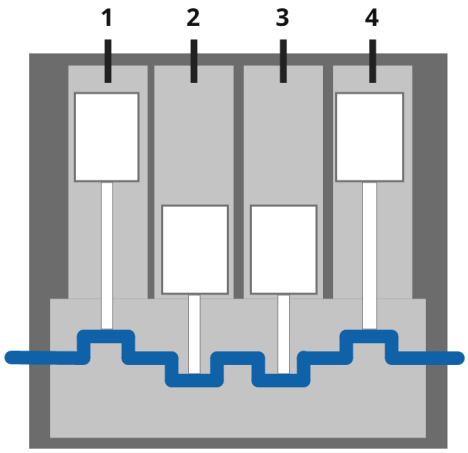


Figure 2.2: Inline-four cylinder engine configuration

Modern engines are started with a starter motor. It connects to the crankshaft and turns it to help start the combustion process.

2.2 Engine Fault Detection

2.2.1 Engine Misfires

Relevant engine misfire information is summarised from [7]. A misfire occurs when there is no ignition in the cylinder. This can be caused by an absence of either air, fuel or ignition. Misfires cause a loss in power and potential damage to the vehicle's catalytic converter if not resolved quickly as unburnt fuel is released from the cylinder via the exhaust. When an engine is misfiring, there is usually a noticeable change in its sound. The engine can also visibly shake considerably more than when all cylinders are firing.

Misfires can also occur intermittently. Intermittent misfires can be very difficult to detect audibly to the human ear as they may only occur once over several minutes.

Current Solutions

Experienced mechanics can identify a misfire from audible and visual changes to the engine operation [8]. They can determine which cylinder is misfiring by disconnecting either the high-tension leads or coil packs to the spark plugs individually for each cylinder. If disconnecting the ignition source to a cylinder does affect the engine sound, that cylinder is not firing.

Sound-based techniques can also be used to identify intermittent misfires, although it is more difficult. The mechanic may also observe the tachometer for a slight inconsistency in the vehicle's idling RPM as a sign of an intermittent misfire.

A vehicle Engine Control Unit (ECU) can detect misfires. They can measure the crankshaft position by use of a hall effect sensor [5]. Anomalies in the crankshaft speed are caused by a misfire. The ECU can detect this and cause an error code that can be visible on the OBD-II interface. There is no standardised method of classifying misfires from crankshaft position information. It should be noted that in testing performed for this dissertation, a misfire related error code was only observed in one vehicle after inducing misfires.

Audio-based engine fault detection has also been investigated to varying degrees of success. One study uses an Artificial Neural Network to classify several different engine faults with three different engines [9]. The results of the study varied in accuracy between 85% and 95% across the three engines. Other unseen engines were not evaluated in the test. Detecting misfires by audio has also been implemented successfully using various feature extraction methods, and a Support Vector Machine achieving 99% accuracy [10]. Four different vehicles were used, two with inline four-cylinder engines and two with V6 configuration engines. However, the approach was limited as it was not tested to detect intermittent misfires and was not tested with audio inside the vehicle.

2.2.2 Compression Testing

An engine compression test is often used as an accurate indicator of its health [11]. High cylinder compression is required for efficient combustion. If the fuel and air mixture within the cylinder is not sufficiently compressed, the engine will lose power due to poor ignition or potentially by misfiring entirely. Compression can also indicate the internal engine wear. Low compression in a cylinder is generally caused either by leaks in the cylinder head at the top of the engine or between the piston and the cylinder wall.

Current Solutions

Mechanics can perform a compression test on a cylinder by removing its spark plug and threading in a pressure gauge [11]. The pressure gauge has a one-way valve used to measure the maximum pressure over many engine cycles. Pressure gauges can be purchased relatively inexpensively and are an invaluable tool to a mechanic. However, in some cases, the use of a pressure gauge to perform a compression test may not be possible or optimal. Compression testing each cylinder in a vehicle, particularly if it has many cylinders, can be a time-consuming process. Spark plugs can also be challenging to access on modern vehicles.

To quickly diagnose low compression in one or more cylinders, mechanics often use a relative

compression test [12]. A current clamp can be attached to the high tension starter motor cable to observe the current draw from the battery when turning the engine over. If a cylinder has lower compression than the other cylinders, the starter motor will draw less current on its compression stroke as it requires less force to move the piston.

A relative compression test takes significantly less time than individually compression testing each cylinder with a pressure gauge as all cylinders can be tested at once. However, relative compression testing requires a specialist oscilloscope and a high current clamp meter [12].

2.3 Digital Signal Processing (DSP)

DSP is the manipulation of discrete signals such as audio data. DSP techniques can transform time-domain signals into the frequency domain and extract features in both time and frequency domains. This section defines and discusses some of the DSP techniques used in this dissertation, focusing on their application to audio signals.

2.3.1 Discrete Fourier Transform (DFT)

The DFT of a Discrete Time Signal (DT-signal) $x(n)$ is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{i2\pi}{N}kn} \tag{2.1}$$

where

$k = [0, N-1], F(k) = \frac{k}{NT} = \frac{ks_r}{N}$

s_r is the sample rate of the time domain audio

n is the given time domain audio sample

N is the number of samples in the time domain audio

The output of a DFT provides the phase and magnitude information for each frequency bin.

Similarly, the inverse DFT of a frequency domain signal $X(k)$ is defined as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{i\frac{2\pi}{N}kn} \tag{2.2}$$

Fast Fourier Transform

The DFT is computationally expensive and has a runtime complexity of $O(N^2)$. This project is intended to be capable of running on smartphones in a somewhat real-time fashion. Because of this, all DFT's are calculated using the Fast Fourier Transform (FFT) [13]. The FFT exploits redundancies across sinusoids resulting in an improved runtime complexity of $O(Nlog_2N)$. However, it only works when N is a power of 2.

2.3.2 Short-Time Fourier Transform (STFT)

The STFT is used to determine the frequency components of a signal at a point in time. The signal is split into small, equal-sized frames of time m, and a DFT is applied to each frame. The STFT is defined by [14]:

$$X(m,k) = \sum_{n=0}^{N-1} x(n+mH) \cdot w(n) \cdot e^{-i2\pi n\frac{k}{N}} \tag{2.3}$$

where

m is the given frame number

k is the given frequency

N is the number of samples in the frame

x is signal function

w is window function

H is hop size

The result is a number of frequency bins for each frame of audio. The number of frequency bins is proportional to the frame size and is described by:

$$\text{No. Frequency Bins} = \frac{framesize}{2} + 1 \tag{2.4}$$

$$\text{No. Frames} = \frac{samples - framesize}{hopsize} + 1 \tag{2.5}$$

The Window Function

Before the DFT is applied to each frame, a windowing function [15] is applied to the signal within the frame region. This essentially weights every value in the frame based on the

shape of the windowing function.

The size of both the frame N and the window w can be specified when performing an STFT, although they are equal in size in most implementations. However, the window size can be smaller than the frame size, in which case any frame values outside of the window function size are set to zero.

Spectral Leakage

Windowing functions can be applied to frames to minimise spectral leakage as discussed in *Chapter 2.5.1.2* of [16].

Spectral leakage occurs when the signal being processed is not a perfect integer number of periods. Typically the points at the beginning and the end of the signal are discontinuous, resulting in high-frequency artefacts in the frequency-domain transform of the audio that are not present in the original signal.

Applying a window removes discontinuities at either end of the frame by smoothing them to zero.

Frame Overlap

The problem with applying windowing functions to frames to remove spectral leakage is that parts of the signal are lost at the edges of the frame. To solve this, frames can be allowed to overlap, determined by the hop size H. This ensures that information removed from previous frames due to windowing is included in subsequent frames.

STFT's are typically plotted visually as spectrograms.

Parameters

- Frame Size - increasing the frame size improves the frequency resolution as more audio samples are available to generate a DFT. It does decrease the time resolution, however, as the frame represents a larger period of time. The inverse is also true.
- Hop size - common hop size values are determined relative to the frame size (e.g. $\frac{K}{4}$).
- Windowing function - this affects the result of the DFT for each frame. Using a rectangular windowing function causes high-frequency artefacts from frame discontinuities. The most common windowing function is the Hann window.

Visualisation

STFT's are most commonly displayed as spectrograms. The complex output of the STFT $S(m, k)$ can be converted to a real output $Y(m, k)$ to be displayed visually by:

$$Y(m,k) = |S(m,k)|^2 \tag{2.6}$$

Each frame is displayed along the x-axis, frequency along the y-axis and the magnitude of each frequency as a colour value of changing intensity.

Humans perceive sound intensity logarithmically. It is very common to display the amplitude logarithmically (in dB) to visualise the sounds as humans hear it. It is calculated for a signal of real-valued amplitudes by:

$$L_p = 10 \cdot log_{10}(\frac{P}{P_0})dB \tag{2.7}$$

While displaying frequencies linearly is often useful when wishing to emphasise harmonic components within the signal, displaying the frequency domain logarithmically can help identify tonal changes in the signal over time. Displaying the frequency spectrum on the log scale provides more space to the low frequencies and less to high frequencies.

2.3.3 The Mel Scale

Humans perceive difference in frequency and in pitch non-linearly [17]. This can be challenging when analysing frequency-domain audio intuitively as the relationship between what we hear and what is visually represented is not scaled correctly.

The Mel scale [17] is logarithmic, whereby equal distances on the scale are of equal difference in pitch to humans. The Mel scale is standardised such that the frequency and Mel scales intercept at 1000Hz/1000 mels. The Mel scale was derived by performing experiments on human subjects to determine the exact perceptual difference that humans perceive in different frequency bands. The Mel scale is defined by:

$$m = 2595 \cdot log(1 + \frac{f}{700}) \tag{2.8}$$

$$f = 700(10^{\frac{m}{2595}} - 1) \tag{2.9}$$

Mel Spectrogram

The Mel scale can be applied to an STFT to create a more perceptually accurate spectrogram known as the mel-spectrogram [18]. Frequency bins from the STFT can be converted to the Mel scale by:

1. Choose the number of Mel bands to compress the STFT to - this is the number of

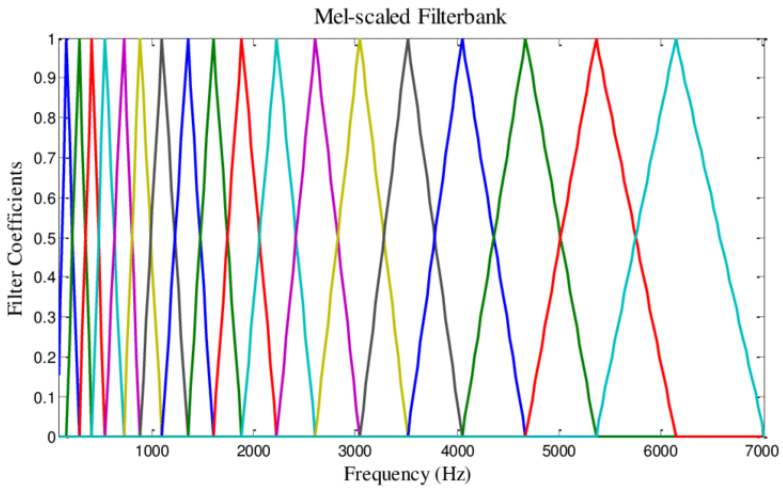


Figure 2.3: An example of 20 Mel filter banks generated from 20 Mel bands applied to 0-7000Hz frequencies.

bands that will be used to represent the frequency domain. This number is chosen based on the spectral resolution required by the application.

2. Construct Mel filter banks
   (a) Convert the lowest and highest frequencies in each STFT frame to Mel by *Eq 2.8*.
   (b) Split the region on the Mel scale between the lowest and highest Mel into equally spaced points for each Mel band.
   (c) Convert the Mel points for each Mel band back to Hertz using the inverse Mel scale formula described by *Eq 2.9*.
   (d) Map the converted frequencies to the nearest frequency bin in the STFT.
   (e) Create a set of triangular filter banks that map frequency to Mel bands as shown in *Figure 2.3*.
3. Apply the Mel filter bank M to the STFT Spectrogram Y by Matrix Multiplication Mel Spectrogram $= MY$.

Mel Filter Banks

Mel filters are simple triangular filters. The filter banks determine to what degree a frequency applies to each filter bank. A filter is defined for each Mel band with a peak at the centre frequency of the Mel band and the y-intercepts of the filters at the centre of the previous and subsequent Mel bands.

The result is a matrix that assigns each Mel band a weight for each frequency bin of the STFT.

2.3.4 Cross-Correlation

Cross-correlation is a measure of similarity of two signals formally described as:

$$Z(n) = \sum_{m=0}^{N-1} f(m) \cdot g(m+n) \tag{2.10}$$

where

f and g are the cross-correlation signals

N is the length of f

$n = [0, len(g) - N]$ and is the lag

Pearson Correlation Coefficient

The sliding dot-product cross-correlation metric is not normalised. This causes difficulties when intuitively interpreting the results. The Pearson Correlation Coefficient measures the linear correlation between the two signals within the range $[-1, 1]$ where -1 is strong negative correlation and 1 is strong positive correlation. It is calculated as the covariance between the two signals divided by the product of their standard deviations as defined by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \hat{x}) \cdot (y - \hat{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \hat{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \hat{y})^2}} \tag{2.11}$$

This can also be applied to the cross-correlation of two signals as defined by:

$$R(n) = \frac{\sum_{i=0}^{N-1}(f(i) - \hat{f})(g(n+i) - \hat{g}(n))}{\sqrt{\sum_{i=0}^{N-1}(f(i) - \hat{f})^2} \cdot \sqrt{\sum_{i=0}^{N-1}(g(n+i) - \hat{g}(n))^2}} \tag{2.12}$$

where

N is the length of f

$\hat{f} = \frac{1}{N}\sum_{i=1}^{N-1} f(i)$ (the mean of f)

$\hat{g}(n) = \frac{1}{N}\sum_{i=1}^{N-1} g(n+i)$ (the mean of $g(n)$)

2.3.5 Root Mean Square (RMS) Energy

The RMS energy of a signal x of length N is defined as:

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{n}^{N-1}|x(n)|^2} \tag{2.13}$$

2.4 Music Information Retrieval (MIR)

MIR is the discipline of extracting information from music. It is typically used in applications such as music classification, transcription, recommender systems and source separation.

Music and internal combustion engine audio share many similar characteristics. Most music is periodic and contains repeating events and patterns. Music can be decomposed by its instruments and sound sources. Audio feature extraction from music can be informed by knowledge of the notes, chords and instruments used to create the music. Music can also be transcribed by detecting notes, their duration, harmonic and melodic information.

In many ways, the sound produced by an engine could be compared to that of a repeating song. It is highly periodic when operating correctly. Individual engine sound events can also be informed by the events that occur when an engine is operating. Engine theory can inform when events occur and their duration.

This section discusses some techniques from MIR that were applied to detect engine faults in this dissertation.

2.4.1 Harmonic-Percussive Separation (HPS)

Harmonic sounds are perceived as pitched sounds that persist over a long period of time. In music, harmonic sounds are what produces notes, chords and melodies. Harmonic sounds appear as horizontal lines in a spectrogram representation. **Percussive sounds**, on the other hand, are sharp impulses such as bangs, claps and clicks. In music, percussive sounds are what produces the beat or rhythm of the song. Percussive sounds appear vertically in a spectrogram representation. This occurs because an impulse is very close to zero everywhere except at a single point in time. The magnitude of the Fourier transform becomes broader the shorter the impulse is. This results in the signal's energy being spread across the entire spectrum.

