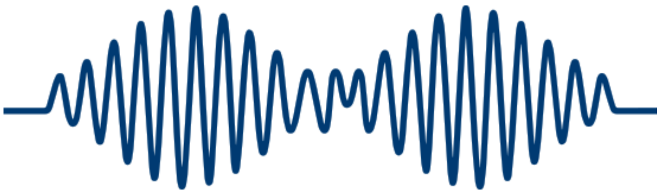
EE-232: Signals and Systems

Project Report

Audio Splicing Detection and Localization in MATLAB®



Group 6 – Members

Name	CMS ID
Danial Ahmad	331388
Muhammad Umer	345834
Syeda Fatima Zahra	334379

1 Table of Contents

2 Table of Figures

3 Introduction

Audio splicing is one of the most common manipulation techniques in the area of audio forensics. Especially after countless breakthroughs in machine learning and deep neural networks, forgery and manipulation in speeches of political figures is highly prevalent and ultimately, detrimental to a productive and functioning society. Detection of splicing in audio signals, therefore, is thus of critical importance. Although the proposed architecture in this report is not highly accurate, it requires very little computation power to localize splicing in audio.

4 Algorithm Flowchart

Following figure summarizes the proposed approach in a simplified flowchart:

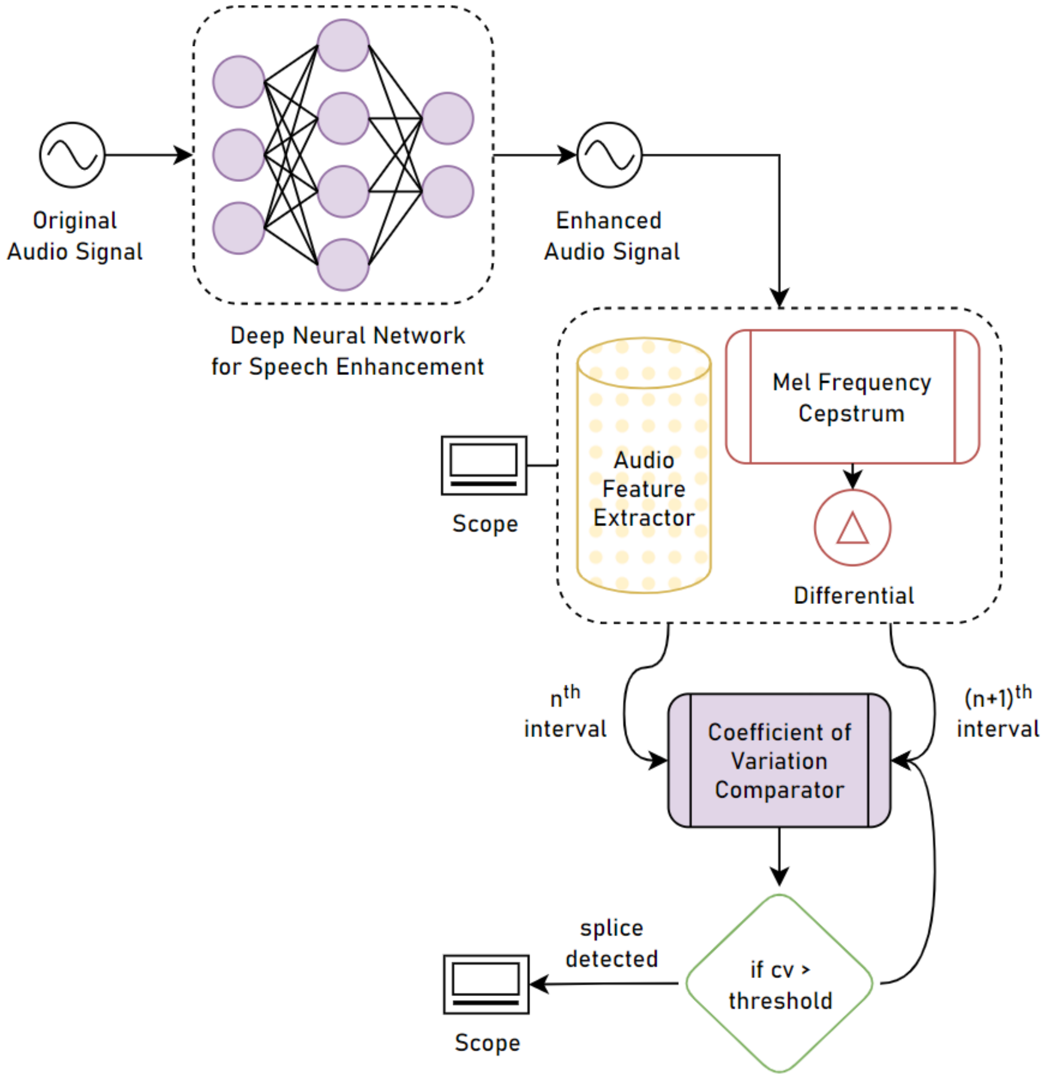


Figure 1 Algorithm Flowchart

5 Methodology

This section briefs the reader on the methodology and analysis techniques used to detect forgery in the sampled audio.

5.1 Audio Preprocessing

Audio preprocessing involves splitting and separating the background song from the speech of the speaker. A common way to accomplish this is to plot a Fourier coefficient of the audio signal and find the contributing frequencies of the speech and the audio before applying a thresholding algorithm to split the audio signals. Following code can be used to split the background and speech from a given audio signal:

```
[audioIn, Fs] = audioread('audio.mp3');
T = 1/Fs;   % Sampling period
L = length(audioIn);  % Length of signal
t = 0:(L-1)*T;  % Time vector
df = Fs/L;
audioFreq = -Fs/2:df:Fs/2-df;

D = fftshift(fft(audioIn)/length(fft(audioIn)));

% Frequency thresholds
lower_thresh = 50;
upper_thresh = 1500;

val = abs(audioFreq)<upper_thresh & abs(audioFreq)>lower_thresh;
bgFFT = D(:, 1); bgFFT(val) = 0;
speechFFT = D(:, 1); speechFFT(~val) = 0;

bgIFFT = ifftshift(bgFFT);
audioBG = ifft(bgIFFT * length(fft(audioIn)));

speechIFFT = ifftshift(speechFFT);
audioSpeech = ifft(speechIFFT * length(fft(audioIn)));

% Write separated mp3 files of background and speech
audiowrite('bg.mp3', real(audioBG), Fs);
audiowrite('enhanced_audio.mp3', real(audioSpeech), Fs);
```

However, the noise pertinent in the speech through this method is still very distorting to the splicing detection algorithm we have proposed.

