

1 2



9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

**-Licenciatura em Engenharia Informática -
- Análise e Transformação de Dados -
Projeto 2024**

**Identificação de dígitos através de
características extraídas de sinais de
áudio**

Meta I – IV

Trabalho realizado por:
Miguel Castela uc2022212972
Miguel Martins uc2022213951

Índice

Conteúdo

Meta I - IV.....	1
1.1 - Importação dos sinais de áudio.....	3
1.2 - Representação gráfica dos sinais importados	4
1.4 - Escolha das melhores características para a discriminação dos dígitos	9
Meta II.....	11
2.5 - Cálculo normalizado do espectro da amplitude mediano, primeiro e terceiro quartis para cada dígito.....	11
2.6 - Comparação de tipos de janela diferentes.....	12
.....	14
2.7 - Identificação de possíveis características espectrais que permitam diferenciação entre dígitos	14
2.8 - Seleção das características que melhor diferenciam os dígitos	15
Meta III.....	17
3.9 - Cálculo e visualização da STFT (Short-time Fourier Transform)	17
3.10 - Cálculo e display de diferentes características de potencia:	18
3.11- Escolha das três melhores características tempo-frequência	25
Meta IV	28
4.12 - Construção das regras de decisão para a distinção dos dígitos, usando as 3 melhores características de cada meta	28
4.13 - Comparação com os dígitos reais e cálculo da percentagem de acerto.....	30
4.14 - Plot3D com as 3 melhores características	31

1.1 - Importação dos sinais de áudio

Para a importação dos sinais de áudio do **participante 40** foi utilizada a função `audioread()` do MatLab que guarda os valores das amplitudes de cada um dos dígitos na variável `audioData`. No primeiro e segundo exercício, é percorrido um *for loop* no qual é lido o primeiro sinal de áudio do primeiro dígito, feito o seu tratamento, e depois lido o primeiro áudio do dígito seguinte:

```
for i = 0:9
    % Lê o primeiro audio de cada digito
    [audioData, samplingRate] =
    audioread(sprintf("Audios/%d_40_%d.wav", i, 0));
```

`sprintf()` é utilizado como forma de percorrer os ficheiros consoante o nome, sendo que neste caso é aberto sempre o ficheiro '0' do dígito 'i' do participante 40, estando estes na pasta 'Audios'

Para os exercícios 3 e 4, dada a necessidade de percorrer todos os 50 ficheiros de cada dígito, foram utilizados dois *loops*, que funcionam de forma semelhante ao anterior.

```
for digit = 0:9
    for i = 0:49
        % Lê o ficheiro de áudio
        [audioData, samplingRate] =
        audioread(sprintf("Audios/%d_40_%d.wav", digit, i));
```