There are many different approaches to creating HPS of a signal. In this project, I use the method outlined in [19]. The method can be summarised as:

1. Calculate the STFT of the signal, as described by *Eq 2.3*.
2. Apply median filtering in both the horizontal and vertical direction to retrieve enhanced spectrograms in each direction:

$$\tilde{Y}_h(m,k) = median(Y(m-l_h,k),\dots,Y(m+l_h,k)) \tag{2.14}$$

$$\tilde{Y}_p(m,k) = median(Y(m,k-l_p),\dots,Y(m,k-l_p)) \tag{2.15}$$

where

$2l_h + 1$ is the length of the harmonic median filter
$2l_p + 1$ is the length of the percussive median filter

3. Apply a *separation factor* β to the masks. Intuitively, it determines a threshold for which a sound can be included in a component by at least a factor of β in order to be included in the other component:

$$M_h(m,k) = \frac{\tilde{Y}_h(m,k)}{\tilde{Y}_p(m,k)+\epsilon} > \beta \tag{2.16}$$

$$M_p(m,k) = \frac{\tilde{Y}_p(m,k)}{\tilde{Y}_h(m,k)+\epsilon} >= \beta \tag{2.17}$$

where

ϵ is a small number used to prevent divide by zero errors

4. Apply the masks to the original spectrogram:

$$X_h(m,k) = S(m,k)\cdot M_h(m,k) \tag{2.18}$$

$$X_p(m,k) = S(m,k)\cdot M_p(m,k) \tag{2.19}$$

5. Apply the inverse STFT to X_h and X_p to get the time domain signal of the harmonic

and percussive components.

Parameters

- Kernel Size - the size of median filter to use. This determines the minimum size required of a horizontal or vertical component to be considered part of the harmonic or percussive component.
- STFT Frame Size - large frame sizes average the frequency spectrum over a larger period of time and, as a result, emphasise harmonic components and vice versa.
- Separation Factor - the larger the value, the less leakage between the harmonic and percussive components occur.

2.4.2 Peak Picking

Peak picking is a commonly used technique in MIR to identify magnitude features in a signal. It works by first identifying all local maxima in the signal. That is any point that is greater than each of its neighbouring samples:

$$\begin{gathered} x_0 \in X \text{ is a local maximum point of function } f \text{ if} \\ (\forall x \in X) d_X(x, x_0) < \epsilon \implies (f(x_0) \geq f(x)) \wedge (f(x_0) \geq \gamma) \end{gathered} \tag{2.20}$$

Additional parameters can also be used to filter peaks by:

- height γ - the required magnitude of the peak
- distance ϵ - the horizontal distance between its neighbouring peaks

2.4.3 Energy-Based Novelty Detection

Novelty detection is commonly used in MIR to identify the onset of a new musical note. Intuitively, energy-based novelty detects sudden increases in energy as candidates for the onset of musical notes.

Energy-based novelty has been widely replaced by spectral-based novelty because most modern music is polyphonic, with many music events occurring at any moment in time. However, it can help detect event onsets from engine sounds.

Energy-based novelty is described in *Chapter 6.1.1* of [16]. Summarised, the method is defined as:

$$E_w^x(n) = \sum_{m=-M}^{M} (x(n+m) \cdot w(m))^2 \tag{2.21}$$

where

x is a DT-signal

w is a discrete window function which is shifted over x

The first derivative of $E_w^x(n)$ is calculated to provide peaks in regions with large changes in energy. The signal is also half-wave rectified so that only positive energy changes are considered:

$$\Delta_{Energy}(n) = |E_w^x(n)(n+1) - E_w^x(n)|_{\geq 0} \tag{2.22}$$

Magnitude thresholding or peak picking (as described in *Section 2.4.2*) can then identify onset points from the novelty function.

3 Method

This chapter describes the development of a system for identifying internal combustion engine faults. Recordings were made from smartphones of several vehicles and were used to inform implementations of a *Engine RPM*, *Misfire Detection* and *Relative Compression Detection* system. Data collection, audio analysis and algorithm implementations are discussed in detail.

3.1 Detecting Vehicle RPM

I created a toy problem to investigate a smaller, more manageable challenge that shared similar characteristics with the larger problem of the project. The goals of investigating the toy problem were to:

- begin researching and applying common audio analysis techniques
- work with smartphone audio to determine the types of data that could be extracted and to learn its limitations
- attempt to identify a property of an internal combustion engine for a very specific audio sample

I attempted to identify the Revolutions Per Minute (RPM) of a vehicle's engine.

The goal of this toy problem was not to form a robust and well-tested algorithm but rather to see if the problem can be solved for a very small subset of data.

3.1.1 Data Collection

A single audio recording of a 2004 Audi TT with an inline four-cylinder engine was made using an iPhone SE outdoors with the bonnet of the vehicle open directly over the vehicle's radiator, pointing towards the engine.

The audio was recorded using the iOS *Voice Memos* application. This produced a mono audio file with a sample rate of $48,000Hz$.

The vehicles tachometer was used to obtain an estimate of the vehicles RPM during the recording.

Figure 3.1: Image of audio the recording process

3.1.2 Audio Analysis

The analysis began by visualising the signal in the time domain (*Figure 3.2a*) and time-frequency domain using a spectrogram representation (*Figure 3.2b*). The audio was also analysed by ear by slowing down the recording by 500% and 800%, sequentially. This facilitated events such as cylinder fires to be audible over a longer period of time. Slowing down the audio also decreased its pitch. This resulted in very high-frequency information being more easily audible to the human ear.

A combination of the visual analysis and listening to the slowed-down signal resulted in identifying events in the spectrogram that corresponded with each engine combustion event (*Figure 3.2c*).

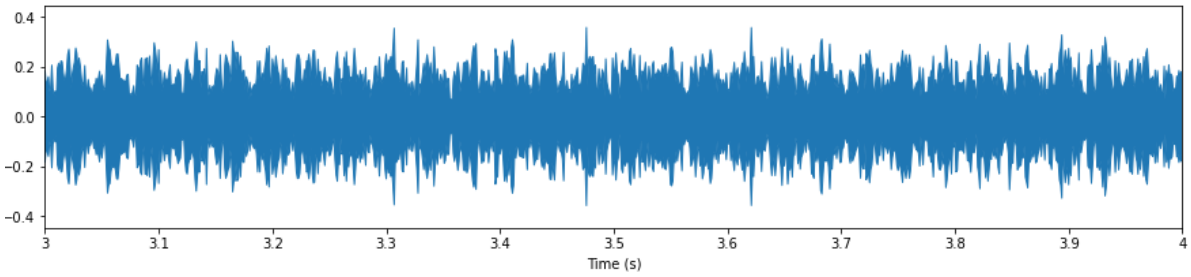
Relationship Between Pitch and RPM

On an inline four-cylinder engine, each piston stroke rotates the crankshaft by 180°. In other words, one rotation of the crankshaft is two piston strokes. RPM can be derived by counting S, the number of piston strokes in a second:

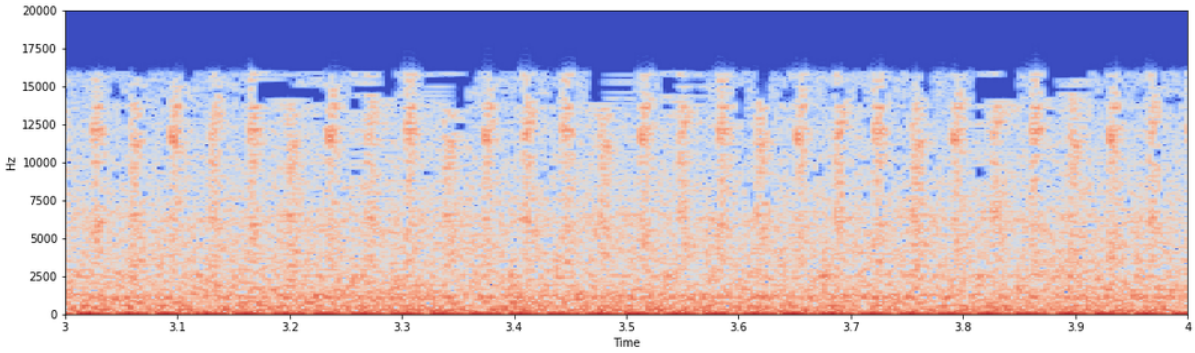
$$\text{RPM} = \frac{S}{2} \cdot 60 \tag{3.1}$$

$$S = \frac{RPM}{60} \cdot 2 \tag{3.2}$$

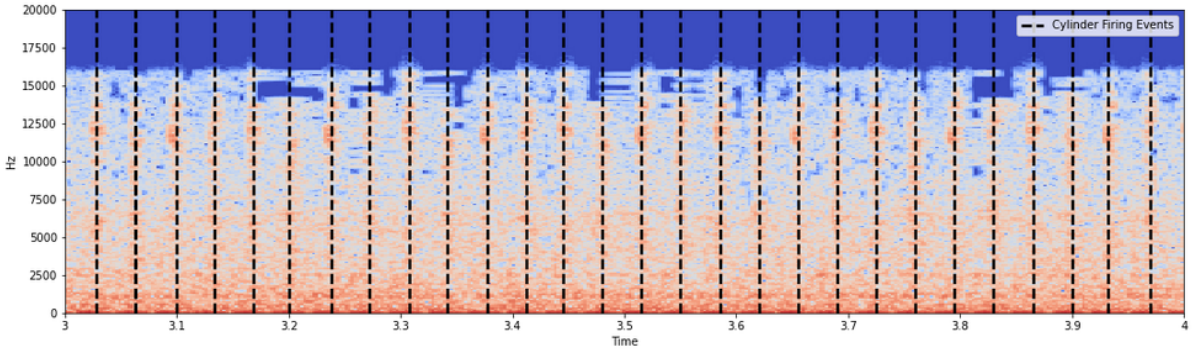
The signal from an engine firing should be periodic. If the engine firing frequency, S, is high enough, a frequency component with a high magnitude should be present in a DFT



(a) DT-signal



(b) STFT with *frame_size* = 512 and *hop_size* = 128



(c) STFT with each cylinder firing event hand labelled. There were 28 counted combustion events in the 1 second audio segment.

Figure 3.2: Time and Time-Frequency visualisations of engine sound recorded on a smartphone

(*Equation 2.1*) of the engine recording. *Figure 3.2c* counted approximately 28 cylinder fires in one second of audio, or in other words, a $28Hz$ frequency which should be within a smartphone microphone's frequency response.

A DFT was performed on the signal *Figure 3.3* to investigate further whether a frequency component exists that could be a candidate engine firing frequency.

The engine RPM was roughly observed by the vehicle tachometer at 820 RPM. Using *Equation 3.2*, we can estimate that a frequency of $\hat{S} = \frac{820}{60} \cdot 2 \approx 27.3Hz$ should be present with high magnitude in the FFT signal if an audio event is present for each new engine stroke during the recording.

Figure 3.3 shows significant peaks at $28.9Hz$, $57.2Hz$ and $85.8Hz$ with the largest peak at $57.2Hz$. The peak at $28.9Hz$ is very similar to $\hat{S}$ based on the approximate tachometer reading and appears to be the observed fundamental frequency of S. The other large peaks are near perfect multiples of the observed S and can be considered the 2nd and 3rd harmonics of S.

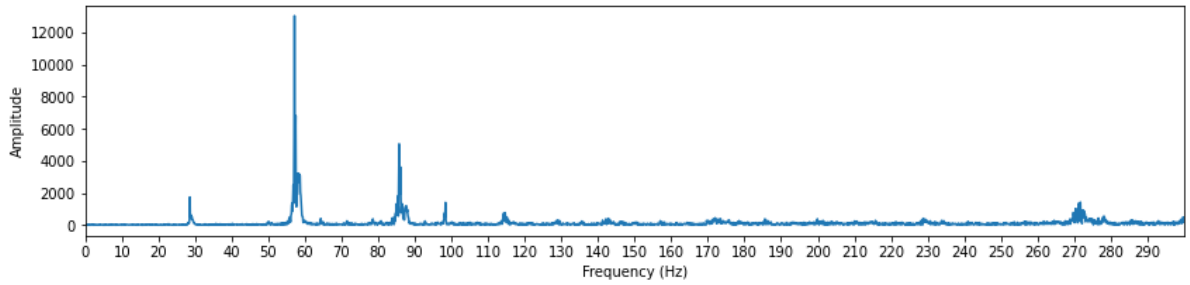


Figure 3.3: FFT between $0 - 300Hz$ of 9 second audio recording of engine idling at 820 RPM.

Harmonic Frequencies

The fundamental frequency is the lowest frequency that is produced by the oscillation of a sound source. The 1st harmonic of a sound is its fundamental frequency, and any subsequent harmonics are multiples of its fundamental [20]. The harmonics produced by an object determine the timbre of its sound.

The fundamental frequency can be the highest magnitude peak, but it is not always the case. For example, in *Figure 3.3* the 2nd harmonic is the most prominent frequency. This may be as a result of the object resonating at a greater magnitude for that particular harmonic, but it may also be due to microphone attenuation of the signal at the fundamental frequency. The frequency response of most smartphones begins to attenuate at frequencies below $50Hz$ and above $10kHz$ [21].

Spectral and Temporal Resolution

The FFT in *Figure 3.3* describes the frequency spectrum for the entire signal. This is useful to provide average RPM information for the period of the recording, but it does not provide any temporal information with regards to RPM at discrete time intervals throughout the recording.

To achieve temporal resolution, an STFT (*Section 2.3.2*) can be used to find the frequencies present over a frame of time. *Figure 3.4* shows an STFT of the recording.

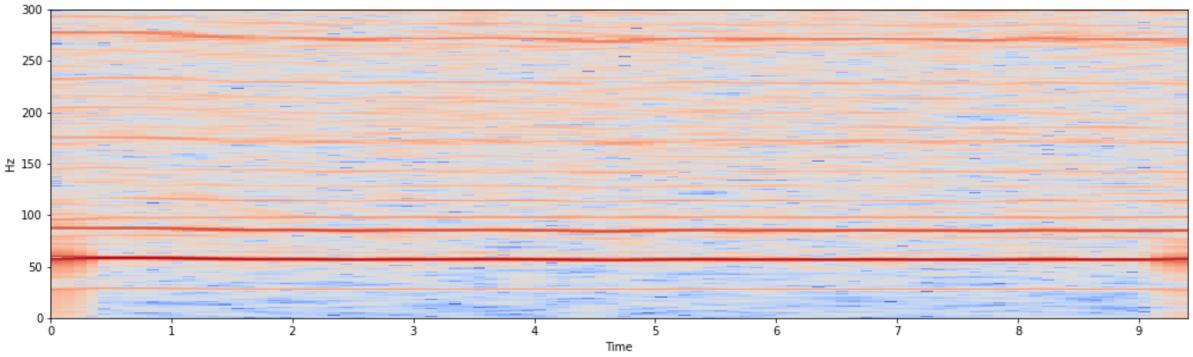


Figure 3.4: STFT of engine recording from $0 - 300Hz$ using a *Hann* window, frame size of 48,000 samples (one second) and hop size of 4800 ($\frac{1}{10}$ second).

High spectral resolution is required for this application as a difference of 100 RPM is reflected by a spectral change of only $3.3Hz$. Achieving spectral resolution of $1Hz$ per frequency bin can measure a difference of 30 RPM.

It should also be considered that increased frame size comes at a cost to temporal resolution. Increasing the window overlap can help to restore some temporal resolution as the same signal is considered across several STFT frames. This does come at a cost to computational performance. Applying an FFT over a larger frame size for a particular point also averages the signal, causing changes in pitch to propagate more slowly.

Figure 3.4 uses a frame size of 48,000 samples. This provides a frequency resolution of $1Hz$ per bin. Each frame is one second in length. To improve this, a hop length of 4800 samples is used to improve the time resolution. This results in an estimate of the RPM at $0.1s$ intervals.

3.1.3 Algorithm

From this observation, a relatively simple algorithm can be created to find the fundamental frequency of the engine for each STFT frame X_i.

Peak picking is used, as described by *Equation 2.20*. A height $\gamma = \hat{X}$ was used where $\hat{X}$ is the mean frame frequency magnitude to ensure that local maxima with very small

magnitudes are not included. A distance of $\epsilon = 15$ samples is used, which assumes that the fundamental frequency must be at least $15Hz$ or running at 450RPM. This should be sufficient as most modern four-cylinder vehicles idle above 500 RPM.

The lowest peak found is assumed to be the fundamental frequency of the engine. The RPM can then be calculated for the selected frame by using *Equation 3.1*.

A Python implementation of the algorithm can be found in *Appendix C.1*.