As such we rely on advanced speech enhancement techniques, particularly those that make use of Deep Neural Networks. The plots of the magnitude of the original audio signal and the speech-enhanced audio signal against time are shown in Figure 2 and Figure 3.

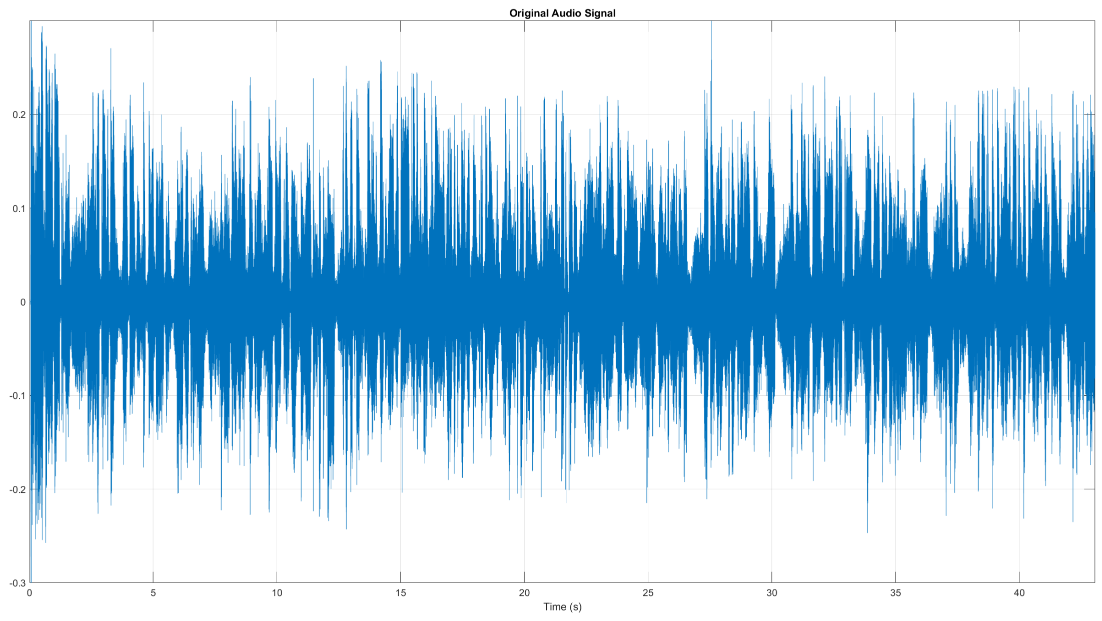


Figure 2 Original Audio Signal

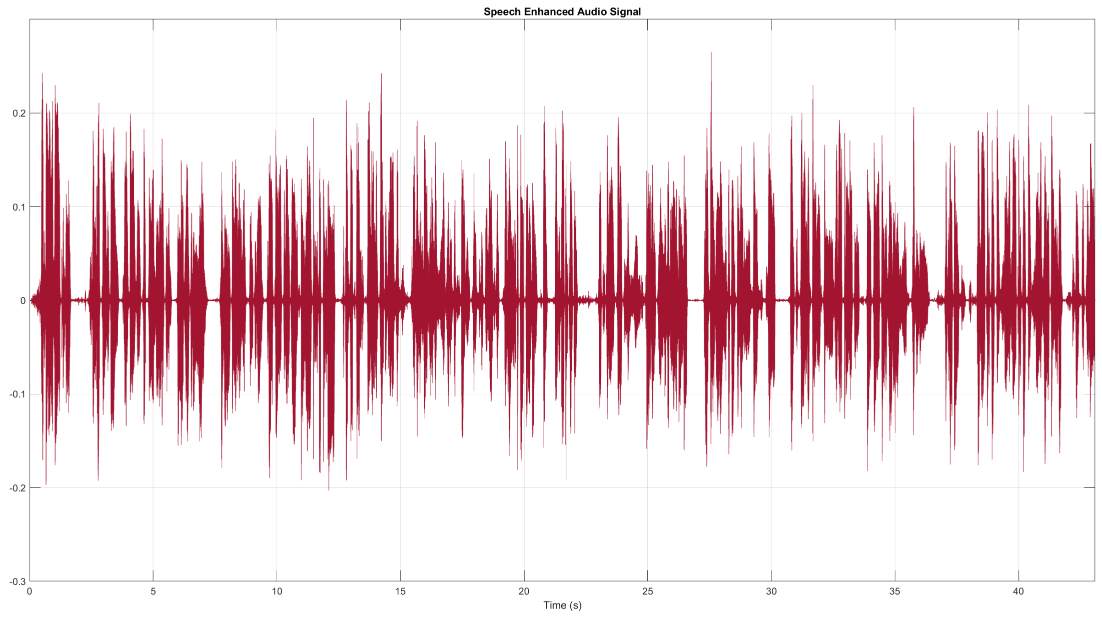


Figure 3 Speech Enhanced Audio Signal

5.2 Speech Detection

Audio Toolbox™ add-on of MATLAB® has a built-in function, detectSpeech(), to detect all bits of speech in an audio signal. Depending on the window and overlap length arguments passed into the function, the number of speech vectors detected can be increased, which is of our interest. It returns a $n \times 2$ matrix; with column one being the starting length of detected speech and column two being the ending length. To convert the lengths into time, element-wise division with the sampling frequency fs is performed. Figure 4 shows the plot of detected speech intervals.

```
idxSpeech = detectSpeech(audioIn(:, 1), fs, 'Window', win, 'OverlapLength', ...
    overlap, 'MergeDistance', mergeDist) ./ fs; % ./ fs for conversion to time
```