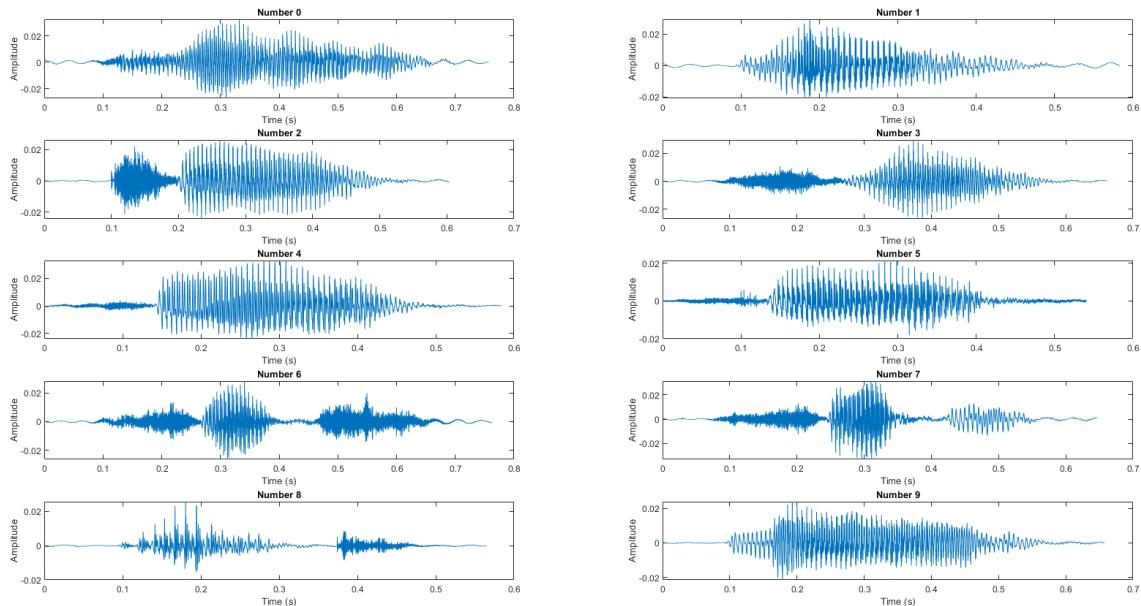


Figura 1 – representação de áudios do participante 40 (0-9), pré-processamento

1.2 - Representação gráfica dos sinais importados

Para a representação gráfica dos sinais utilizados, incluímos aspectos sugeridos no exercício seguinte, como a remoção do silêncio no início e no fim, normalização dos valores com base na amplitude máxima e ao adicionar silêncio no final para os sons ficarem todos com a mesma duração. Isto melhora a representação gráfica dos diferentes dígitos e consequentemente da identificação futura de diferentes características

Para a remoção do silêncio no início e no fim, construímos o método `removeSilence()`, que, dada uma *threshold* estipulada, vai procurar o primeiro valor de amplitude que esteja acima desta *threshold* remover os que estejam à esquerda e vai procurar o último e remover os da direita:

```
function trimmedAudioData = removeSilence(audioData, threshold)

startIndex = find(abs(audioData) >= energyThreshold, 1, 'first');
finalIndex = find(abs(audioData) >= energyThreshold, 1, 'last');

trimmedAudioData = audioData(startIndex:finalIndex);

end
```

Como estamos a tratar de valores discretos de amplitude, não considerámos necessária a criação de janelas de tempo para o cálculo da energia. Como o valor de energia é igual ao quadrado da amplitude, podemos utilizar diretamente os valores da amplitude.

Para a normalização com base na amplitude máxima, construímos o método `normalizeSignal()`:

```
function audioData =
normalizeSignal(audioData) audioData =
audioData ./ max(abs(audioData));
```

Para manter os sinais todos com a mesma duração, construímos o método `fillSilence()`:

```
function audioData = fillSilence(audioData, duration, samplingRate)
    desiredLength = samplingRate / (1/duration);
    audioData = [audioData; zeros(desiredLength - length(audioData), 1)];
end
```

Estipulámos como duração máxima 0.6s, ou seja, $\text{samplingRate}/(1/0.6)$ samples. Assim, definimos que todos os sinais têm de ter aquele número de samples, e caso não tenham, preenche o valor das samples novas com 0's

Depois de feitas as alterações, criamos o vetor de tempo que servirá como eixo das abscissas. Para isto criamos um vetor com $n = n^{\circ}$ samples elementos, a começar em 0. Dividimos pelo samplingRate para converter os índices para tempo

```
time = (0:length(audioData)-1) / samplingRate;
```

Obtemos os resultados apresentados a seguir, após dar `plot` a cada gráfico:

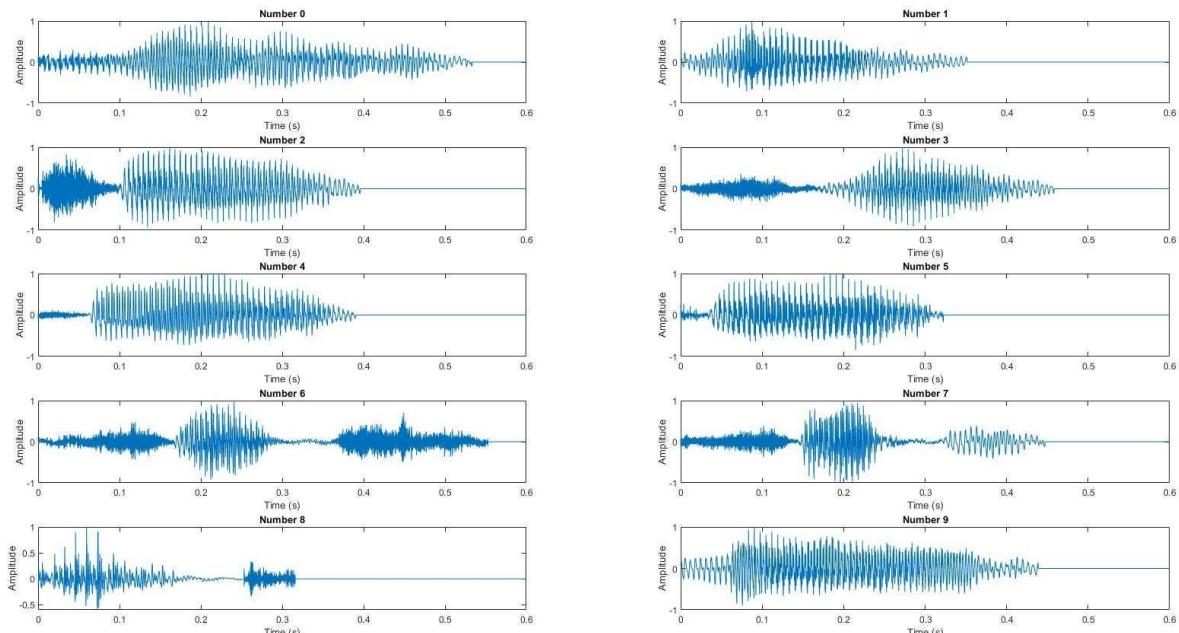


Figura 2 - Representação dos dígitos após tratamento dos dados

1.3 - Implementação de *features* temporais

Foram analisadas as 6 características temporais diferentes, que são:

- Amplitude Máxima
- Desvio Padrão
- Zero-crossing Rate
- Energia total
- Frequência Fundamental
- Intensidade

Amplitude Máxima: Representa o volume máximo do sinal de áudio. Diferentes dígitos podem ser pronunciados com diferentes níveis de ênfase ou volume. Por exemplo, alguém pode dizer "nove" com mais força do que "um". Portanto, a amplitude máxima pode ajudar a capturar essas variações no volume entre diferentes dígitos.

Desvio Padrão: O desvio padrão mede a dispersão dos pontos de dados em torno da média. No contexto do áudio, pode indicar a variabilidade no tom ou frequência dos dígitos falados. Diferentes dígitos podem ter padrões de tom diferentes, o desvio padrão pode ajudar a capturar essas variações no tom.

Zero-crossing Rate: Esta é a taxa na qual o sinal de áudio muda de sinal (ou seja, cruza a linha de amplitude zero) dentro de um determinado período. Pode ser indicativo da taxa de mudança e da frequência de transições no sinal de áudio. A taxa de cruzamento por zero pode ajudar a capturar taxas de variação, nos diferentes dígitos, na transição entre amplitudes negativas e positivas nos padrões de fala.

Energia Total: A energia total representa a magnitude geral dos sinais de áudio. Pode estar relacionada à duração e intensidade do áudio do dígito. Esta característica pode ajudar a diferenciar entre dígitos com base na sua força ou intensidade geral.

Frequência Fundamental: Refere-se ao componente de frequência mais baixa em forma de onda periódica. Na fala, corresponde à frequência mais baixa da vibração das cordas vocais. Diferentes dígitos podem ter frequências fundamentais diferentes devido a variações na pronúncia ou características vocais, então, a frequência fundamental pode ajudar a capturar essas diferenças de tom entre os dígitos.

Intensidade: A intensidade representa a potência (força) do sinal de áudio por área unitária. Pode estar relacionada ao volume percebido do dígito falado. Semelhante à amplitude máxima, a intensidade pode capturar variações no volume entre diferentes dígitos, pois dígitos pronunciados com mais intensidade podem ter valores de intensidade mais altos. Esta característica pode ajudar a diferenciar entre dígitos com base no seu volume percebido.

Ao analisar essas seis características temporais, podemos capturar vários aspectos do sinal de áudio associados a diferentes dígitos falados, como variações de volume, padrões de tom, energia da fala e dinâmica da fala, o que pode ser útil para distinguir entre os dígitos 0-9.