3.2 Detecting Misfires

The method used to detect misfires builds on two priors: 1) there are audible differences from the engine sound that professional mechanics can classify, and 2) misfires affect the crankshaft speed and hence affect the RPM of the vehicle. Therefore, previous work to detect the RPM of an engine from *Section 3.1* can help identify the effect of misfires on it in some way.

3.2.1 Data Collection

This investigation aimed to identify intuitive time-frequency domain features that are present in misfiring engines in real-world situations that are invariant to ambient noise. To ensure that this is the case, a dataset was obtained with vehicles recorded outdoors that were not recorded at any particular fixed position from the engine. All recordings were taken with the engine's bonnet open and with the microphone no more than two metres from the vehicle's radiator.

Data were retrieved from two vehicles, a 2004 Audi TT and a 2006 Nissan Micra Sport, both with inline four-cylinder engines.

First, the engines were recorded at low idle, operating normally. Single-cylinder misfires were then induced on the engines by disconnecting the ignition wiring to one of the cylinders as seen in *Figure 3.5*. Single-cylinder misfires in each of the four cylinders were recorded. Combinations of two-cylinder misfires were also induced, both with even and odd stroke offsets (cylinders 1 and 2, 2 and 3, 3 and 4, 1 and 4).

Three-cylinder misfires were not attempted in this experiment as four-cylinder engines struggle to run at all on just one cylinder and often stall.

Intermittent misfires were also recorded. They were induced by removing the ignition wiring of one of the cylinders for very short periods of time. The number of cycles that the engine intermittently misfired for could not be measured accurately when performing the experiment. A cycle takes approximately 140ms if an engine is idling at 800RPM, and the

Figure 3.5: Inducing a cylinder misfire by disconnecting the coil pack

average human reaction time to auditory stimulus is 235ms [22], so there were most likely at least two misfires in succession.

3.2.2 Audio Analysis

The recordings of cylinder misfires were visualised. *Figure 3.6* shows spectrogram representations of engines with various cylinder misfires. In each of the misfire examples, when the ignition source is removed, a noticeable drop in RPM can be observed. However, the RPM does recover to the vehicle's original idle RPM as the ECU attempts to counteract the drop due to the misfire.

Inharmonic Overtones

An interesting observed difference between the misfiring and non-misfiring engine audio is the presence of inharmonic overtones between the harmonics of the engine stroke component. Note that the number of visible overtones changes depending on which cylinders are misfiring.

For example, *Figure 3.6a* shows that three evenly spaced inharmonic overtones appear between the fundamental frequency and the 2nd harmonic. The same is true between the 2nd and 3rd harmonics.

Likewise, in *Figure 3.6b*, when only a single cylinder is misfiring, there are three evenly spaced overtones between the fundamental frequency and the 2nd harmonic and between the 2nd and the 3rd harmonics. However, when the second misfire in *Figure 3.6b* begins, only one inharmonic overtone is visible.

Recall from *Section 2.1* that four-cylinder engines follow a 1-3-4-2 firing order. In *Figure 3.6b*, cylinder 1 fires between the misfires of cylinder 2 and 3. In *Figure 3.6c*, cylinder 1 misfires adjacent to cylinder 2. When the second misfire begins, there is a noticeable drop in

RPM as in the other examples. However, there is no visible change in the number of inharmonic overtones between harmonics, unlike in previous examples.

A hypothesis was formed to explain the observed spectral phenomenon in misfiring engine signals. Under normal operation, the DFT fits a sine wave at the engine firing frequency. However, if a misfire occurs, it causes a discontinuity in the engine firing signal. This discontinuity results in a less perfect fit at that frequency. Sine waves of differing frequency also fit better to the signal over time. For each cylinder that misfires, there exists a region of discontinuity in the engine firing signal where the cylinder should have fired. The position of these discontinuities also changes depending on which cylinders are misfiring. *Appendix A* discusses in detail, an emulation that was created to test this hypothesis.

Intermittent Misfires

Figure 3.6d shows a spectrogram of an intermittently misfiring engine. It should be noted that on close inspection, the inharmonic overtones are present, but only over a small number of frames. A large frame size is used with a small hop size which causes the spectral information of the misfire to propagate over several frames. While this may reduce the accuracy with which we can identify the time of the intermittent misfire, it helps to visually and algorithmically identify the misfiring regions.

3.2.3 Algorithm

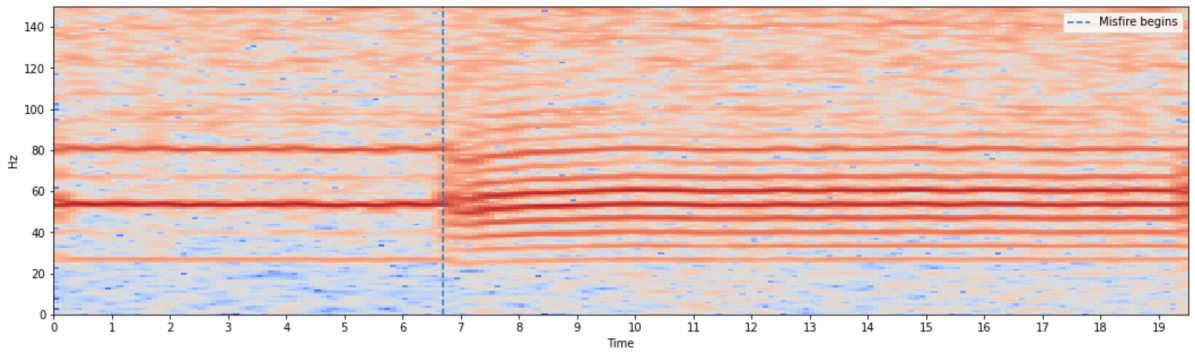
An algorithm to detect misfires was created based on findings from the audio analysis performed in *Section 3.2.2*.

An STFT of the signal is performed with the *framesize* $= 48000$ and the *hopsize* $= 4800$ as shown in spectrogram representations in *Figure 3.6*.

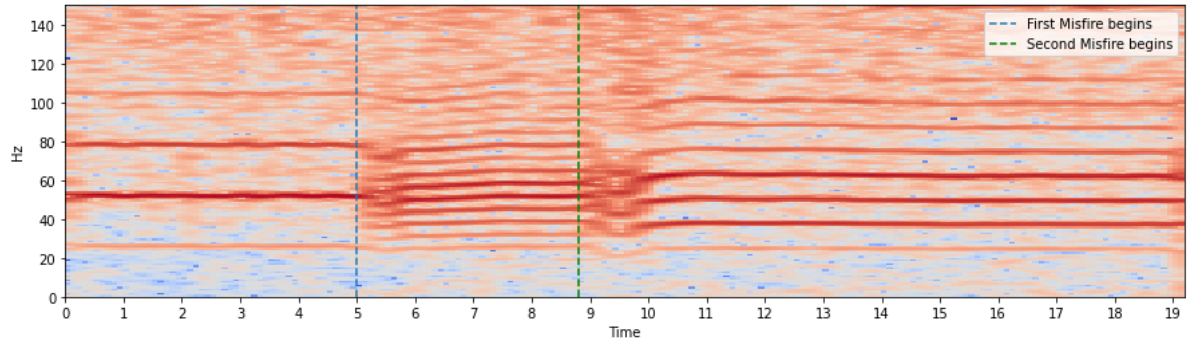
HPS (*Section 2.4.1*) is applied to the STFT signal. A large kernel size $2l_p + 1 = 100$ is used. This was chosen to ensure that only harmonic components with a length of 100 samples or greater are considered in the component. A margin $\beta = 10$ is used to remove noise from the signal.

Peak picking (Section 2.4.2) is performed on the harmonic component. The number of peaks found between $0 - 100Hz$ is used as a feature to determine if there is a misfire. More than three peaks found labels the frame as containing a misfire.

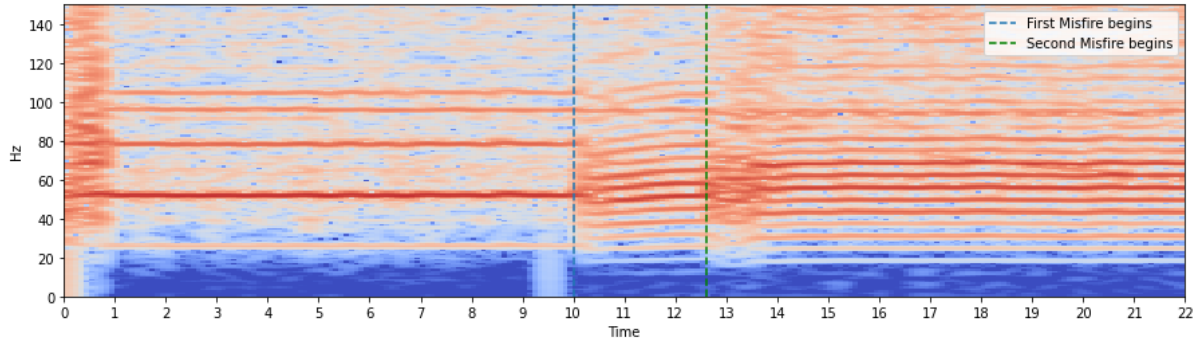
The number of misfire frames present in the signal is then used to determine a signal-level classification. If more than 80% are labelled misfire frames, the signal is labelled as having a constant misfire. If more than 5% of frames but less than 80% are labelled with a misfire, the signal is labelled to have an intermittent misfire. These thresholds were chosen as values that worked well with the test data.



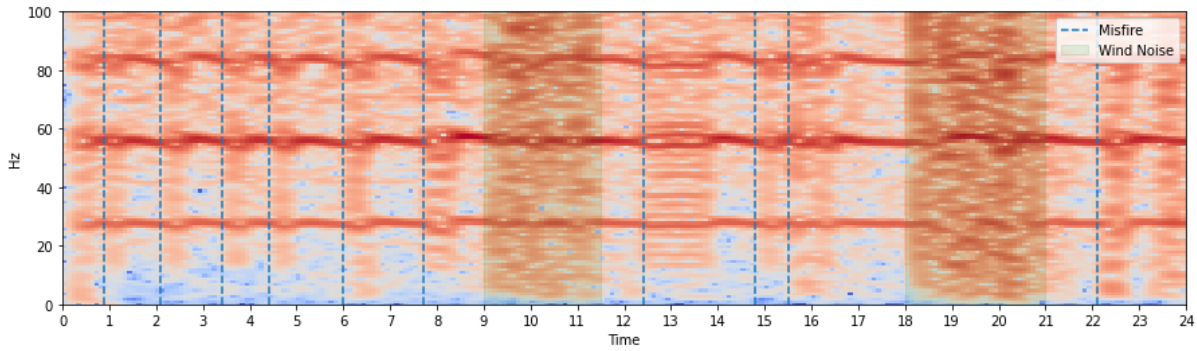
(a) Single Cylinder misfire in Cylinder 2 that starts at approx. 6.7 seconds as shown by the blue line.



(b) Cylinders 2 and 3 misfiring. Cylinder 2 is disconnected at approx. 5 seconds as shown by the blue line and Cylinder 3 at approx. 8.8 seconds as shown by the green line.



(c) Cylinders 1 and 2 misfiring. Cylinder 1 is disconnected at approx. 10 seconds as shown by the blue line and Cylinder 3 at approx. 12.6 seconds as shown by the green line.



(d) Intermittent misfires hand labelled by blue lines. Wind noise is also heavily present in this recording and is denoted by the green highlighted regions.

Figure 3.6: STFT of various cylinder misfires of an Audi TT inline-4 cylinder engine.

A Python implementation of the algorithm can be found in *Appendix C.2*.

3.3 Relative Compression Detection

The method used to compare cylinder relative compression hypothesises that a *'cranking sound'* is produced when the engine is starting that is related to the starter motor in some way. If the starter motor draws less current for a cylinder, it may affect the properties of that sound relative to that of the other cylinders.

The intended use case is for a mechanic or vehicle owner to turn the vehicle over without starting the engine. Some vehicles have a maintenance feature to prevent the car from starting when turning it over by pressing down the accelerator pedal.

3.3.1 Data Collection

Data were retrieved from one vehicle, a 2004 Audi TT with an inline four-cylinder engine. The engine in the Audi TT had over 200,000 miles when the tests were performed. Despite this, the engine compression is still above manufacturer tolerances, and the engine ran with no issues.

Starter motors are designed to turn the engine over only for short periods of time. They draw a large amount of current and discharge the vehicle's battery quickly. To ensure that sufficient power was provided to the motor throughout the data collection process, a garage battery charger was used.

First, the engine was compression tested with a pressure gauge to provide a baseline reading for each cylinder. The recommended approach [11] is to remove all spark plugs and insert the compression tester into each spark plug hole as seen in *Figure 3.7*. The engine was then turned over six times, and the compression was read from the gauge. It should be noted that the pressure gauge only reads the highest compression value in each cylinder rather than the instantaneous pressure. The vehicle was compression tested twice. The results of the test are shown in *Figure 3.1*.

Figure 3.7: Compression testing using a pressure gauge

The compression tests were also recorded and used in analysis to help identify the spectral components corresponding to a cranking event.

Cylinder	Compression Test 1 (psi)	Compression Test 2 (psi)
1	180	183
2	174	178
3	170	175
4	186	190

Table 3.1: Pressure gauge readings of two separate compression tests on the 2004 Audi TT engine.

The engine was then recorded turning over with and without induced low compression in the cylinders. Low compression was induced by inserting the compression tester into the spark plug hole of the cylinder by only a couple of threads. This allowed for air to escape through the spark plug hole. The compression tester also measured the induced drop in compression. The tester was held in place to prevent it from disconnecting during the test.

There were limitations to this method. Obtaining a consistent induced low compression was difficult because the compression tester was frequently dislodged from the spark plug hole due to the cylinder pressure. As a result, in some cylinders, lower induced compression was achieved than in others.

To obtain a reference event of the cylinder 1 compression stroke, the coil pack was removed but remained connected to the ignition wiring. When the engine was turned over, a loud sparking sound was produced that could be easily identified in the spectrogram representation of the audio. This technique can be seen in *Figure 3.8*.

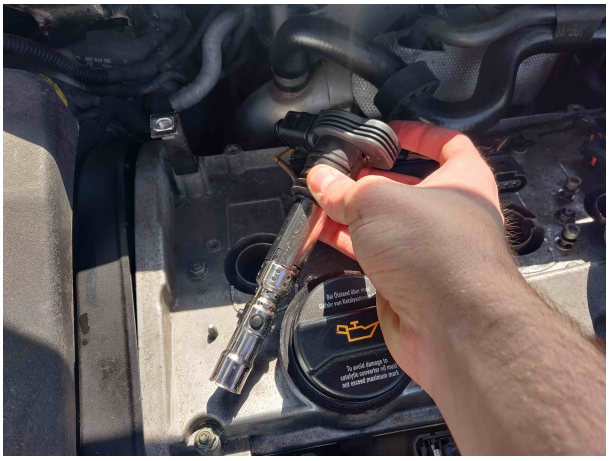


Figure 3.8: Using the spark sound from the cylinder 1 coil pack as a reference event

Finally, recordings were made with no compression induced in cylinders by removing the spark plugs. This was to test whether a distinction could be made between cylinders with low and no compression, and if it could be detected algorithmically.

3.3.2 Audio Analysis

To begin, the recordings from the compression tests were analysed. A mel-spectrogram representation of the signal was used to more easily identify spectral features of interest. *Figure 3.9* shows the mel-spectrogram representation of the audio. In these recordings, only a single cylinder comes under compression. The audio was also listened to, slowed down by 500% and 800%, sequentially. This analysis resulted in identifying the starter motor sound and its harmonics in the spectrogram. They are labelled by hand as blue lines in *Figure 3.9*. Notice that as cylinder 1 comes under compression, the starter motor frequency decreases significantly. When the compression test was being recorded, the vehicle tachometer was observed, and a noticeable drop in engine RPM could also be seen.