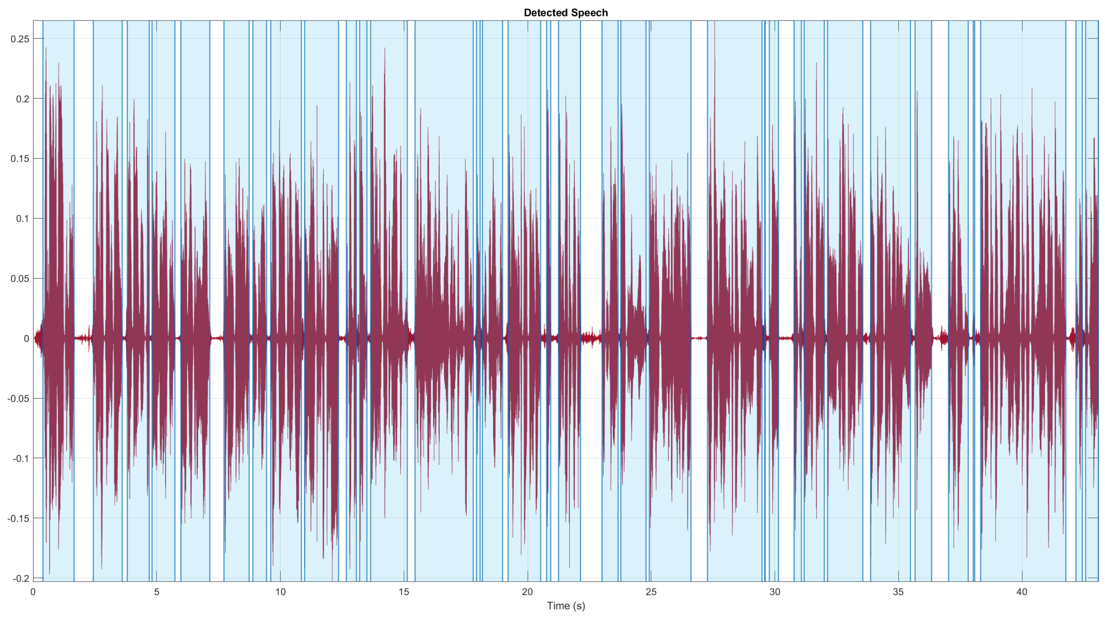


Figure 4 Detected Speech Intervals

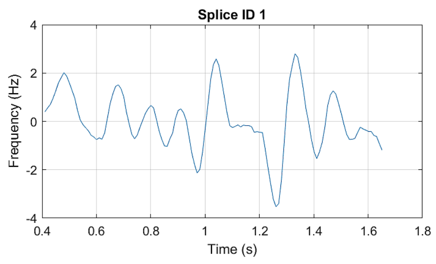
5.3 Audio Feature Extractor

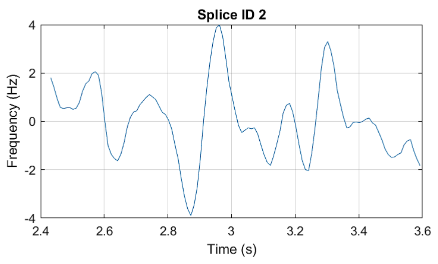
Audio Toolbox™ add-on of MATLAB® has a built-in function, audioFeatureExtracto(), for encapsulating multiple audio feature extractors into a streamlined approach. Like detectSpeech(), it too relies heavily on window and overlap lengths, and extracting as much of the features in a reasonable amount of vectors is of interest to us.

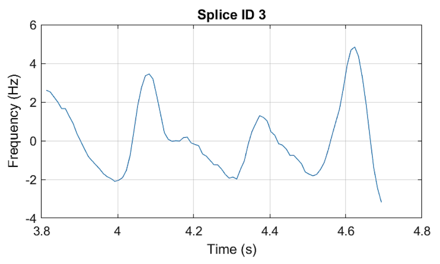
Feature extractors encapsulated include, but are not limited to, linearSpectrum, melSpectrum, barkSpectrum, mfcc, gtcc, pitch, zero-cross rate, spectralEntropy, etc. However, what really is of interest for the present case are the mel-frequency cepstral coefficients; more specifically, the differential of the mfcc, mfccDelta.

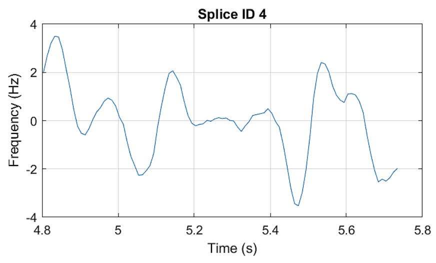
5.3.1 Mel-frequency cepstrum (mfc)

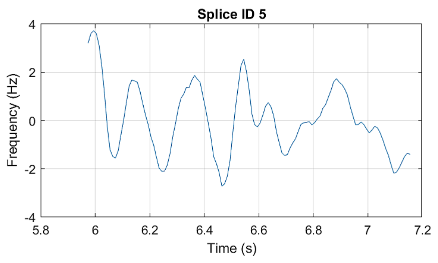
In mfc, the frequency bands are equally spaced on the mel-scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in the normal spectrum. Hence, allowing for better sound representation. As stated earlier, what we specifically target are the delta features of the mfc, mfccDelta, so as in order to understand the dynamics of power spectrum better, and directly in relation to time. The following subplots shows the plots of mfccDelta for each individual detected speech intervals:

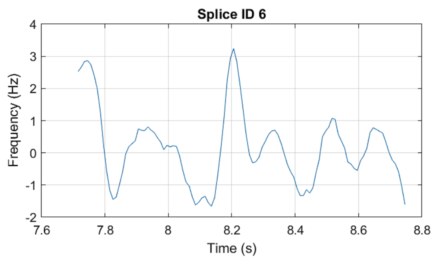

Splice ID 1
Frequency (Hz)
Time (s)
-4 -2 0 2 4
0.4 0.6 0.8 1 1.2 1.4 1.6 1.8

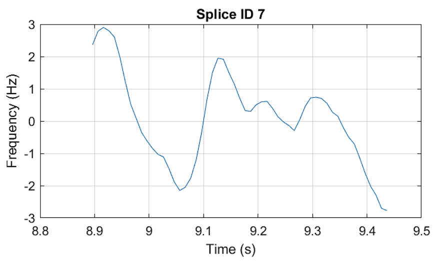

Splice ID 2
Frequency (Hz)
Time (s)
-4 -2 0 2 4
2.4 2.6 2.8 3 3.2 3.4 3.6

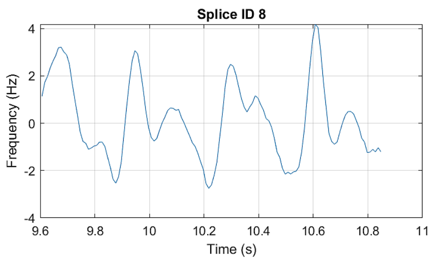

Splice ID 3
Frequency (Hz)
Time (s)
-4 -2 0 2 4 6
3.8 4 4.2 4.4 4.6 4.8

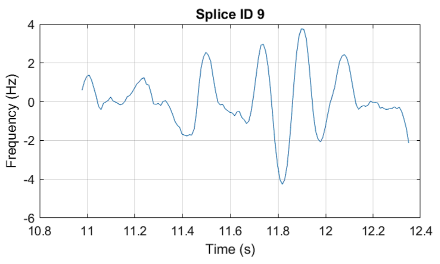

Splice ID 4
Frequency (Hz)
Time (s)
-4 -2 0 2 4
4.8 5 5.2 5.4 5.6 5.8

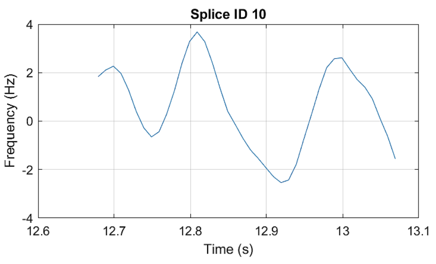

Splice ID 5
Frequency (Hz)
Time (s)
-4 -2 0 2 4
5.8 6 6.2 6.4 6.6 6.8 7 7.2

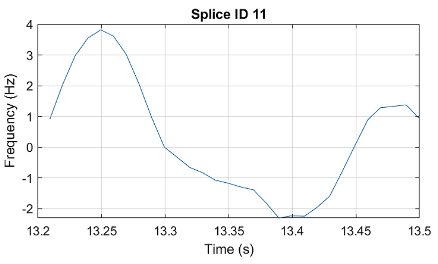

Splice ID 6
Frequency (Hz)
Time (s)
-2 -1 0 1 2 3 4
7.6 7.8 8 8.2 8.4 8.6 8.8

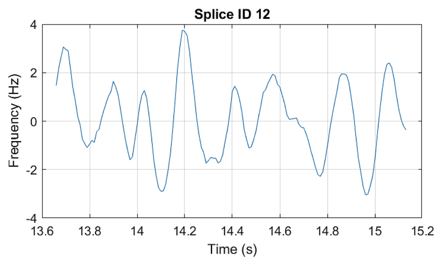

Splice ID 7
Frequency (Hz)
Time (s)
-3 -2 -1 0 1 2 3
8.8 8.9 9 9.1 9.2 9.3 9.4 9.5

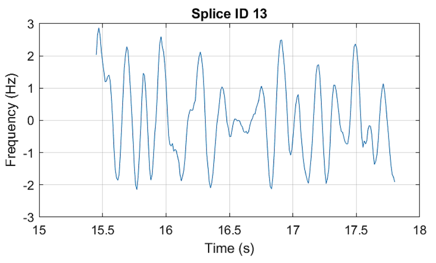

Splice ID 8
Frequency (Hz)
Time (s)
-4 -2 0 2 4
9.6 9.8 10 10.2 10.4 10.6 10.8 11

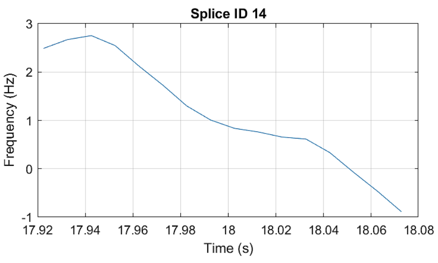

Splice ID 9
Frequency (Hz)
Time (s)
-6 -4 -2 0 2 4
10.8 11 11.2 11.4 11.6 11.8 12 12.2 12.4

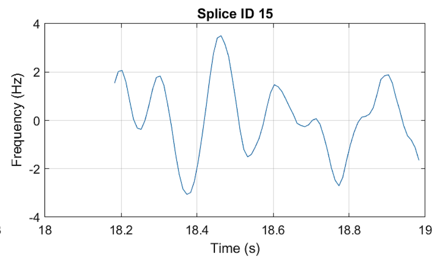

Splice ID 10
Frequency (Hz)
Time (s)
-4 -2 0 2 4
12.6 12.7 12.8 12.9 13 13.1

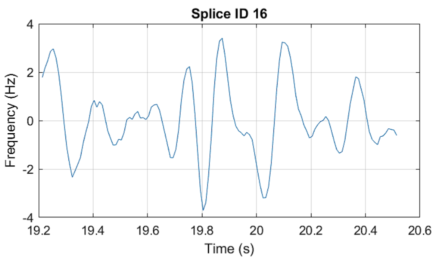

Splice ID 11
Frequency (Hz)
Time (s)
-2 -1 0 1 2 3 4
13.2 13.25 13.3 13.35 13.4 13.45 13.5

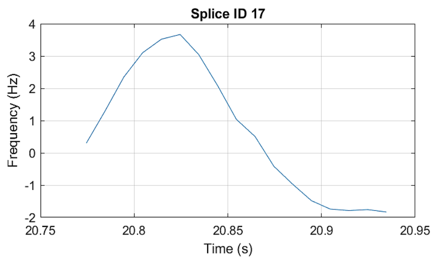

Splice ID 12
Frequency (Hz)
Time (s)
-4 -2 0 2 4
13.6 13.8 14 14.2 14.4 14.6 14.8 15 15.2