Para cada uma destas características, foi criado uma matriz (10,50) inicializado com 0's, que vai depois conter o valor resultante da respetiva análise, para cada som e para cada dígito, como por exemplo:

```
maxAmplitudes = zeros(10, 50);
```

Para o cálculo de cada uma destas características, é percorrido cada áudio individualmente, da forma mencionada no ponto [1.1 - Importação dos sinais de áudio](#), calculados todos os valores e guardados no vetor correspondente.

- A amplitude máxima é dada pelo maior valor de audioData:

```
maxAmplitudes(digit+1, i+1) = max(abs(trimmedAudioData));
```

- O desvio padrão é calculado com a função std() existente no MatLab:

```
standardDeviation(digit+1, i+1) = std(trimmedAudioData);
```

- A energia, visto que estamos a trabalhar com valores discretos, é dada pelo quadrado do valor da amplitude de cada elemento, dado por:

```
energy(digit+1, i+1) = sum(trimmedAudioData.^2);
```

- O zero-crossing rate, ou taxa de mudança de sinal, é calculada ao converter todos os valores para o seu sinal (-1,0 ou 1) sign(), calcular a diferença entre valores seguidos diff(), transformar no valor absoluto abs(), fazer a soma sum() dos valores resultantes, sendo que cada 2 corresponde a uma passagem, dividir por 2 para obter o valor total de passagens e por fim dividir pela duração para obter a taxa:

```
zeroCrossingRates(digit+1, i+1) =  
sum(abs(diff(sign(trimmedAudioData)))) / (2 *  
length(trimmedAudioData) / samplingRate);
```

- A frequência fundamental é calculada ao obter a média do tom de cada sample:

```
fundFreq(digit+1, i+1) = mean(pitch(trimmedAudioData,  
samplingRate));
```

- A intensidade ou loudness, obtida pela função *integratedLoudness()*:

```
loudness(digit+1, i+1) =  
=integratedLoudness(audioData, samplingRate);
```

É de notar que, para os pontos 1-5 utilizamos o *trimmedAudioData* para os dados, ou seja, áudios com o silêncio inicial e final removidos. No entanto, para a intensidade, isto fazia com que alguns sons ficassem demasiado curtos para a função usada, ou seja, *integratedLoudness()* devolvia valores nulos. Devido a isso, utilizámos os dados originais, sem *trim*. A normalização afetaria os valores de todos estes dados à exceção do zero-crossing rate, pelo que não foi utilizada de todo.

O plot é feito na função *plotScatterplot* que recebe as matrizes e labels como argumentos

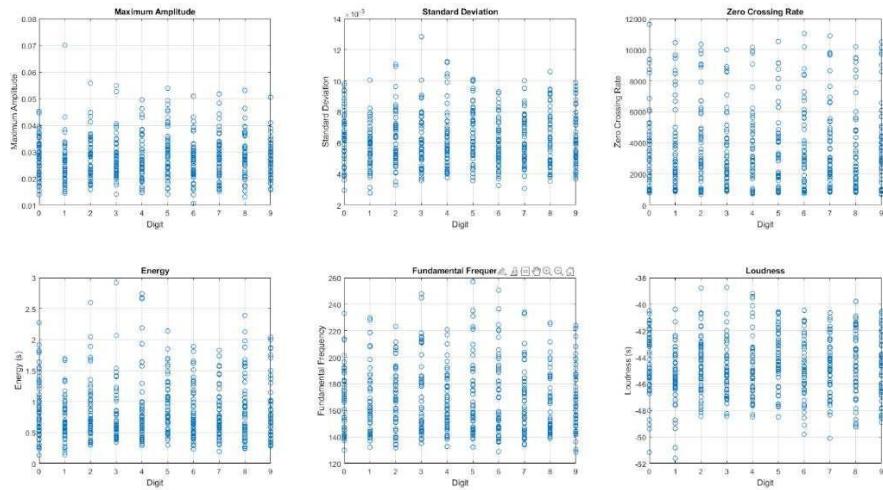


Figura 3 -representação gráfica das características temporais

Para o plot dos gráficos, criamos a matriz *matrix[10,50]* que é utilizada para ter os valores correspondentes ao número em questão:

```
matrix = repmat(0:9, 50, 1) %criação matriz
```