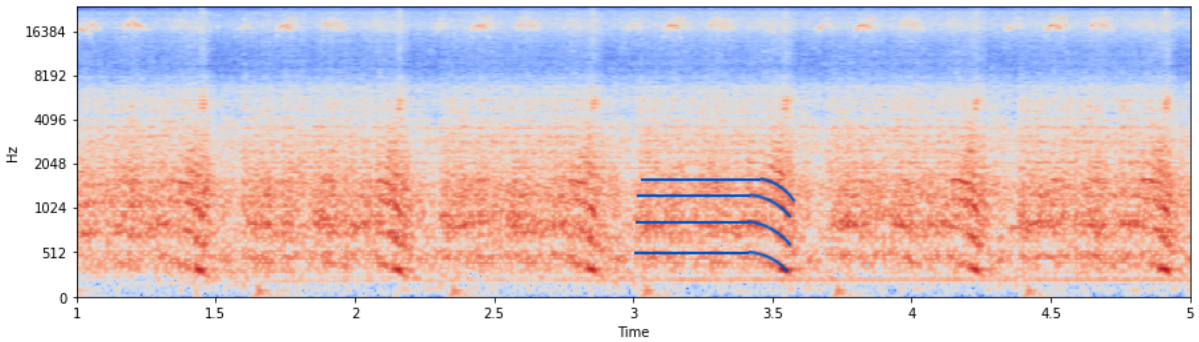
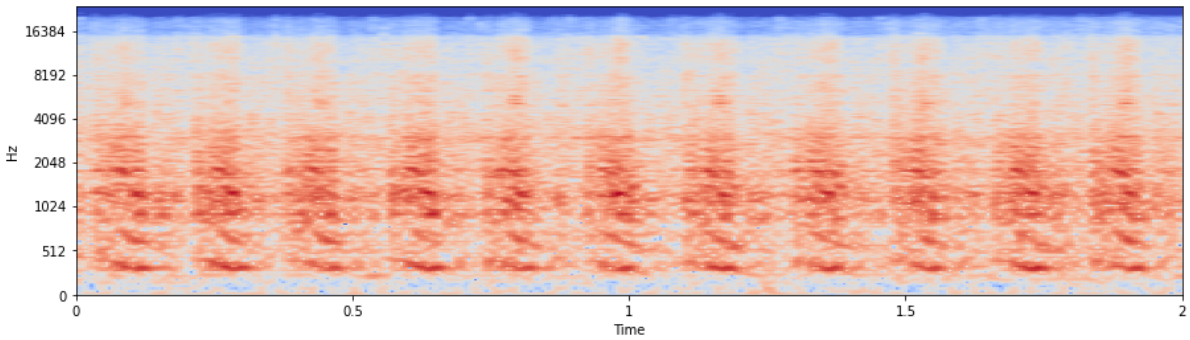


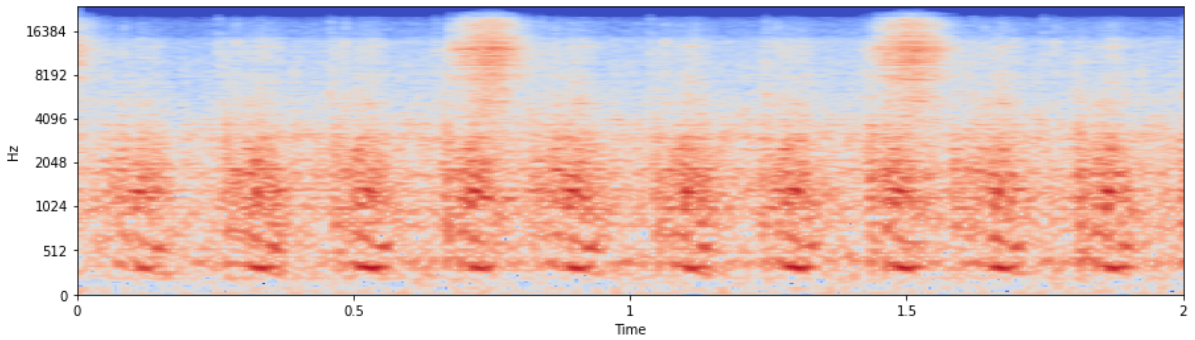
Figure 3.9: A mel-spectrogram representation of a compression test with compression tester inserted on cylinder 1. An STFT with *framesize* = 2048 and *hopsize* = 256 is used. 256 mel bands are applied. Hand drawn blue lines show starter motor spectral features. The frequency of these features decreases as the starter motor comes under load.

The starter motor frequency and its harmonics were also identifiable in the recordings made of the engine turning over as seen in *Figure 3.10*. A drop in the pitch of the starter motor frequency can also be seen as each cylinder comes under compression, although it is more subtle as the starter is always under load.

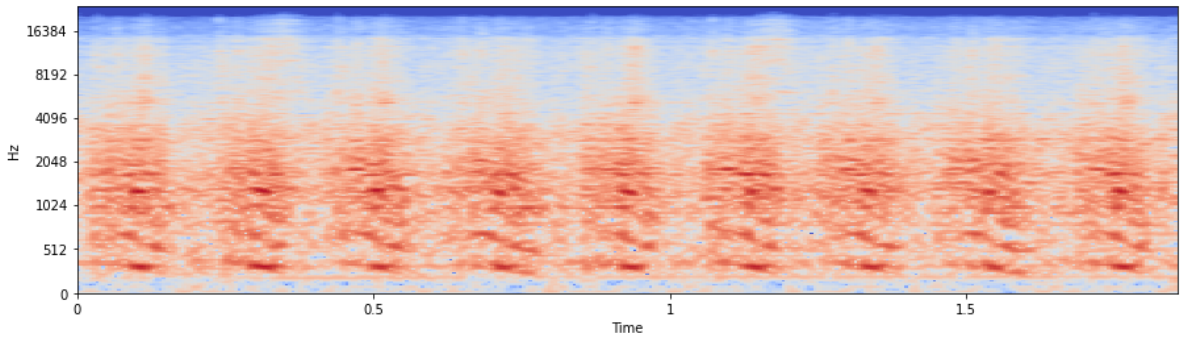
A whistling sound can be heard in the recordings visualised in *Figure 3.10b* and *Figure 3.10c*. The whistling sound is a result of air escaping from the spark plug hole. This sound can also see seen in the spectrogram representations, particularly in the $8000 - 16000Hz$ region in *Figure 3.10b* on the 4th and 8th cranking events.



(a) Baseline cranking with all spark plugs inserted. Compression between $170psi$ and $190psi$.



(b) Induced low compression in cylinder 2 with a measured $140psi$ of compression down from $174psi$ under normal operation.



(c) Induced low compression in cylinder 3 with a measured $135psi$ down from $170psi$ under normal operation.

Figure 3.10: Mel-spectrogram of various relative compression tests.

Autocorrelation

Autocorrelation is the correlation of a signal with a version of itself transformed in the time domain. It is used commonly to find periodic patterns in signals. Relative compression recordings were autocorrelated for all lags and plotted as shown in *Figure 3.11*.

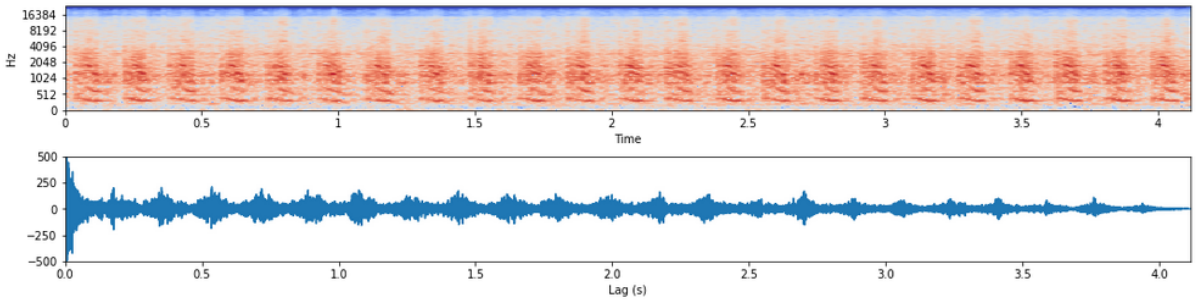


Figure 3.11: An example of the mel-spectrogram of the baseline relative compression signal and its autocorrelation for all lag values.

Autocorrelation is very effective at determining if a signal is periodic. However, it was found that it is not effective at finding a periodic discontinuity in a signal because most periods still align correctly and will only reflect a slight decrease in similarity.

While the signal should be periodic in general, periods that differ in similarity significantly from the others should also be identified. To achieve this, individual cylinder cranking events are segmented and compared in similarity to the signal rather than comparing the entire lagged signal with itself.

3.3.3 Algorithm

In the sections below, details of the algorithm used to detect relative compression differences are discussed in detail. However, first, the algorithm is outlined at a higher level:

1. Split the time-domain signal into short-time five cycle or 20 stroke segments (*Section 3.3.4*).
2. **Segment** each cylinder cranking event (*Section 3.3.5*).
3. For each time-domain segment, calculate the **statistical similarity** of each cylinder (*Section 3.3.6*).
4. Calculate the **self-similarity** of each cylinder from the statistical similarity output (*Section 3.3.7*).
5. Calculate the **pair-wise similarity** of cylinder pairs from the statistical similarity output (*Section 3.3.8*).

6. Use the **mean self-similarity** and the **mean pair-wise similarity** to form a relative compression error value that increases with cylinder dissimilarity and vice versa, decrease with cylinder similarity. This output can then be thresholded based on what is an acceptable level of similarity (*Section 3.3.8*).

3.3.4 Signal Segmentation

The DT-signal is split into short-time segments, 20 cranking events or five engine cycles in length. All statistical comparisons are then calculated for each short-time segment. Small segment sizes are used as it was found that cranking similarity attenuates over long cranking periods. There is a noticeable change in the engine's RPM between the start and end of recordings, where the engine is turned over for long periods. It is believed that this occurs due to battery depletion. As a result, this affects the cranking event similarity. Considering 20 cranking events allows for several cycles to be considered to reduce the effect of anomalous noise that may occur without significant effects of the similarity attenuation over time.

The following steps in the algorithm can be considered as being applied to a single short-time segment and repeated for each segment.

3.3.5 Crank Segmentation

Each crank of the engine produces a somewhat spectrally percussive sound. These sounds appear in a spectrogram representation as a vertical impulse, spread across the entire frequency spectrum.

HPS (*Section 2.4.1*) is applied to the STFT signal. A separation factor $\beta = 1$ is used. The separation factor is typically used to prevent harmonic components from leaking into the percussive component and vice versa. However, while engine cranking events contain significant percussive sound, they also contain a harmonic component from the starter motor. Because of this, leaking harmonic components into the percussive component is advantageous when attempting to segment cranking events.

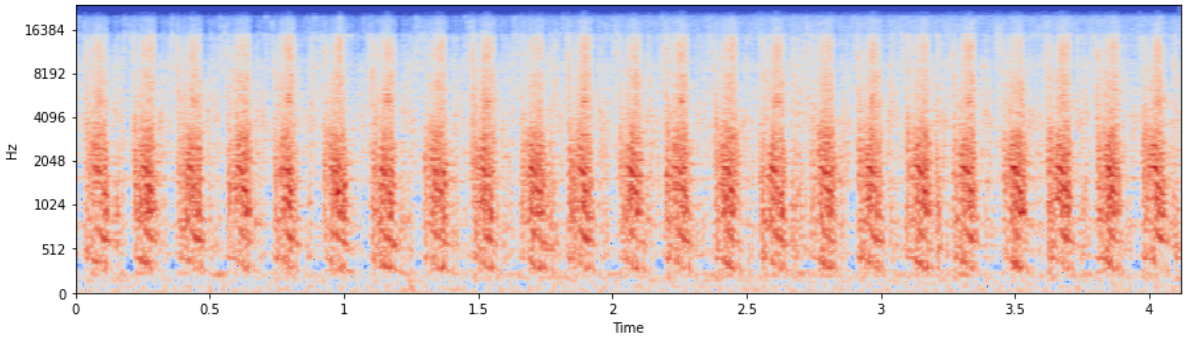
The RMS energy (*Section 2.3.5*) is calculated for each frame of the percussive component. The result is a Short-time RMS energy $(X_p)_{RMS}$ signal shown in *Figure 3.12b*.

Peak picking (*Section 2.4.2*) is used with a large distance $\gamma = 0.1s$ to identify the local maxima for each cranking event. A large γ was used to ensure that only one peak is found for each cranking event. Most starter motors will not spin the crankshaft at more than 300 RPM or 10Hz. The peaks are shown in *Figure 3.12b* as vertical dotted lines. They are then used as anchor points to estimate the start and end samples of the cranking event.

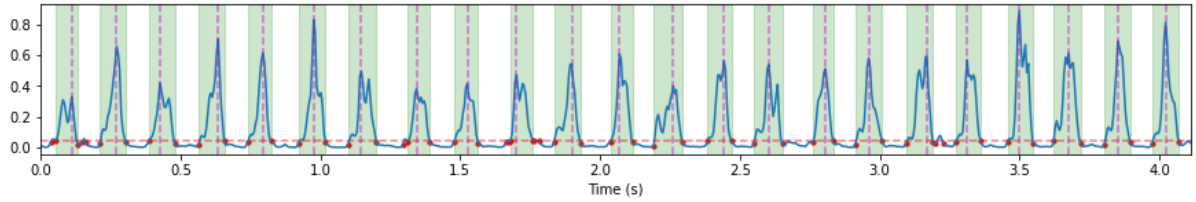
Finally, a magnitude threshold is calculated as the mean of $(X_p)_{RMS}$ multiplied by a constant.

The first sample to the left of the local maxima of each cranking event with a magnitude less than the threshold is chosen as the start of the event. Similarly, the first sample to the right of the local maxima is chosen as the end of the event. The threshold is shown as the horizontal line in *Figure 3.12b*. The result is a start sample s and end sample e for each cranking event $(s_0, e_0)...(s_{n-1}, e_{n-1})$ where n is the number of cranking events.

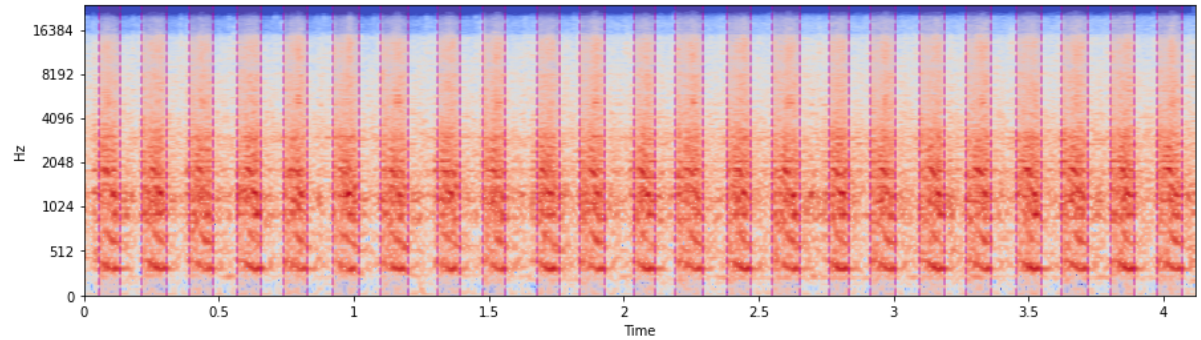
Initially, energy-based novelty detection (*Section 2.4.3*) was used to find the onset of engine cranking events. However, it was found that event onsets were triggered by noise between cranking events. Using HPS reduces the noise between the segments.



(a) Percussive component X_p of HPS with $\beta = 1$ and $2l_p + 1 = 100$ on mel-spectrogram shown in *Figure 3.10a*.



(b) Short-time RMS energy of X_p. The frames with the highest magnitude per crank are calculated by performing peak picking and are marked with vertical dotted lines. A large distance between peaks and a high height threshold is used for peak picking. The horizontal dotted line shows the magnitude threshold chosen to select the start and end frames for each crank segment.



(c) Mel-spectrogram of engine turning over with each cranking event segmented.

Figure 3.12: Visualisations of the crank segmentation process

A Python implementation of this algorithm can be found in *Appendix C.3.1*.