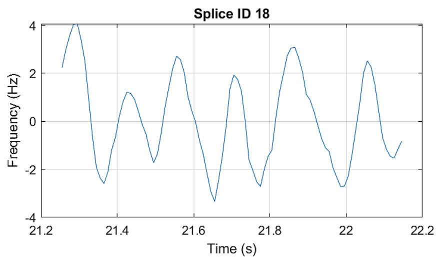

Splice ID 13
Frequency (Hz)
Time (s)
-3 -2 -1 0 1 2 3
15 15.5 16 16.5 17 17.5 18

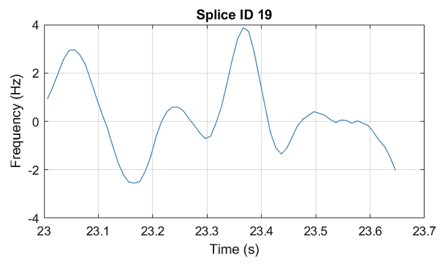

Splice ID 14
Frequency (Hz)
Time (s)
-1 0 1 2 3
17.92 17.94 17.96 17.98 18 18.02 18.04 18.06 18.08

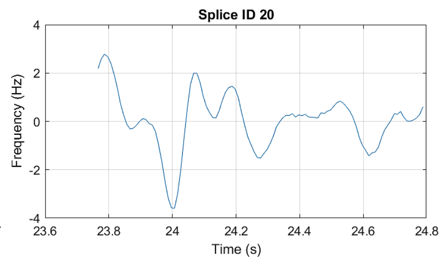

Splice ID 15
Frequency (Hz)
Time (s)
-4 -2 0 2 4
18 18.2 18.4 18.6 18.8 19

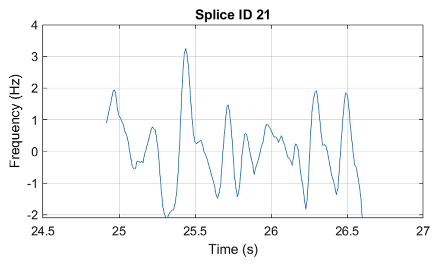

Splice ID 16
Frequency (Hz)
Time (s)
-4 -2 0 2 4
19.2 19.4 19.6 19.8 20 20.2 20.4 20.6

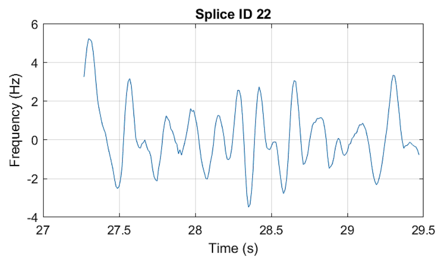

Splice ID 17
Frequency (Hz)
Time (s)
-2 -1 0 1 2 3 4
20.75 20.8 20.85 20.9 20.95

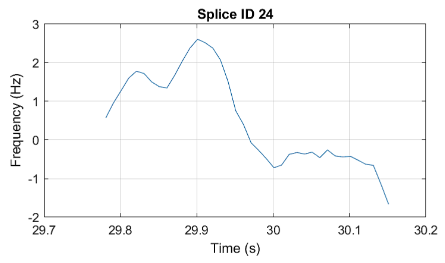

Splice ID 18
Frequency (Hz)
Time (s)
-4 -2 0 2 4
21.2 21.4 21.6 21.8 22 22.2

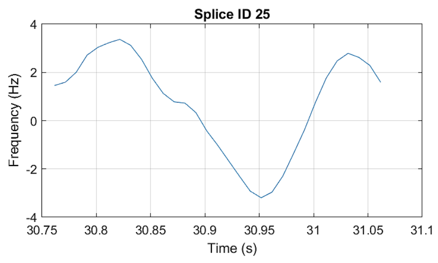


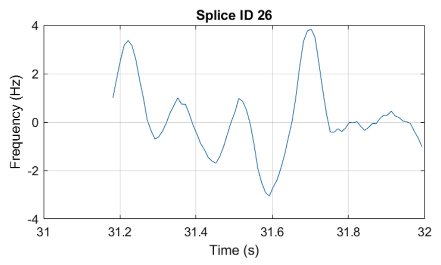


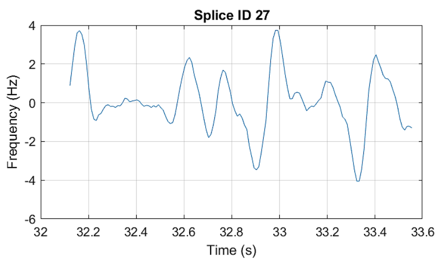


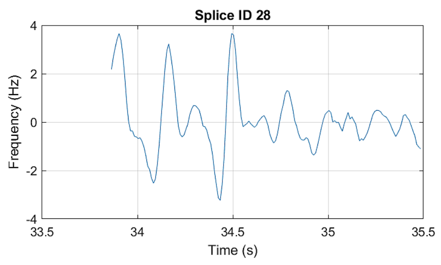


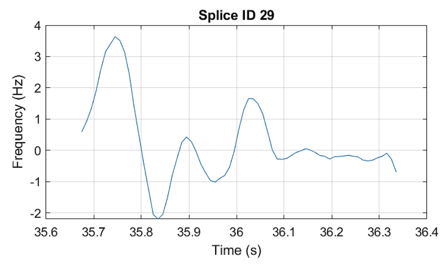


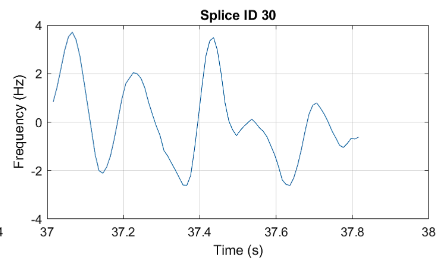


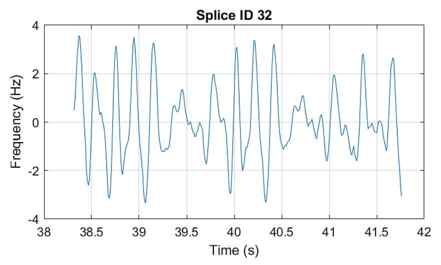


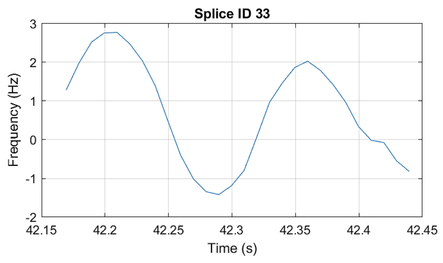


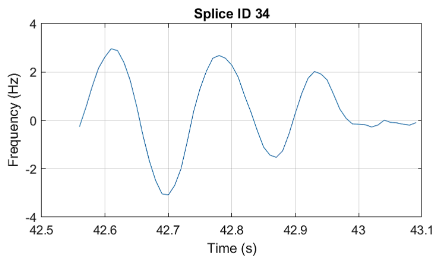












5.4 Feature Maps and Relative Statistics

After extracting the mfccDelta vectors of each individual speech interval, a viable measure of comparison should be established for the detection of splices. For that purpose, we draw the contour and surf plots of the normalized mean and standard deviations of the mfccDelta intervals. As the plots display uniformity, we thus conclude that the coefficient of variation may be a suitable metric to compare the features of each adjacent interval. The coefficient of variation is computed through the following formula:

$$\text{Coefficient of Variation } CV = \frac{\sigma}{\mu}$$

, where σ is the standard deviation, and μ is the mean of the interval.