Exemplo de plot para a Amplitude Máxima:

```
Subplot(2, 3, 1);
plot(reshape(matrix, [], 1), maxAmplitudes(:, 'o');
%boxplot(maxAmplitudes, 'Labels', arrayfun(@num2str, 0:9,
'UniformOutput', false)); (presente numa função à parte)
xlabel('Digit');
ylabel('Maximum Amplitude');
title('Maximum Amplitude');
set(gca, 'XTick', 0:9); grid
on;
```

1.4 - Escolha das melhores características para a discriminação dos dígitos

Com base nos resultados anteriores, identificamos que as características que mais diferem entre os dígitos são a **energia total**, a **taxa de mudança de sinal** e a **frequência fundamental**. Estas características foram selecionadas pela sua capacidade de melhor discriminar entre os diferentes dígitos, isto porque os valores para os diferentes dígitos diferem mais nestas características do que na amplitude Máxima, Desvio Padrão e Intensidade. Verifica-se também que na **energia total**, apesar dos dígitos 1,2 e 8,9 apresentarem valores semelhantes é fácil distinguir o dígito 4 do 5 do 10 pelos níveis de energia totais que estes apresentam. Na **taxa de mudança de sinal** houve uma forte distinção de dígitos a partir dos valores, então, apesar dos dígitos 1,3, 5, 6 e 9 terem valores semelhantes os outros dígitos variam muito, tendo valores muito dispersos e característicos, capazes de identificar unicamente cada um. Na **frequência fundamental** os dígitos 4, 5, 8 e 10 apresentam valores muito diferentes, sendo possível identificar estes dígitos a partir dos seus valores de frequência fundamental.

Isto foi verificado pela observação dos gráficos, e então, para distinguir que características são as melhores. O plot é feito na função `plotBoxplot` e `scatter3` que recebe as matrizes e labels como argumentos, resultando na apresentação gráfica seguinte:

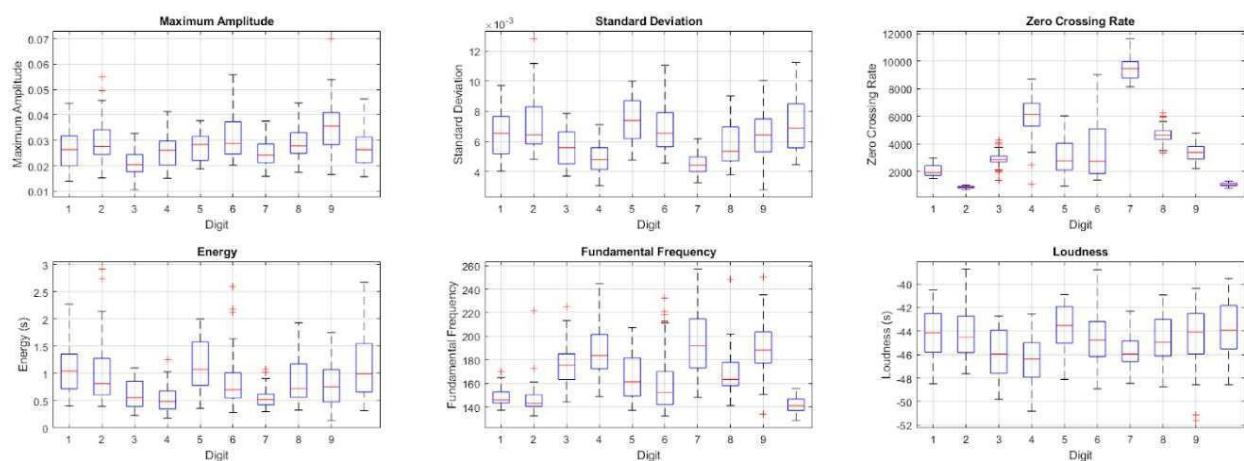


Figura 4 - representação gráfica das características temporais (em boxplot)