3.3.6 Statistical Similarity

Section 3.3.5 provides labelled samples for the start and end of each cranking event. These labels can be used to define a template for each cranking event. Each template is cross-correlated over the DT-signal to determine its similarity with every other cranking event.

Pre-processing

Before applying cross-correlation, the DT-signal was pre-processed. A bandpass filter is applied to the signal to remove any spectral noise or unwanted features. By considering the recordings shown in *Figure 3.10*, information above 2,000Hz and below 200Hz can be discarded as the regions do not contain visually identifiable spectral features of interest. These are features that represent the starter motor sound for a cranking event. Three bandpass regions were considered between $200 - 400Hz$, $450 - 750Hz$ and $200 - 750Hz$. These regions contained the fundamental frequency, 2nd harmonic and both the fundamental and 2nd harmonic of the starter motor, respectively. These regions are highlighted in *Figure 3.13*. $200 - 400Hz$ was chosen as it was found that, when used, it produced the highest self-similarity. The self-similarity metric is discussed in more detail in *Section 3.3.7*.

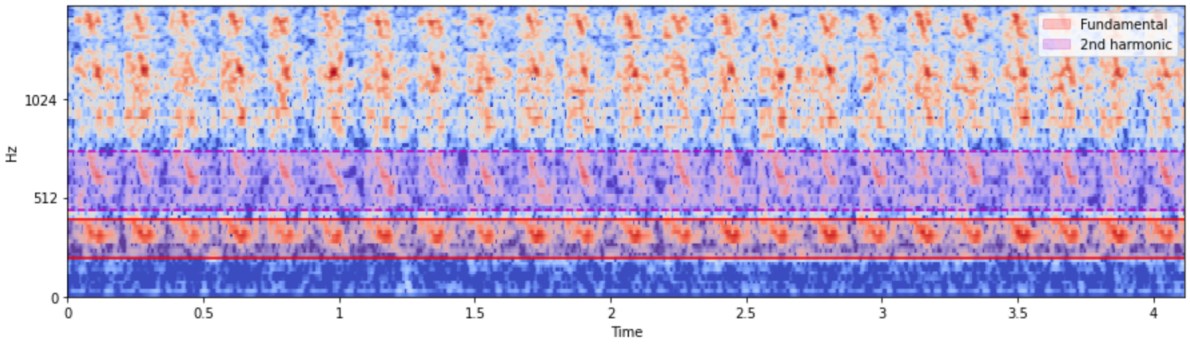


Figure 3.13: A mel-spectrogram of an engine turning over under normal operation. The fundamental frequency of the starter motor sound is highlighted in red and the 2nd harmonic in magenta.

Cross-Correlation

Pearson normalised cross-correlation (*Eqn 2.12*) is applied to the filtered DT-signal. Each cranking event is cross-correlated with the DT-signal generating a matching space $M(i, l)$ for each cranking event i and lag l. $M(i, l)$ for all values of i and l are visualised in *Figure 3.14*. The peaks in the matching space are the points where the template event best fits other cranking events in the signal.

To identify each of these samples of best fit, the local maxima are found for each cranking event template, defined by:

$$P(i,p) = \max_{s_p \leq j \leq e_p} M(i,j) \tag{3.3}$$

where

i is the cranking event

p is the index of the cranking event to compare

s is the starting sample of each cranking event

e is the end sample of each cranking event

It can be observed that $P(i,i) = 1.0$. The results of $P(i,p)$ for all values of i p are shown in *Figure 3.14* as red points.

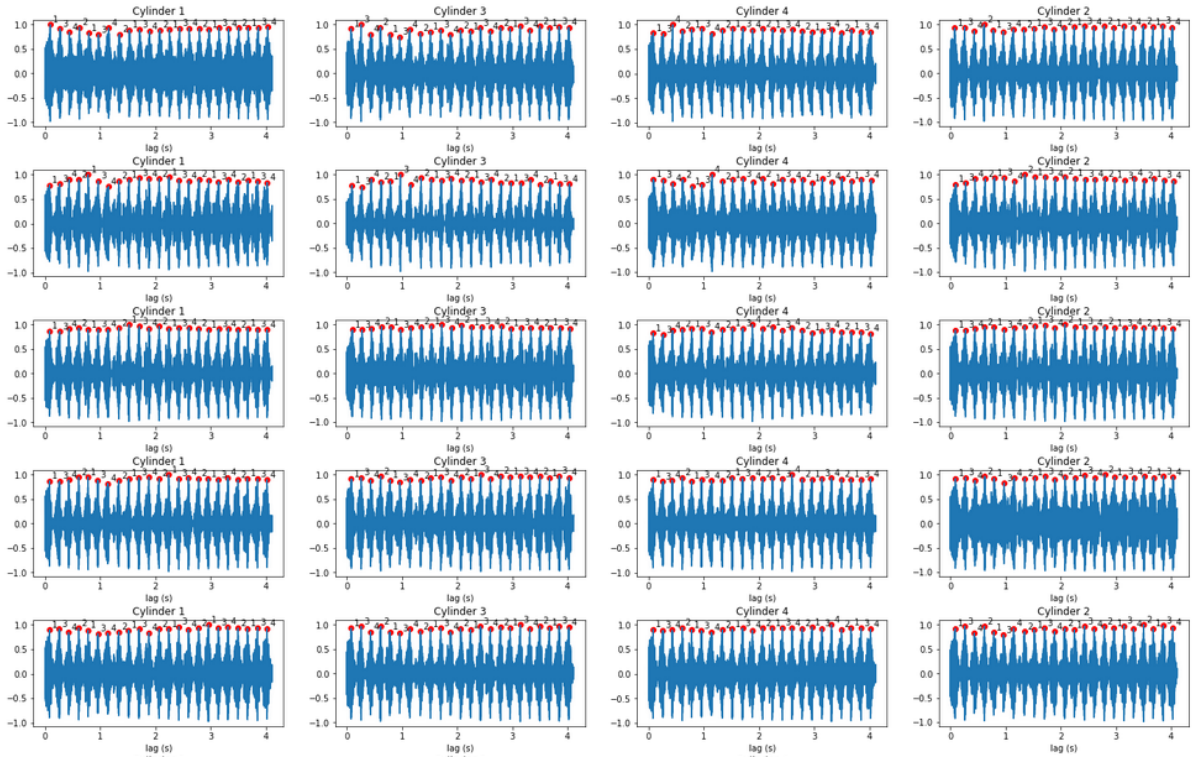


Figure 3.14: Examples of cross-correlation $M(i,I)$ applied to the baseline signal shown in *Figure 3.10a* with templates of each cylinder crank applied over the signal. Red peaks show the points where the template best matches with subsequent cranking events $P(i,p)$ and are labelled by cylinder in the firing order.

A Python implementation of this algorithm can be seen in *Appendix C.4*.

3.3.7 Cylinder Compression Self-Similarity

Self-similarity is calculated as the mean average similarity of each cranking event with all other cranking events of the same cylinder for a short-time segment. Intuitively, engines should crank with approximately the same compression with each crank. While there is some variability between cranks, this should be within the margin of error and considered normal operation. Self-similarity describes the margin of error so that it can be used as a frame of reference when comparing cranks from different cylinders.

Formally, the self-similarity of a cylinder crank for a cranking event i can be defined as:

$$P_{self}(i) = \frac{1}{|m_i| - 1} \sum_{j \in m_i, j \neq i} P(i, j) \tag{3.4}$$

where

$$m_i = \{x \in \mathbb{Z} \mid 0 \leq x < n, x \equiv i \pmod 4\} n \text{ is the number of cranking events (20)}$$

To find the cranking event that best describes the cylinder cranks, the cranking event i with the highest self-similarity for each cylinder $c = [0, 3]$ is calculated and used as the reference cranking event for that cylinder. This is defined by:

$$S_{self}(c) = \max_{j \in m_c}(P_{self}(j)) \tag{3.5}$$

Similarly, the cranking event index with the highest self-similarity for a cylinder can be found by:

$$I_{self}(c) = \operatorname{argmax}_{j \in m_c}(P_{self}(j)) \tag{3.6}$$

A Python implementation of this algorithm can be found in *Appendix C.5*.

3.3.8 Pair-wise Similarity

A pair-wise comparison of cranking events for cylinders (s, d) : $(0, 2)$, $(0, 3)$, $(1, 2)$ and $(1, 3)$ is performed. It is defined by:

$$S_{pair}(src, dst) = \max_{j \in m_{dst}, x \neq src} P(src, j) \tag{3.7}$$

where

$$src = I_{self}(s)$$
$$dst = d$$
$$m_{dst} = \{x \in \mathbb{Z} \mid x0 \leq x < n, x \equiv dst \pmod 4\}$$

Note that the cranking event with the highest self-similarity $I_{self}(s)$ is used as the source cylinder cranking event.

The maximum is selected here to attempt to emulate what is measured by a pressure gauge. A pressure-based compression tester does not show instantaneous pressure, but rather the highest compression value over many strokes. The idea here is the same, the highest similarity over a number of strokes is captured.

The self-similarity for the *src* cylinder of each pair is then subtracted by the pair-wise similarity value for each cylinder to provide a similarity difference between the baseline, self-similarity and the pair-wise cylinder similarity:

$$X(src, dst) = S_{self}(src) - S_{pair}(src, dst) \tag{3.8}$$

$X(src, dst)$ provides a relative compression error value. It approaches zero if the pair-wise similarity matches that of the self-similarity. If the pair-wise similarity is greater than that of the self-similarity, the value becomes negative. It can occur in reality that the compression between cylinders happens to align better. The value is rectified such that any negative values have a relative compression error of zero.

A Python implementation of this algorithm can be found in *Appendix C.6*.

3.3.9 Detecting No Compression

When there is no compression in a cylinder, the starter motor does not come under load for that cylinder's stroke. When this occurs, there is essentially no cranking event. As a result, the crank segmentation algorithm often incorrectly segments the cylinder with no compression and the subsequent cylinder cranking event as a single event. This can be observed in *Figure 3.15*.

This predictable error in crank segmentation can be used to detect when there is no compression in a cylinder by observing two measurable features of crank segments, 1) cranking event **start positions** and 2) cranking event **segment sizes**.

The mean distance between the start of each cranking event and its subsequent cranking event is calculated. A threshold of 30% variance from the mean is then set as the tolerance

for all cranking events. Any cranking events with a distance outside of the threshold classify the recording as having a no compression cylinder.

Similarly, the mean segment size is calculated, and a variance of 40% of the mean was used to threshold cranking events of cylinders with no compression.

Thresholds were determined by first measuring the mean and variance for both the start positions and segment sizes. This was done for recordings of healthy engines and engines with low compression. The variance was observed for the recording of the engine with no compression. The chosen thresholds were found to separate the low compression examples from the no compression examples.

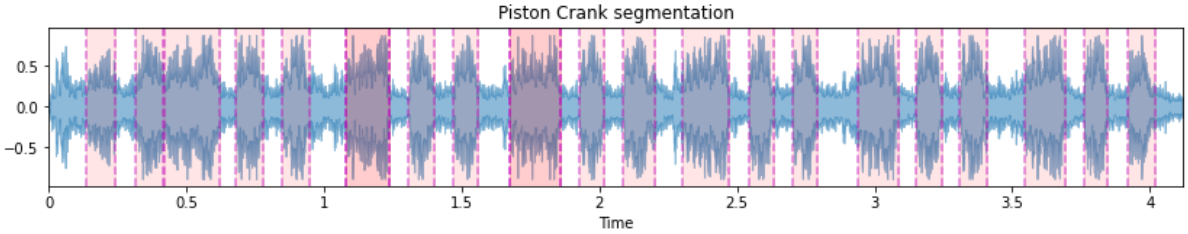


Figure 3.15: DT-signal of engine turning over with no compression induced in one cylinder. Crank segments are shown as coloured regions.

A Python implementation of this algorithm can be found in *Appendix C.7*.

4 Evaluation

Each engine fault detection method was implemented in a library I created called the Engine Audio Recognition Library (EARL). The Python implementation of the algorithms used in EARL can be found in *Appendix C*. This chapter presents the results used to evaluate the implementation of the methods described in *Chapter 3*.

4.1 RPM and Misfire Detection

4.1.1 No Misfires

To test the robustness and accuracy of the RPM detection and misfire detection algorithms, recordings were made of a 2004 Audi TT, 2006 Nissan Micra Sport, 2017 Dacia Duster and 2014 Skoda Superb idling under normal operation. All vehicles tested were fitted with inline four-cylinder engines. The Dacia and Skoda were diesel, and the Audi and Nissan were petrol variants. The recordings were made in different locations indoors, outdoors and inside the vehicle to test for microphone placement robustness.

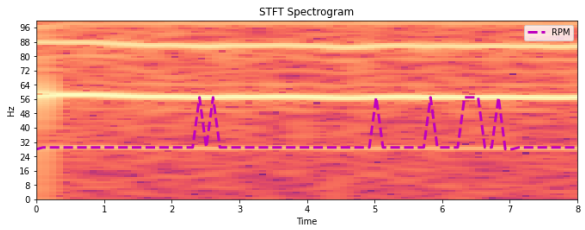
The RPM and a misfire label were determined for every STFT frame in each recording. Test result visualisations are shown in *Figure 4.1*.

The correct vehicle RPM was found in all examples. However, there are a couple of interesting edge cases where the fundamental engine frequency was matched to the wrong frequency bin:

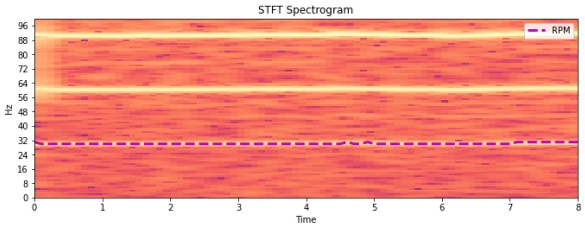
- **Low fundamental frequency magnitude** - *Figure 4.1a* shows a case where the fundamental frequency of the engine is too low in places to be included in the harmonic component of the HPS of the signal. Instead, the 2nd harmonic is chosen as the RPM, causing a sudden spike in the RPM value. It was found from the tests that the fundamental frequency of the engine is more present in recordings taken inside the vehicle, such as in *Figure 4.1d*. This makes intuitive sense as the engine tone often sounds lower inside the vehicle than outside.
- **Wind Noise** - *Figure 4.1e* was recorded with strong winds outdoors. It is clear from

the spectrogram representation that the wind noise occludes engine frequencies and other spectral information significantly. There are several frames where lower frequency bins have high magnitudes due to the wind noise and cause the RPM value to spike suddenly.

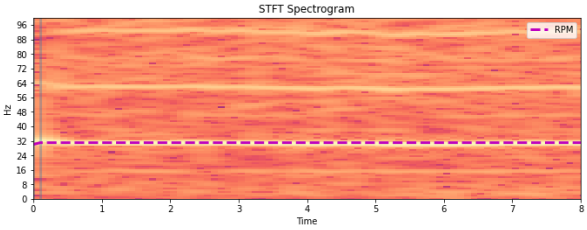
The wind noise also caused misfire detection false positives as seen in *Figure 4.1e*. Two frames are also classified as misfire frames in *Figure 4.1d*. However, 5% of frames are required to be classified as misfire frames in order to classify the recording as containing intermittent misfires. This resulted in the correct, true negative classification.



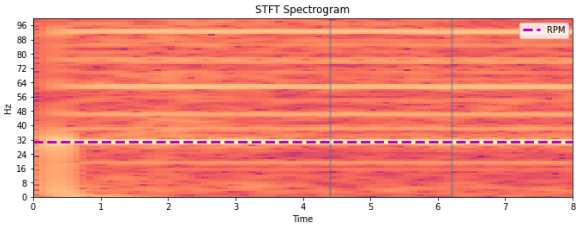
(a) 2004 Audi TT recorded indoors



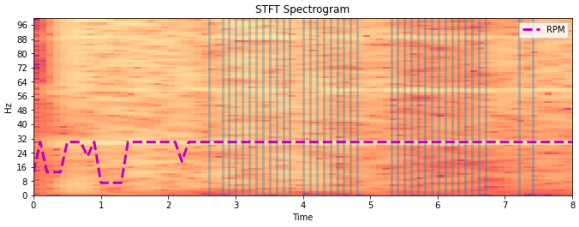
(b) Dacia Duster recorded outdoors



(c) Nissan Micra recorded outdoors



(d) Skoda Superb recorded inside the vehicle



(e) Skoda Superb recorded outdoors on a windy day. Misfire frames are shown as faint vertical blue lines.