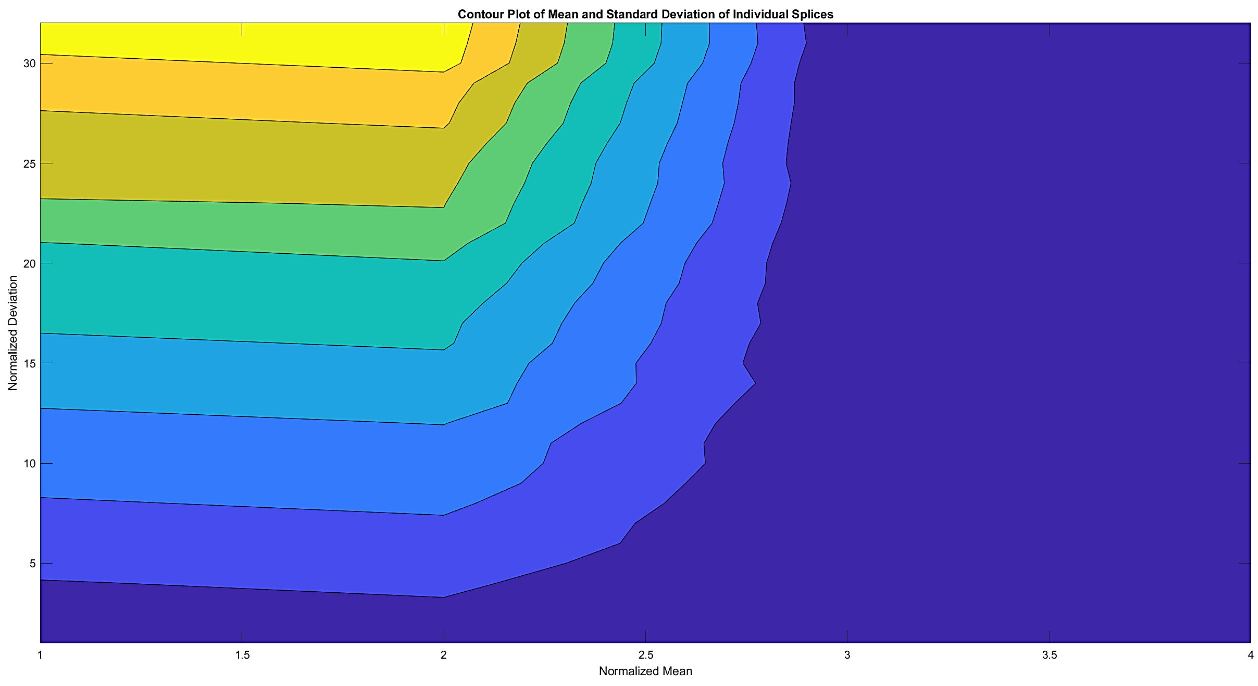


Figure 5 Contour Plot

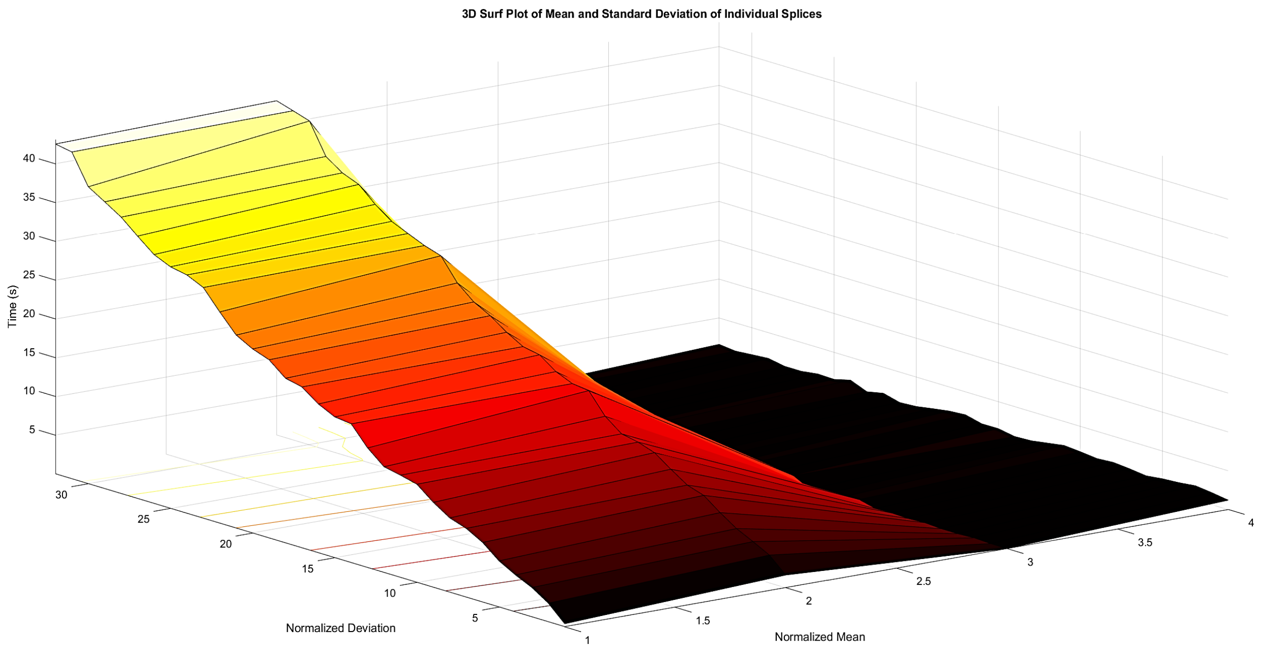