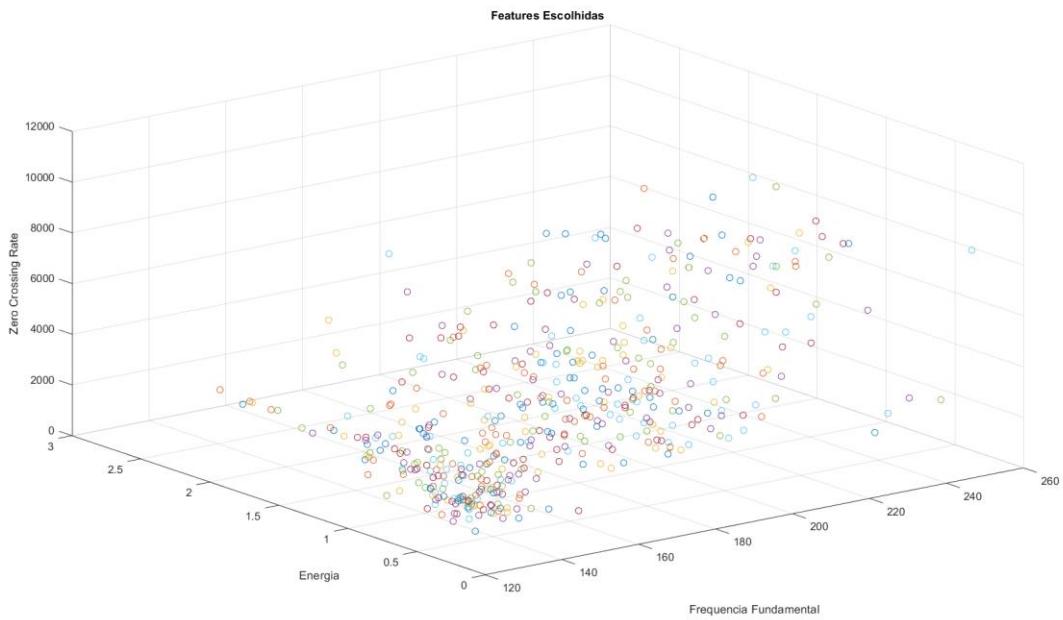


Figura 5 - scatterplot 3D das 3 melhores características da meta I (cada dígito tem uma cor diferente)

Meta II

2.5 – Cálculo normalizado do espectro da amplitude mediano, primeiro e terceiro quartis para cada dígito

Para obtermos os espectros de amplitude mediano de cada dígito, normalizado pelo número de amostras, recorremos ao módulo dos coeficientes da série complexa de Fourier (equivalente, e apenas para números positivos). Isto é, os C_m dados.

```
% Compute the Fourier transform
X = fft(audioData);

%Get only the first half, since its symmetrical
X = X(1:floor(length(X)/2));
% Compute the absolute value of the Fourier coefficients
X = abs(X);
% Normalize by the number of samples (median amplitude spectre,
% normalized by the number of samples)
amplitude_spectrum = X / length(audioData);
```

Recorremos também ao cálculo da mediana, do primeiro e do terceiro quartil dos valores absolutos dos coeficientes da série complexa de Fourier desta maneira:

```
% The result is a 1xfreqNx10 matrix. To reshape it into a freqNx10 matrix:
medians = squeeze(medians);

% Calculate the first and third quartile for each frequency
first_quartile = quantile(amplitudes(:,:,:,i+1), 0.25, 1);
third_quartile = quantile(amplitudes(:,:,:,i+1), 0.75, 1);
```

Estes cálculos foram feitos com base nos 50 exemplos de cada dígito, normalizados pelo número de amostras. Esta normalização é importante pois quanto mais amostras existirem, mais representações poderá haver para uma determinada frequência, uma vez que há mais dados para serem considerados. Ao normalizar pelo número de amostras, estamos basicamente a calcular a média das contribuições de cada amostra para essa frequência específica.