Figure 4.1: Vehicles recorded operating normally at idle

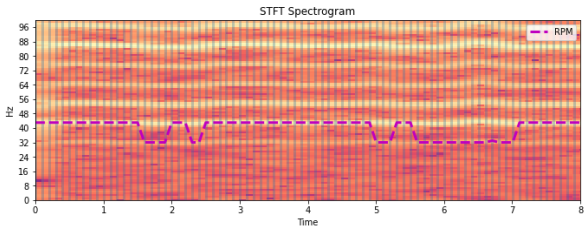
4.1.2 Single Cylinder Misfires

To test the misfire algorithm for true positives and false negatives, the 2004 Audi TT and 2006 Nissan Micra were induced with misfires in various cylinders, and the results were observed. A third-party audio clip obtained via YouTube of a Volkswagen Passat 1.9 TDI with a single-cylinder misfire was also classified. The third-party recording used an unknown microphone several feet from the vehicle, indoors, to record the engine idling. The single-cylinder results are shown in *Figure 4.2*.

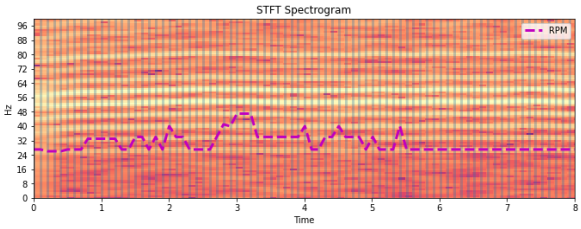
At least 80% of frames must be classified as misfires to classify the recording as containing a constant misfire. The tests were very effective, with at least 93% of the controlled test frames being classified as misfires.

The Volkswagen Passat recording in *Figure 4.2f* was an example of sub-optimal recording conditions. It contained talking, the engine was not idling at all times during the clip, and an unknown microphone was used. 51% of frames were classified as containing misfires. This classified the vehicle as having an intermittent misfire, which, while not strictly correct, considering the circumstances of the recording is an auspicious result.

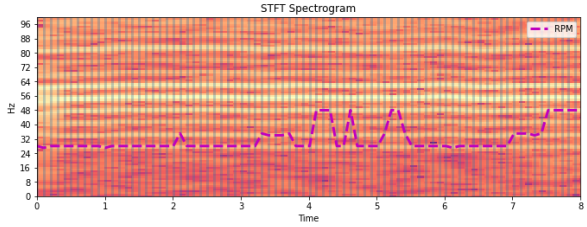
RPM fluctuates considerably throughout these recordings. The RPM detection algorithm struggles to find the fundamental engine frequency. When the engine begins misfiring, sub-harmonic frequencies begin to appear in the signal.



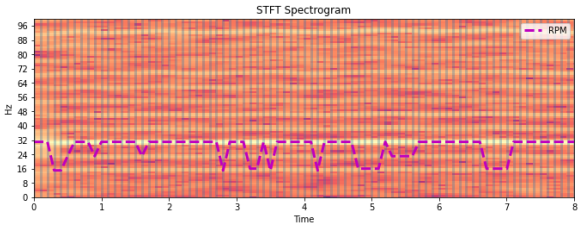
(a) Audi TT cylinder 1 - 97% of frames classified as misfires



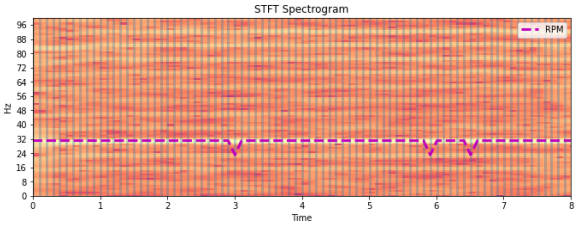
(b) Audi TT cylinder 2 - 100% of frames classified as misfires



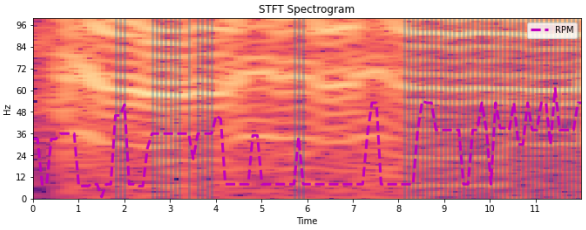
(c) Audi TT cylinder 3 - 100% of frames classified as misfires



(d) Nissan Micra cylinder 1 - 100% of frames classified as misfires



(e) Nissan Micra cylinder 1 - 93% of frames classified as misfires

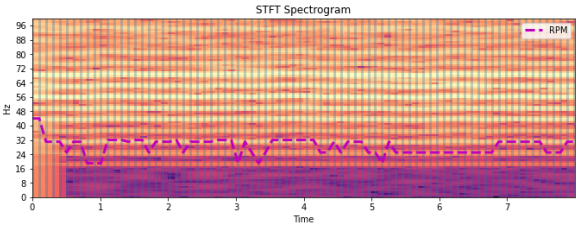


(f) Volkswagen Passat third-party recording of single-cylinder misfire - 51% of frames classified as misfires

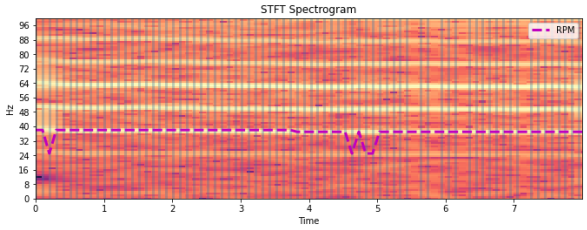
Figure 4.2: Vehicles with single-cylinder misfires

4.1.3 Two Cylinder Misfires

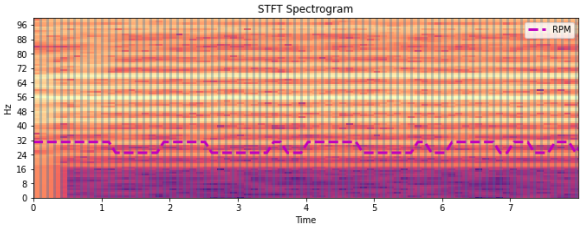
The same test performed in *Section 4.1.2* was repeated with pairs of cylinder misfires. A minimum of 95% of frames were classified as misfire frames. The RPM detection algorithm fluctuated less with two-cylinder misfires, particularly when even offset cylinders were misfiring, such as cylinder 2 and 3. The test results are shown in *Figure 4.3*.



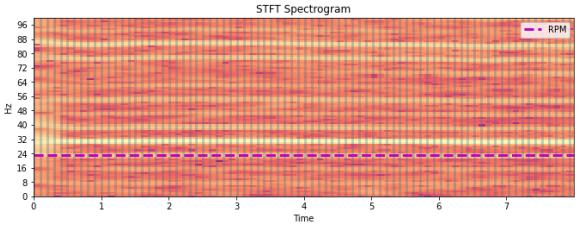
(a) Audi TT cylinder 1 and 2 - 100% of frames classified as misfires



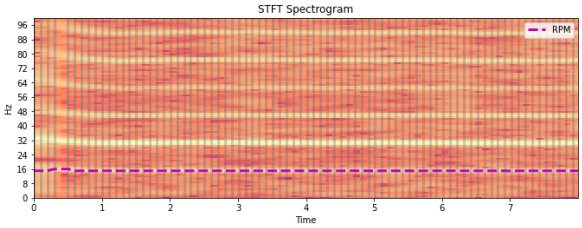
(b) Audi TT cylinder 2 and 3 - 95% of frames classified as misfires



(c) Audi TT cylinder 3 and 4 - 100% of frames classified as misfires



(d) Nissan Micra cylinder 1 and 2 - 100% of frames classified as misfires



(e) Nissan Micra cylinder 2 and 3 - 99% of frames classified as misfires

Figure 4.3: Vehicles with two-cylinder misfires

4.1.4 Intermittent Misfires

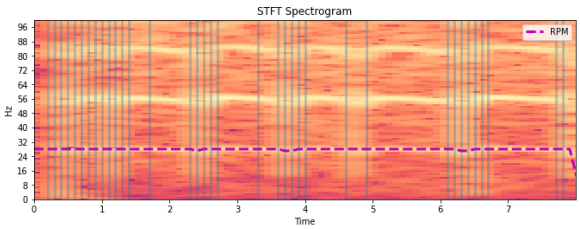
Intermittent misfires were tested in two ways: 1) by inducing intermittent misfires, and 2) using engines with organic intermittent misfire issues.

A 2004 Audi TT was used as the testing vehicle for detecting induced misfires. The results can be seen in *Figure 4.4a*. Although not perfect, the intermittent misfire regions all contain several misfire labelled frames.

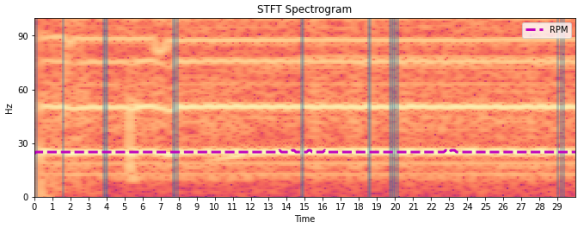
An intermittent misfire fault was also found organically in a 2002 Audi TT. The misfire occurred very intermittently, sometimes several times within a matter of seconds and other

times not for many minutes. *Figure 4.4b* shows the algorithm detecting the misfiring regions. The recording was also taken from inside the vehicle.

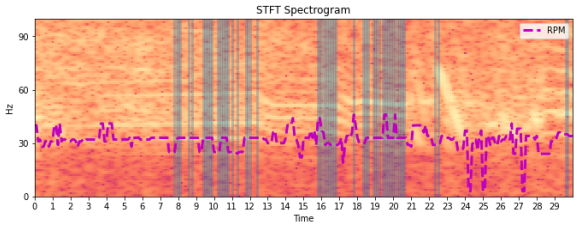
Finally, a recording of an intermittent misfire caused by an engine fault in a Mazda CX-5 was tested. The recording was taken in sub-optimal conditions, inside the vehicle while it was being driven. The issue caused the engine to misfire for several seconds at a time before continuing to operate normally. *Figure 4.4c* shows the algorithm successfully detecting the misfiring regions caused by the engine fault.



(a) 2004 Audi TT induced intermittent misfire - 42% of frames classified as misfires



(b) 2002 Audi TT intermittent misfire - 11% of frames classified as misfires



(c) Mazda CX-5 intermittent misfire - 21% of frames classified as misfires

Figure 4.4: Vehicles with intermittent misfires

4.2 Relative Compression Detection

The measure of relative compression error was tested in two vehicles, a 2004 Audi TT and a 2014 Skoda Superb. The Audi TT was induced with low compression to varying degrees in each cylinder.

The Skoda Superb is a diesel variant vehicle. Compression testing equipment for diesel engines was unable to be obtained for the duration of the project. Because of this, compression testing or inducing low compression in its cylinders was not possible. However, it is a vehicle with low mileage (150,000 km) and good overall health. It should serve as a reasonable real-world baseline for the results.

In each case, the sound of the engine cranking was recorded. The self-similarity (*Table 4.1*), pair-wise similarity (*Table 4.2*) and relative compression error (*Table 4.3*) were calculated for each recording.

4.2.1 Self-Similarity

The calculated pair-wise similarity results are detailed in *Table 4.1*. The *'Audi Baseline'* and *'Skoda Baseline'* recordings represent the self-similarity of each cylinder with no induced low compression. The Skoda self-similarity is noticeably lower and has a higher variance between cylinders than that of the Audi's results. This may occur because:

- The crank segmentation may not be as consistent. This would result in unwanted noise being considered within each cranking event segment, reducing the similarity score.

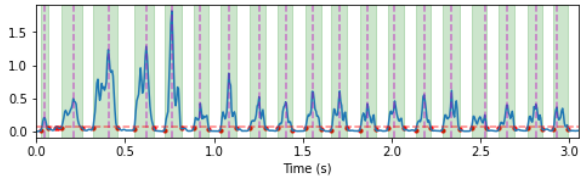
  To test the crank segmentation consistency, the cranking event segmentation for each recording was visualised, as seen in *Figure 4.5*. It can be observed that there is more inconsistency in cranking event segment sizes for the recording of the Skoda (*Figure 4.5b*).

- There may be more noise generated by other engine components in the spectral region considered for self-similarity comparison ($200 - 400hz$).

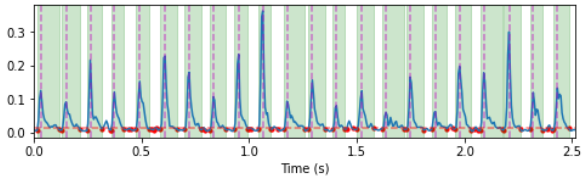
Self-similarity remains high for tests with induced low compression in cylinders. There is a noticeable difference in the self-similarity of cylinder 3 and cylinder 4 in the *'Audi Cyl 3 (110psi)'* and *'Audi Cyl 4 (70psi)'* results, respectively. These are the cylinders in which the low compression was induced.

Test	Cyl 1	Cyl 2	Cyl 3	Cyl 4
Audi Baseline	0.92	0.96	0.94	0.91
Skoda Baseline	0.79	0.72	0.82	0.74
Audi Cyl 1 (145psi)	0.93	0.94	0.91	0.93
Audi Cyl 2 (110psi)	0.91	0.96	0.92	0.94
Audi Cyl 3 (110psi)	0.93	0.95	0.83	0.94
Audi Cyl 4 (70psi)	0.96	0.96	0.96	0.79

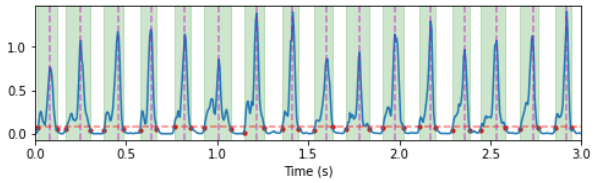
Table 4.1: A comparison of the self-similarity values for each cylinder for different audio recording tests



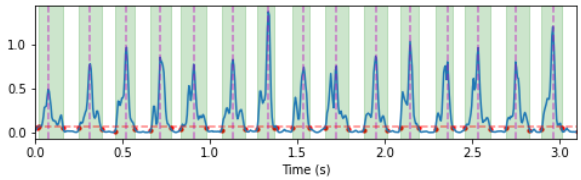
(a) Audi TT baseline



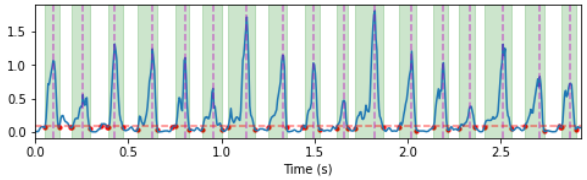
(b) 2014 Skoda Superb baseline



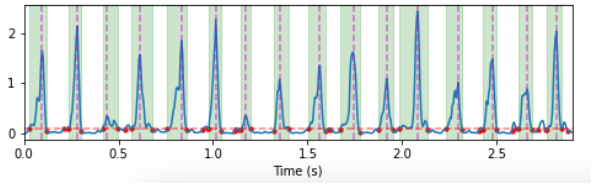
(c) Audi TT Cylinder 1 (145psi)



(d) Audi TT Cylinder 2 (110psi)



(e) Audi TT Cylinder 3 (110psi)



(f) Audi TT Cylinder 4 (70psi)

Figure 4.5: Time-Domain Crank Segmentation for each recording

4.2.2 Pair-wise Similarity

The calculated pair-wise similarity results are detailed in *Table 4.2*. Both baseline recordings show high similarity between cylinder pairs. The pair-wise similarity of *'Cyl 1 (145psi)'* is also high. It should be noted that 145psi is within the tolerated compression difference of the Audi TT engine and should not be classified as having low relative compression. All other recordings show significant drops in the pair-wise similarity for corresponding cylinder pairs.