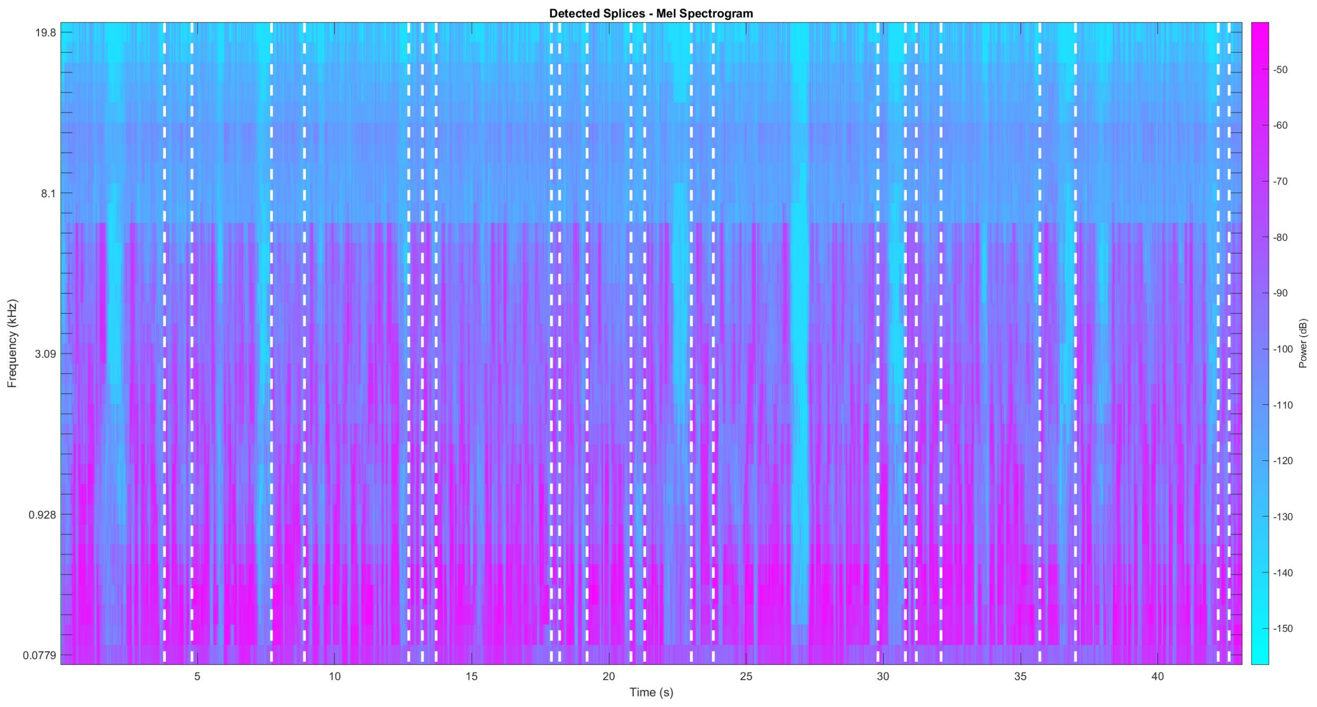
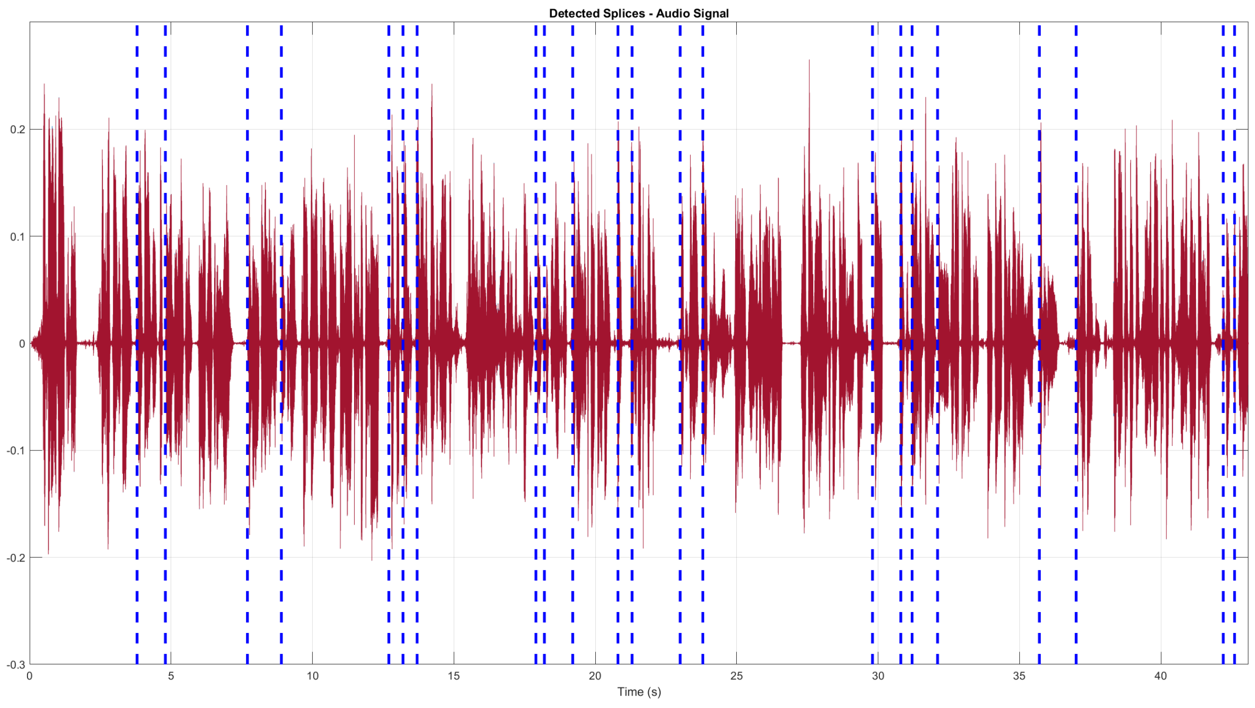


Figure 6 Surf Plot

5.5 Splice Localization

The final stage of splice detection is localization of the splice on the mel-spectrogram, as well as the audio signal. This can be achieved by marking the starting length of the latter adjacent interval while computing CV as the starting point of the splice. Following results show the located splices in the audio signal:





6 Code & Splice Localization

```
%% Read audio and find speech intervals
[audioIn, fs] = audioread('enhanced_audio.mp3');
t = linspace(0, length(audioIn) / fs, length(audioIn));

windowDuration = 0.005; % seconds

numWindowSamples = round(windowDuration * fs);
win = hann(numWindowSamples, 'periodic');

percentOverlap = 10;
overlap = round(numWindowSamples * percentOverlap / 100);

mergeDuration = 0.1;
mergeDist = round(mergeDuration * fs);

idxSpeech = detectSpeech(audioIn(:, 1), fs, 'Window', win, 'OverlapLength', ...
    overlap, 'MergeDistance', mergeDist) ./ fs; % Divide by ./ fs for conversion to time

%% Extract features
aFE = audioFeatureExtractor( ...
SampleRate = fs, ...
    Window = hamming(round(0.03 * fs), 'periodic'), ...
    OverlapLength = round(0.02 * fs), ...
    mfcc = true, ...
    mfccDelta = true, ...
    mfccDeltaDelta = true, ...
    pitch = true, ...
    spectralCentroid = true, ...
    zerocrossrate = true, ...
    shortTimeEnergy = true); % Feature extractor

features = extract(aFE, audioIn);

idx = info(aFE);
tFE = linspace(0, length(audioIn) / fs, length(features));

%% Plots
featureSplices = zeros(length(idxSpeech), 4);
detectThresh = 0.05;

for i = 1:length(idxSpeech)
    boundLower = interp1(tFE, 1:length(tFE), idxSpeech(i, 1), 'nearest');
    boundUpper = interp1(tFE, 1:length(tFE), idxSpeech(i, 2), 'nearest');
    feats = features(:, idx.mfccDelta);

    if (tFE(boundUpper) - tFE(boundLower)) <= 0.15 % Skip splices less than 0.2 seconds
        continue
    end

    featureSplices(i, :) = [tFE(boundLower) tFE(boundUpper) ...
                            mean(feats(boundLower:boundUpper))
std(feats(boundLower:boundUpper))];
```

```
%     % To plot all individual mfccDelta of splices against time
%     figure
%     plot(tFE(boundLower:boundUpper), feats(boundLower:boundUpper))
%     grid
%     xlabel('Time (s)')
%     ylabel('Frequency (Hz)')
%     title(['Splice ID ', num2str(i)])

end

%% Detect splices
featureSplices = featureSplices(any(featureSplices, 2), :);

figure
plot(t, audioIn, 'Color', '#0072BD')
axis([0 t(end) -0.3 0.3])
title('Detected Splices - Audio Signal')
xlabel('Time (s)')
grid

detectedSplices = zeros(1, length(featureSplices));

for p = 1:length(featureSplices)

    if p == length(featureSplices)
        break
    elseif abs(featureSplices(p + 1, 3) / featureSplices(p + 1, 4) ...
            - featureSplices(p, 3) / featureSplices(p, 4)) > detectThresh
        detectedSplices(p) = featureSplices(p + 1, 1);
    end

end

detectedSplices = round(nonzeros(detectedSplices), 1);

for k = 1:length(detectedSplices)
    line([detectedSplices(k); detectedSplices(k)], [-0.3; 0.3], 'LineWidth', 2.5, ...
        'LineStyle', '--', 'Color', '#A2142F');
end

figure
melSpectrogram(audioIn, fs)
title('Detected Splices - Mel Spectrogram')

for k = 1:length(detectedSplices)
    line([detectedSplices(k); detectedSplices(k)], [0.0779; 19.8], 'LineWidth', 2.5, ...
        'LineStyle', '--', 'Color', '#FFFFFF');
end

colormap cool
disp(['Total number of audios being spliced: ', num2str(length(detectedSplices))]);
disp(' ');
disp('   Start (s)');
disp(detectedSplices)
```

Total number of audios being spliced: 22

Localization Table

ID	Start Time (s)
1	3.8
2	4.8
3	7.7
4	8.9
5	12.7
6	13.2
7	13.7
8	17.9
9	18.2
10	19.2
11	20.8
12	21.3
13	23.0
14	23.8
15	29.8
16	30.8
17	31.2
18	32.1
19	35.7
20	37.0
21	42.2
22	42.6

7 Limitations of Spectrogram-based Splicing Detection

The proposed method currently faces critical limitations in generalization tasks; mel-spectrogram based splicing detection is best suited for audio files that have magnitude correlation coefficient per speech interval near to each other. When an audio sample fails to meet this criterion, the proposed method marks even the slightest bit of variation in a frequency band as a detected audio splice. To counteract this limitation, a dynamic threshold calculator may be trained using large amounts of data, however, training a deep neural network for the sake of splicing detection itself would yield better results if computational power were not limited.

Another limitation of spectrogram-based audio splicing detection methods is the presence of unbounded noise. Occurrence of even a few noise peaks in the spliced interval can significantly increase the CV value, automatically marking it as a splice. This problem can be alleviated, to an extent, by considering feature maps of the extractor that are independent of noise and further enhancing the comparator pipeline.

8 Conclusion

In this report, a novel method for audio splicing detection and localization has been proposed. The MATLAB® Audio Toolbox™ offers powerful audio feature extraction pipelines, as well as pre-trained deep learning models for detection of characteristics. One such function is that of speech detection, which we utilize to cut out intervals of the original signal into sub signals that we compare against each other. The power dynamics of the speech intervals over time were taken into account by differentiating the mel-frequency cepstral coefficients with respect to time and compared by defining a suitable evaluation criterion.

We conclude that although the accuracy of splice localization is subpar relative to methods utilizing deep neural networks, our proposed approach performs suitably well for when the end-user is limited on computational power. Future work ideas to further enhance the accuracy of the architecture is to compare more than just the mel-frequency cepstral coefficients and take the weighted sum of all the features before passing them into the comparator. A dynamic threshold calculator can also be incorporated after the feature extraction pipeline to improve the performance of the system.