Test	Cyl (1, 3)	Cyl (1, 4)	Cyl (2, 3)	Cyl (2, 4)
Audi Baseline	0.96	0.95	0.98	0.95
Skoda Baseline	0.87	0.89	0.89	0.88
Audi Cyl 1 (145psi)	0.87	0.92	0.86	0.94
Audi Cyl 2 (110psi)	0.79	0.69	0.83	0.88
Audi Cyl 3 (110psi)	0.78	0.61	0.56	0.85
Audi Cyl 4 (70psi)	0.90	0.53	0.78	0.66

Table 4.2: A comparison of the pair-wise similarity values for different audio recording tests

4.2.3 Relative Compression Error

A relative compression error value for each cylinder pair was calculated. The results are shown in *Table 4.3*. It can be observed that the baseline recordings either have very small or negative relative compression errors. Negative values indicate that the pair-wise similarity was larger than that of the self-similarity of the source cylinder. These negative values can be interpreted as having a very low relative difference. The relative error can be observed to increase with lower induced compression. A relative compression error of 8%, 22%, 39% and 44% can be seen respectively in recordings *'Audi Cyl 1 (145psi)'*, *'Audi Cyl 2 (110psi)'*, *'Audi Cyl 3 (110psi)'* and *'Audi Cyl 4 (70psi)'*.

Test	Cyl (1, 3)	Cyl (1, 4)	Cyl (2, 3)	Cyl (2, 4)
Audi Baseline	-0.04	-0.03	-0.03	0.00
Skoda Baseline	-0.07	-0.10	-0.16	-0.16
Audi Cyl 1 (145psi)	0.06	0.01	0.08	0.00
Audi Cyl 2 (110psi)	0.11	0.22	0.13	0.09
Audi Cyl 3 (110psi)	0.32	0.16	0.39	0.10
Audi Cyl 4 (70psi)	0.07	0.44	0.08	0.29

Table 4.3: A comparison of the calculated relative compression error for different audio recording tests

Table 4.4 shows the absolute compression difference between cylinders based on the compression test results from *Table 3.1*. It can be seen that the relative compression error for a cylinder pair does not match that of the absolute recorded compression difference between each cylinder pair. This could be due to the compression test result only showing the maximum compression in the cylinder, when in reality, the compression varies more over the duration of the test. It could also be due to segmentation inconsistency, as mentioned previously.

It can be concluded that while there is a roughly linear correlation between the calculated relative compression error and the low compression induced, the pair-wise similarity results do not accurately provide an absolute compression difference between the cylinder pairs. However, a significant error for any cylinder pair can be used to indicate that there is low compression in at least one cylinder.

Test	Cyl [1, 3] (psi)	Cyl [1, 4] (psi)	Cyl [2, 3] (psi)	Cyl [2, 4] (psi)
Audi Baseline	10	6	4	12
Audi Cyl 1 (145psi)	35	41	4	12
Audi Cyl 2 (110psi)	10	6	60	76
Audi Cyl 3 (110psi)	70	6	64	12
Audi Cyl 4 (70psi)	10	110	4	100

Table 4.4: A comparison of the calculated compression difference measured in psi

5 Discussion

5.1 System Strengths and Limitations

Chapter 4 presents results found by testing the system on several vehicles. It interprets the results and provides context as to how well the system works. In this section, the strengths as well as some of the constraints and limitations of the system are discussed more generally.

5.1.1 Strengths

The system successfully identifies vehicle RPM when idling. It detects both intermittent and constant misfires and has shown promising results at detecting inconsistent compression between cylinders.

The system was found to be invariant to microphone type and position. Both Android and iOS smartphones were used. The vehicle was recorded indoors, outdoors, and inside the vehicle and faults were found to be identified accurately. In many cases, recording the engine from inside the vehicle was found to be more effective as it serves as a lowpass filter to remove high-frequency engine noise.

The system also performs well across a range of four cylinder vehicles, with both petrol or diesel variants. Low compression and misfires produce similar spectral effects across all four cylinder vehicles. It was proven from testing vehicles with organic engine faults that the same spectral features are present as that of vehicles tested with induced faults.

5.1.2 Limitations

The system is sensitive to noise generated by loud sources, in particular wind noise from recording outdoors. Unfortunately, smartphone microphones are susceptible to sounds generated by wind noise. Wind creates high magnitude noise across the entire frequency spectrum that prevents engine sounds from being identified in the signal. As a result of the added noise, misfire false positives were detected.

There were also dataset limitations with the investigation. While several vehicles were tested with the misfire detection algorithm, performing compression tests was more difficult. Many of the vehicles that were made available for the duration of the project were diesel engines that required a set of compression testing tools that could not be obtained for the duration of the project. This resulted in only a single vehicle being tested with induced low compression and with baseline compression figures. Vehicle recordings with organically occurring low compression could also not be tested because of a lack of access to vehicles with compression issues.

Using a pressure gauge to obtain baseline compression readings also had limitations when interpreting the relative compression test algorithm results. The pressure tester only displays the maximum compression achieved for the compression test rather than an instantaneous compression within the cylinder.

5.2 Challenges

The greatest challenge of the project was its cross-disciplinary nature. It required knowledge in computer science, audio engineering and mechanical engineering to understand how engines work, how the data is collected and how to extract information from the data. Extensive background research was performed, particularly on the theory of combustion engines, to have sufficient understanding of the topic to analyse the engine audio.

The fields of DSP and MIR fields are both broad and mature domains. It was challenging to identify the techniques that would be best suited to engine fault detection. I found that by testing an ensemble of the most common techniques on the data worked, I formed an understanding of what techniques worked best to solve the problem.

While it was found that, for the most part constraining the project to smartphone audio was an advantage, it was not without its challenges. Smartphone manufacturers vary the microphones used in their handsets widely. This results in significant differences in quality and frequency response. There is limited literature testing the frequency response of iOS devices [21], but no information regarding Android device microphones could be found. Furthermore, fragmentation on the Android platform results in several different audio recording applications being used. Each of these recording applications have different settings regarding the recording quality, and many were found to default to a highly compressed recording format.

The data collection process also had a number of challenges. While some faults, such as inducing misfires, were relatively easy to collect data for, inducing low cylinder compression in a measurable way required some creativity. The difficulty regarding inducing low compression was that the tester o-ring created a seal even when only screwed in a few turns

into the spark plug hole. The final technique used was to firmly hold the compression tester in the spark plug hole without screwing it in.

Obtaining organic test data was also a challenge. It relied on friends and family encountering issues with their vehicles, recording the fault and sending it for analysis. While several recordings of misfires were received, there were very few recordings of engine cranking because of the more complex nature of the test.

5.3 Future Work

Whilst a system for detecting engine faults was successfully developed, there is further system evaluation that could not be completed. Additional enhancements, features, and developments can be added based on the work completed in this project.

Improvements could be made to the RPM detection algorithm such that, in the absence of the fundamental engine frequency, as seen in some tests, the fundamental frequency could be inferred by identifying the harmonic frequencies present. This technique has been found to be similar to the way that humans perceive sound [23].

No reference event could be obtained during this project to identify in what cylinder the faults are occurring. Future work should continue to investigate potential time-frequency features that could be used as reference events to identify the cylinder in which the fault is present.

While both constant and intermittent misfires were successfully detected in this project, the number of misfires and the cylinders in which they occurred was not. While two-cylinder misfires with even offsets could easily be identified, odd offset cylinders misfiring produces very similar inharmonic components to that of a single-cylinder misfire, making them harder to identify. Further analysis will be required to investigate if other identifying features can be found.

This project successfully detected misfires and low engine compression faults in engines. However, these faults can be caused by failures in many different engine components. For example, misfires could be as a result of bad spark plugs, ignition wiring, coil packs, mass airflow sensor, vacuum leaks and more [7]. Some of the causes were partially investigated and identified in audio during this project. For example, low compression due to a cylinder head leak can be identified by a distinctive sound of air escaping from the region. For many of these faults, there are distinctive characteristics in the engine operation that causes the fault. For example, if an intermittent misfire is caused by the mass airflow sensor, it may also be unable to idle smoothly [24]. Future work may aim to expand on the idea by either finding the source of the faults or by narrowing the possible causes.

To form a more complete evaluation of the relative compression detection algorithm, compression tests and induced low compression will be recorded for several other petrol and diesel vehicles. Rather than using a pressure gauge to measure the maximum compression achieved in each cylinder, a relative compression tester should be used to identify instantaneous starter motor load [12] to provide a more accurate relative compression baseline for evaluating the tests.

The system was designed with mobile device compatibility. A mobile application could record the audio, analyse it and provide the user with results as faults are found. Applying the work as a mobile application would resolve many of the audio recording inconsistencies experienced, particularly on the Android platform. The application could capture the raw microphone audio rather than having to process compressed audio recorded by a third party mobile application. Furthermore, a mobile application implementation could be distributed to mechanics who work on faulty vehicles every day. They could record unrecognised faults, which could be used for further evaluation and to help improve the system. The application could also detect sub-optimal recording conditions such as loud background or wind noise and inform the user when recording the audio.

6 Conclusion

This dissertation aimed to investigate information obtained regarding the operation of an internal combustion engine from smartphone audio.

It was found that experienced mechanics frequently identify engine faults by ear, showing promising evidence that the information could be present in audio recorded of the vehicle. The investigation focused on detecting engine misfires and low relative compression.

Spectral analysis of the audio accurately identified the engine RPM, and constant and intermittent misfires from recordings of an internal combustion engine idling.

Significant inconsistency in relative compression was successfully identified by statistically comparing cylinder compression events from recordings of vehicles turning over. While it was found that low relative compression was identified, the cylinders in which the low compression occurred could not be accurately distinguished. Further evaluation is required to investigate more accurate statistical approaches to identify low relative compression. More accurate control testing methods should also be used to obtain more granular data for evaluation.

Android and iOS devices both yielded successful results when used to identify engine faults. It was also found that misfire detection results were invariant to recording location. High accuracy was achieved recording from both inside and outside the vehicle. However, it was found that mobile device microphones are susceptible to wind noise, causing occlusion of spectral information used to identify faults. Further work could aim to investigate methods of mitigating the effects of wind noise in obtaining engine recordings.

The results show promise that future work to add additional features and improvements is possible. Furthermore, a mobile application could be created to implement the work completed in the project practically.

Bibliography

[1] Central Statistics Office. Transport omnibus, 2019. ISSN 2009-5643.

[2] Marco Grossi. A sensor-centric survey on the development of smartphone measurement and sensing systems. *Measurement*, 135:572–592, 2019. ISSN 0263-2241. doi: https://doi.org/10.1016/j.measurement.2018.12.014.

[3] Road Safety Authority. Ncts make & model data (csv) 2019. `https://www.rsa.ie/Documents/NCT/2019%20Pass%20Fail%20rates/Pass%20Fail%20rates%20by%20centre%202019.csv`. retrieved April 2021.

[4] Publications Office of the European Parliament. Directive 98/69/ec of the european parliament, 1998. `https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31998L0069:EN:HTML`.

[5] Jerzy Merkiszt, Piotr Bogusz, and Rafał Grzeszczyk. Overview of engine misfire detection methods used in on board diagnostics. *Combustion Engines*, 8:1–2, 01 2001.

[6] J. Heywood. *Internal Combustion Engine Fundamentals 2E*. McGraw-Hill Education, 2018. ISBN 9781260116113.

[7] Mobil. Simple steps for engine misfire diagnosis and repair. `https://www.mobil.com/en/lubricants/for-personal-vehicles/auto-care/vehicle-maintenance/diy-engine-misfire-diagnosis-and-repair`. retrieved April 2021.

[8] agradetools.com. Old school mechanic trick # 5: Finding & identifying misfires quickly w/o special tools. `https://agradetools.com/old-school-mechanic-trick-5-finding-identifying-misfires-quickly-w-o-special-too` retrieved April 2021.

[9] S.N. Dandarea and S.V. Dudulb. Multiple fault detection in typical automobile engines: A soft computing approach. *WSEAS Transactions on Signal Processing*, 10:254–262, 01 2014.

[10] Josh Siegel, Sumeet Kumar, Isaac Ehrenberg, and Sanjay Sarma. Engine misfire detection with pervasive mobile audio. pages 226–241, 01 2016.

[11] Popular Mechanics. Cars 101: How to do a compression test. https://www.popularmechanics.com/cars/how-to/a8520/cars-101-how-to-do-a-compression-test-14912158/. retrieved April 2021.

[12] Ditex. Understanding the relative compression analysis. https://autoditex.com/page/relative-compression-test-67-1.html. retrieved April 2021.

[13] J. Cooley, P. Lewis, and P. Welch. The finite fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 17(2):77–85, 1969. doi: 10.1109/TAU.1969.1162036.

[14] J. Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3): 235–238, 1977. doi: 10.1109/TASSP.1977.1162950.

[15] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978. doi: 10.1109/PROC.1978.10837.

[16] Meinard Müller. *Fundamentals of Music Processing*. 01 2015. ISBN 978-3-319-21944-8. doi: 10.1007/978-3-319-21945-5.

[17] S. S. Stevens and J. Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940. ISSN 00029556. URL http://www.jstor.org/stable/1417526.

[18] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. A tutorial on deep learning for music information retrieval. *CoRR*, abs/1709.04396, 2017. URL http://arxiv.org/abs/1709.04396.

[19] Jonathan Driedger, Meinard Müller, and Sascha Disch. Extending harmonic-percussive separation of audio signals. 01 2014.

[20] Joshua Fineberg. Guide to the basic concepts and techniques of spectral music. *Contemporary Music Review*, 19(2):81–113, 2000. doi: 10.1080/07494460000640271. URL https://doi.org/10.1080/07494460000640271.

[21] Andrew Smith. Using ios devices for noise and vibration measurements. *Sound & Vibration*, 2017, 03 2017. doi: 10.4271/2015-01-2283. URL http://www.sandv.com/downloads/1703smit.pdf.

[22] Aditya Jain, Ramta Bansal, Avnish Kumar, and K. D. Singh. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International journal of applied & basic medical research*, 5 (2):124–127, 2015. ISSN 2229-516X. doi: 10.4103/2229-516X.157168. URL https://doi.org/10.4103/2229-516X.157168.

[23] P. A. Cariani and B. Delgutte. Neural correlates of the pitch of complex tones. I. Pitch and pitch salience. *J Neurophysiol*, 76(3):1698–1716, Sep 1996.

[24] Mechanic Base. 8 symptoms of a bad maf sensor, location & replacement cost. https://mechanicbase.com/engine/mass-air-flow-sensor-symptoms-maf/. retrieved April 2021.

A Emulating Spectral Misfire Features

Section 3.2.2 identifies inharmonic overtones that occur only when an engine is misfiring. This section attempts to emulate the phenomenon with sine waves.

To attempt to prove the hypothesis of why inharmonic overtones are visible when misfires are occurring, the engine firing signals were emulated as sine waves. While in reality, an engine's firing signal is more complex than that of a sine wave of a fixed frequency, it should suffice to test the hypothesis.

When a cylinder misfire occurs, the discontinuity generated is reflected in the signal as a magnitude of zero for that region. A sine wave with this property can be created to emulate this event. The spectrogram representations seen in *Section 3.2.2* show that there are significant 2^{nd} and 3^{rd} harmonics also present in the signal. These can also be emulated by sine waves.

Furthermore, another spectral event occurs when a cylinder misfires. The engine is thrown out of balance, causing it to swing on its engine mounts. This event generates a sound and occurs periodically at the frequency of the misfire. Another sine wave can be added to emulate this event.

An FFT can then be performed on the generated sine wave with discontinuities and the results observed. *Figure A.1* shows the simplified sine wave components used to emulate an engine firing and an FFT of the emulated signal.

Emulations are visualised first for an engine operating normally.

A single cylinder misfire is emulated by flattening a region of the signal that is one period of the fundamental frequency in length every four periods and by introducing a signal that represents the engine shaking (as seen by the pink signal in *Figure A.1*). To most closely emulate a frame of the STFT shown in *Figure 3.6*, the signal is the same as the STFT frame length of 1 second.

The same is repeated for single-cylinder, two-even and two-odd misfires with the misfire

regions being flattened and an engine shaking signal being introduced. Notice that in the case of the two-odd offset misfires, the engine shaking signal only occurs for the period of the first misfire. This is because there is not sufficient momentum for the sound to be sustained into the second misfire as it occurs directly after the first cylinder misfire.

The resulting FFT of the complex signal matches that of the inharmonic overtones seen in *Section 3.2.2*. The degree to which each sine wave contributes to the FFT can also be seen by the colour of the plotted signal.

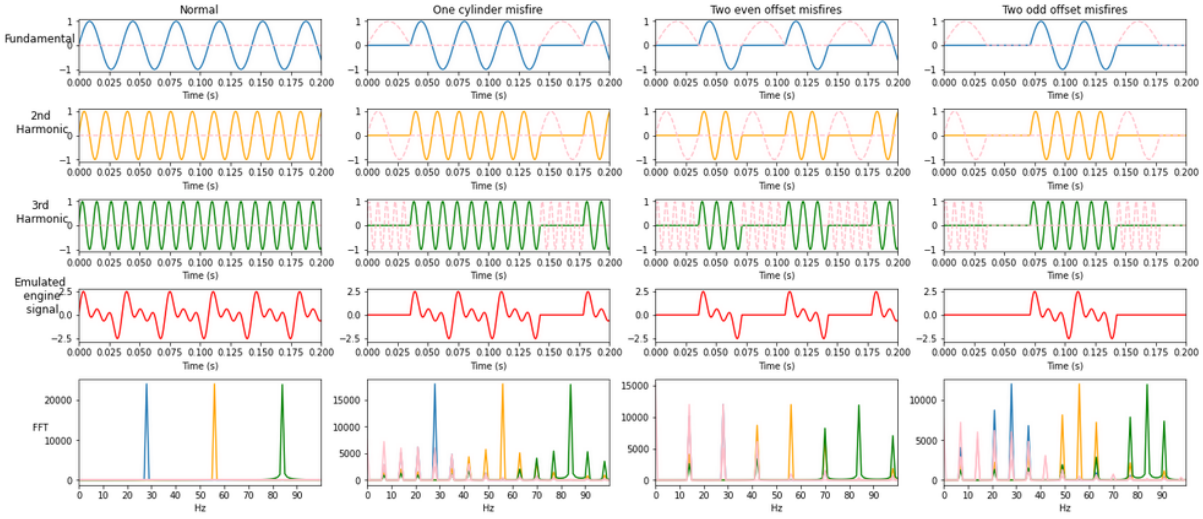


Figure A.1: Using sine waves at 28Hz, 56Hz and 84Hz to emulate the fundamental frequency, 2nd harmonic and 3rd harmonic respectively of an engine firing at 840RPM. Misfires are emulated as regions with no magnitude. Examples are shown for cylinders evenly spaced and oddly spaced in the firing order respectively. FFTs of the signals are generated, showing the frequency components found to be present in each signal and how much each harmonic component contributes to each frequency component found.

B Identifying the Starter Motor Sound

Figure 3.9 shows the recording of the compression test performed on cylinder 1. The blue lines follow a harmonic sound that drops in frequency when the cylinder comes under compression. It is clear that the sound is related to the starter motor in some way. However, it may be useful to intuitively understand what engine event it represents. This section discusses the potential source of this sound.

The hypothesis is that the sound was either caused by the starter motor itself or by the flywheel which connects the starter motor to the crankshaft. The frequency of the noise caused by both was calculated to see if they could potentially create a signal at this same frequency.

B.1 RPM

To begin, the RPM of the engine when the starter motor is turning over was approximately calculated. While the RPM of the vehicle powered by the starter motor is much too slow to generate a signal in the 200-400Hz range, it may prove to be a useful feature to help identify the source of the sound.

Roughly speaking, from *Figure 3.9*, the time taken for four engine strokes is $680ms$ or for a single stroke $170ms$. The engine cranking frequency can be calculated with this information by:

$$f_{\text{crank}} = \frac{1000ms}{170ms} = 5.88Hz$$

Recall from *Eq 3.1* that:

$$RPM = \frac{5.88Hz}{2} \cdot 60 = 176.4RPM$$

B.2 Starter Motor Frequency

The starter motor connects to the flywheel by the use of a gear. If the number of teeth on both the flywheel and the starter motor are known, the starter motor frequency can be calculated.

On the Audi TT vehicle tested in this example, there are $C_{\text{flywheel}} = 132$ flywheel teeth and $C_{\text{starter}} = 10$ starter motor teeth.

The ratio of flywheel turns to starter motor turns can then be calculated by:

$$\frac{C_{flywheel}}{C_{starter}} = \frac{132}{10} = \frac{13.2}{1}$$

$f_{starter}$ can then be calculated by:

$$f_{\text{starter}} = \frac{C_{flywheel}}{C_{starter}} \cdot \frac{f_{\text{crank}}}{2} = 13.2 \cdot \frac{5.88}{2} = 38.81Hz$$

B.3 Flywheel Tooth Frequency

This is the frequency at which a tooth turns on the flywheel. It can be calculated by:

$$f_{\text{flywheel tooth}} = C_{flywheel} \cdot \frac{f_{\text{crank}}}{2} = 132 \cdot \frac{5.88}{2} = 388.08Hz$$

While $f_{starter}$ is too low to be the cause of the sound directly, $f_{\text{flywheel tooth}}$ seems to be a very likely candidate as the source of the sound. It has a frequency very similar to that of the fundamental of the signal seen in *Figure 3.9*. Furthermore, when the starter motor comes under more load, it requires more force to turn the flywheel and as a result generates a sound of a higher magnitude.

C Code Listings

This appendix contains Python implementations of algorithms discussed in the dissertation.

C.1 RPM Detection

```
import librosa
import numpy as np
from scipy.signal import find_peaks

def get_instantaneous_rpm(H, sample_rate):
    freqs = librosa.fft_frequencies(sr=sample_rate,
                                    n_fft=sample_rate)

    max_freq = np.argmax(freqs >= 100)

    H, _ = librosa.magphase(H)

    rpms = []
    max_frame_bins = []
    for f in H.T:
        frame = rectify(f[0:max_freq])
        max_frame_bin = np.argmax(frame)

        peaks = find_peaks(frame, height=np.mean(frame))[0]

        min_bin = np.min(peaks)
        min_freq = freqs[min_bin]
        rpms.append((min_freq / 2) * 60)

    return rpms
```

C.2 Misfire Detection

```
from earl.helpers import rectify
import numpy as np
from scipy.signal import find_peaks
import librosa
from collections import Counter
from enum import Enum


def detect_misfires(sr, spec):
    class MisfireStates(Enum):
        NO_MISFIRE = 0
        MISFIRE = 1
        INTERMITTENT = 2

    INTERMITTENT_THRESH = 0.8
    MISFIRE_THRESH = 0.05

    NO_MISFIRE_PEAK_NUM = 5
    MISFIRE_CAP = 14


    H, P = librosa.decompose.hpss(
        spec,
        kernel_size=(100, 10),
        margin=(10, 50),
    )

    # Get limit of frequencies to consider.
    freqs = librosa.fft_frequencies(sr=sr,
                                    n_fft=sr)
    max_freq = np.argmax(freqs >= 100)

    H, _ = librosa.magphase(H)

    # Label each frame with a misfire state.
    frame_labels = []
    for f in H.T:
```

```
        frame = rectify(f[0:max_freq])
        peaks = find_peaks(frame,
                           height=np.mean(frame)/3)[0]
        n_peaks = len(peaks)

        if (n_peaks <= NO_MISFIRE_PEAK_NUM or
            n_peaks >= MISFIRE_CAP):
            frame_labels.append(MisfireStates.NO_MISFIRE)
        else:
            frame_labels.append(MisfireStates.MISFIRE)

    # Calculate the probability of misfire.
    misfire_frame_labels = [p for p in frame_labels
                            if p is not MisfireStates.NO_MISFIRE]
    prob_misfire = (len(misfire_frame_labels) / len(H.T))

    counts = Counter(misfire_frame_labels)

    # Make decision on classification.
    if prob_misfire <= MISFIRE_THRESH:
        return (MisfireStates.NO_MISFIRE, frame_labels)
    elif prob_misfire <= INTERMITTENT_THRESH:
        return (MisfireStates.INTERMITTENT, frame_labels)
    else:
        return (counts.most_common(1)[0][0], frame_labels)
```

C.3 Relative Compression Detection

C.3.1 Crank Segmentation

```
from earl.helpers import energy_onset_detect
import librosa
from scipy.signal import find_peaks
import numpy as np

def segment_cranks(y, sr, mel_spec, mode='time'):
    HOP_LEN = 256
    N_MELS = 256
```

```
HPSS_KERNEL_SIZE = 100
HPSS_MARGIN_SIZE = 1
PERCUSSIVE_RMS_FRAME_LEN = 510

CRANK_CENTRES_TIME_THRESH = 0.1

_, P = librosa.decompose.hpss(
    mel_spec,
    kernel_size=HPSS_KERNEL_SIZE,
    margin=HPSS_MARGIN_SIZE,
)

p = librosa.feature.rms(
    S=P,
    frame_length=PERCUSSIVE_RMS_FRAME_LEN,
)[0]

# Crank centres
t = np.linspace(0, len(y) / sr, p.shape[0])

crank_centre_points = find_peaks(
    p,
    distance=np.argmax(t >= CRANK_CENTRES_TIME_THRESH),
    height=np.mean(p),
)[0]

# Crank start and end points
p_peak_thresh = np.mean(p) / 3

p_peak_points = [i for i in range(1, len(p) - 1)
                 if (p[i-1] > p_peak_thresh and
                     p[i] <= p_peak_thresh)
                     or (p[i+1] >= p_peak_thresh and
                         p[i] < p_peak_thresh)]

crank_start_points = [p_peak_points[
                      np.argmax(p_peak_points > centre) - 1
                    ] for centre in crank_centre_points]
crank_end_points = [p_peak_points[
```

```
                        np.argmax(p_peak_points > centre)
                    ] for centre in crank_centre_points]

    if mode == 'time':
        crank_start_t = [t[i] for i in crank_start_points]
        crank_centre_t = [t[i] for i in crank_centre_points]
        crank_end_t = [t[i] for i in crank_end_points]
        return crank_start_t, crank_centre_t, crank_end_t
    elif mode == 'samples':
        s = [r * (PistonCrank.N_MELS -
                  (PistonCrank.N_MELS - PistonCrank.HOP_LEN))
             for r in range(mel_spec.shape[1])]
        crank_start_s = [s[p] for p in crank_start_points]
        crank_centre_s = [s[p] for p in crank_centre_points]
        crank_end_s = [s[p] for p in crank_end_points]

        return crank_start_s, crank_centre_s, crank_end_s
```

C.4 Statistical Similarity

```
import numpy as np
from earl.xcorr import correlate_template

def get_crank_correlations(samples, sample_rate,
                           crank_starts, crank_ends):
    crank_correlations = []
    crank_correlation_peaks = []

    t = np.linspace(0, len(samples)/sample_rate, len(samples))

    for crank in zip(crank_starts, crank_ends):
        cross_corr = correlate_template(
            samples,
            samples[crank[0]:crank[1]],
            mode='same',
        )

        crank_correlations.append(cross_corr)
```

```
        # Peak finding
        # Finds the max correlation value within each crank segment.
        crank_correlation_peaks.append(
            [c[0] + np.argmax(cross_corr[c[0]:c[1]])
             for c in zip(crank_starts, crank_ends)])

    return crank_correlations, crank_correlation_peaks
```

C.5 Self-Similarity

```
import numpy as np

def cylinder_self_correlations(crank_correlations,
                               crank_correlation_peaks):
    FIRING_ORDER = [1, 3, 4, 2]

    max_corrs = [None, None, None, None]
    max_corrs_std = [None, None, None, None]
    max_corr_idxs = [None, None, None, None]

    for i in range(len(crank_correlation_peaks)):
        cyl_order = i % 4
        cyl = FIRING_ORDER[cyl_order] - 1

        ps = [crank_correlations[i][crank_correlation_peaks[i][p]]
                for p in range(len(crank_correlation_peaks[i]))
                    if p % 4 == cyl_order and p != i]

        correlation = np.mean(ps)

        if (max_corr_idxs[cyl] is None or
            max_corrs[cyl] < correlation):
            max_corrs[cyl] = correlation
            max_corrs_std[cyl] = np.std(ps)
            max_corr_idxs[cyl] = i

    return max_corrs, max_corr_idxs
```

C.6 Pair-wise Similarity

```
import numpy as np

def cylinder_pair_correlations(crank_correlations,
                               crank_correlation_peaks):
    FIRING_ORDER = [1, 3, 4, 2]

    cylinder_peaks = {(1, 4): [], (1, 3): [],
                      (2, 3): [], (2, 4): []}

    for pair in cylinder_peaks:
        s_i = pair[0] - 1
        t_i = pair[1] - 1

        cylinder_correlation = crank_correlations[s_i]
        cylinder_correlation_peaks =
            crank_correlation_peaks[s_i]

        cylinder_peaks[pair] = [
            cylinder_correlation[
                cylinder_correlation_peaks[p_i]
            ] for p_i in range(len(cylinder_correlation_peaks))
                if FIRING_ORDER[p_i % 4] == pair[1]]

    return [[key, np.max(val)]
            for key, val in cylinder_peaks.items()]
```

C.7 No Cylinder Compression

```
import numpy as np

def no_compression_cylinder(crank_starts_t, crank_ends_t):
    CRANK_DIST_THRESH = 0.3
    CRANK_LEN_THRESH = 0.4

    # Start crank distances
    crank_dists = [crank_starts_t[i+1] - crank_starts_t[i]
```

```
                for i in range(len(crank_starts_t) - 1)]
    avg_crank_dist = np.mean(crank_dists)

    for dist in crank_dists:
        if (dist > (avg_crank_dist +
            (avg_crank_dist * CRANK_DIST_THRESH)) or
            dist < (avg_crank_dist -
            (avg_crank_dist * CRANK_DIST_THRESH))):
            return True

    # Crank event lengths
    crank_lens = [c[1] - c[0]
                  for c in zip(crank_starts_t, crank_ends_t)]
    avg_crank_len = np.mean(crank_lens)

    for crank_len in crank_lens:
        if (crank_len > (avg_crank_len +
            (avg_crank_len * CRANK_LEN_THRESH)) or
            crank_len < (avg_crank_len -
            (avg_crank_len * CRANK_LEN_THRESH))):
            return True

    return False
```

C.8 Relative Compression Error

```
from earl.helpers import band_pass_filter

def relative_compression_error(y, sr):
    LOWCUT = 200
    HIGHCUT = 400

    # Get the cranks
    crank_start, _, crank_end = segment_cranks(
        y=y, sr=sr, mode='samples')
    crank_start_t, _, crank_end_t = PistonCrank(
        y=y, sr=sr, mode='time')
```

```
# Check for missing cranks because of no compression.
if no_compression_cylinder(crank_start_t, crank_end_t):
    return -1

# Bandpass filtering.
y_filt = band_pass_filter(y, sr, LOWCUT, HIGHCUT)

# Find all correlations
crank_correlations, crank_correlation_peaks =
    get_crank_correlations(y_filt, sr,
                           crank_start, crank_end)

# Find each cylinders best self-correlation
max_self_corrs, max_self_corr_idxs =
    cylinder_self_correlations(crank_correlations,
                               crank_correlation_peaks)

# Choose the best self correlation crank for each cylinder.
best_correlations = [crank_correlations[c]
                     for c in max_self_corr_idxs]
best_correlation_peaks = [crank_correlation_peaks[c]
                          for c in max_self_corr_idxs]

# Compare cylinder pairs
pair_correlations =
    cylinder_pair_correlations(best_correlations,
                               best_correlation_peaks)

return [(p[0], max_self_corrs[p[0][0] - 1] - p[1])
        for p in pair_correlations]
```