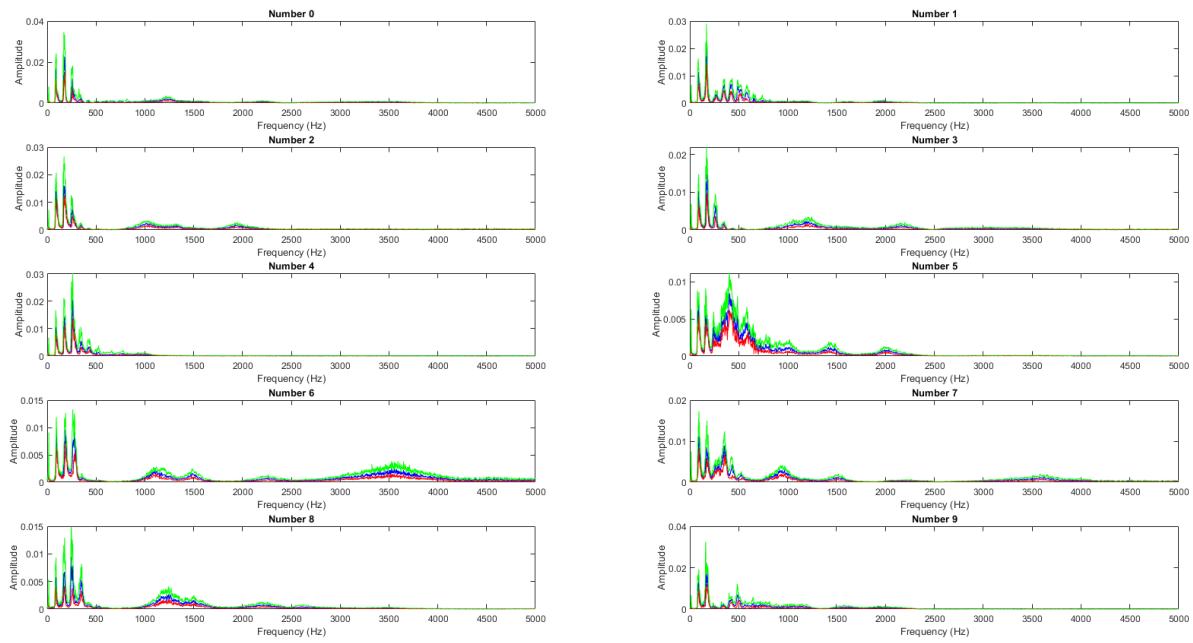


Figura 6 – representação gráfica da amplitude mediana, primeiro e terceiro quartis para cada dígito

- média
- primeiro quartil
- terceiro quartil

2.6 – Comparação de tipos de janela diferentes

É impossível computar a transformada de Fourier no seu intervalo ($(-\infty, +\infty]$) , isto porque os computadores só processam intervalos finitos de valor (equivalente a calcular a transformada de Fourier de um sinal multiplicado por uma função retangular). Isto traduz-se, no domínio da frequência, para a convolução do sinal com uma função retangular.

Devido a isto, os sinais acabam por ser cortados nas suas extremidades. Isto causa um fenômeno denominado de spectral leakage, onde os lobos laterais são mais altos do que o esperado

◆ Escolha da Janela

$$y = \sum_{k=1}^{\infty} a_k \sin(w_k t)$$

- Para garantir resolução:
 - Largura espectral da janela ↓
 - Lobos laterais devem ser baixos

As funções de janelas são utilizadas para gerir efeitos específicos, proporcionando controlo sobre estes. As duas janelas escolhidas, para além da retangular, para a comparação foram as de Hamming e Blackman (a janela Hann não foi escolhida pois demonstrava pouca diferença experimental nos resultados da janela de Hamming obtidos), obtendo os seguintes resultados:

Sem janela

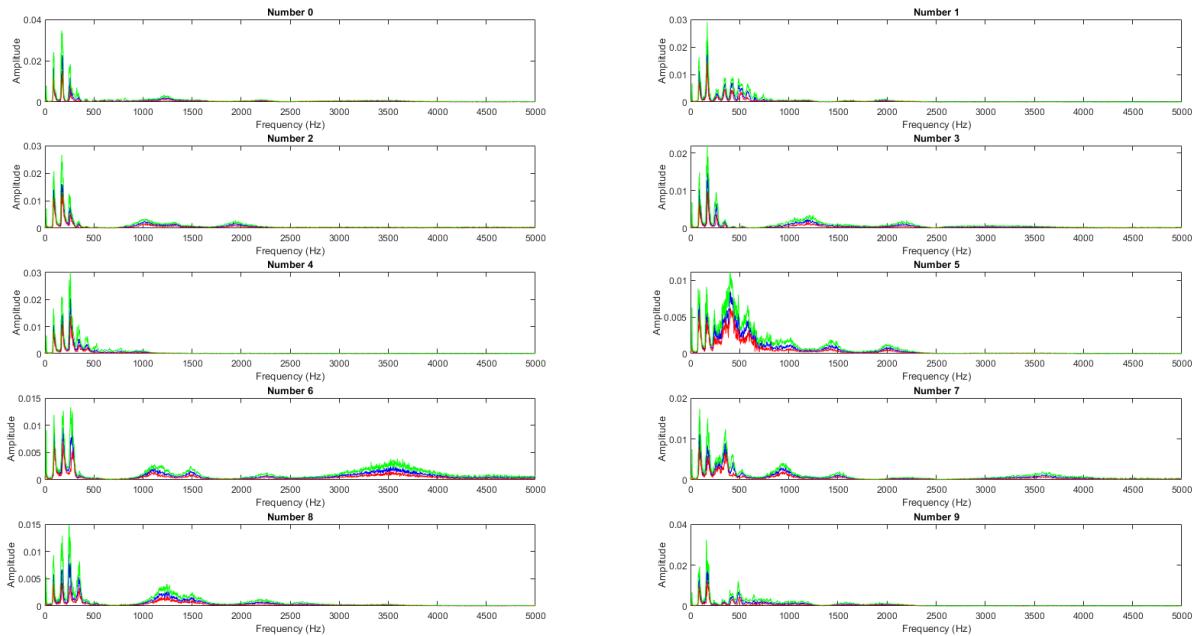


Figura 7 - Representação sem janela

Hamming

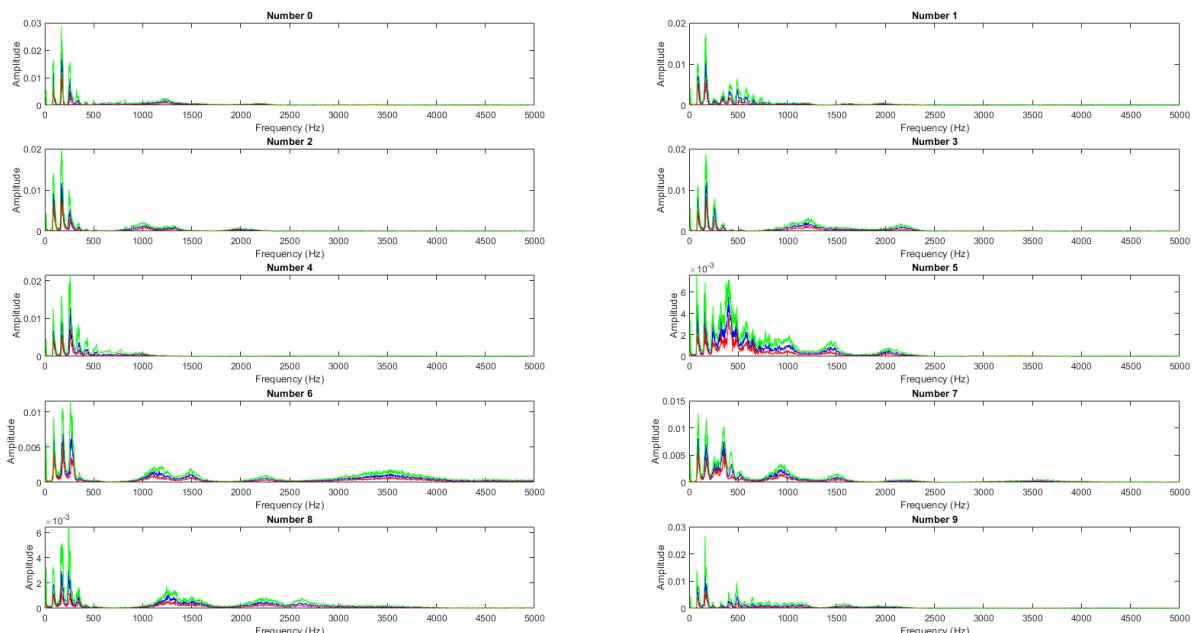


Figura 8 – Representação da janela de Hamming

Blackman

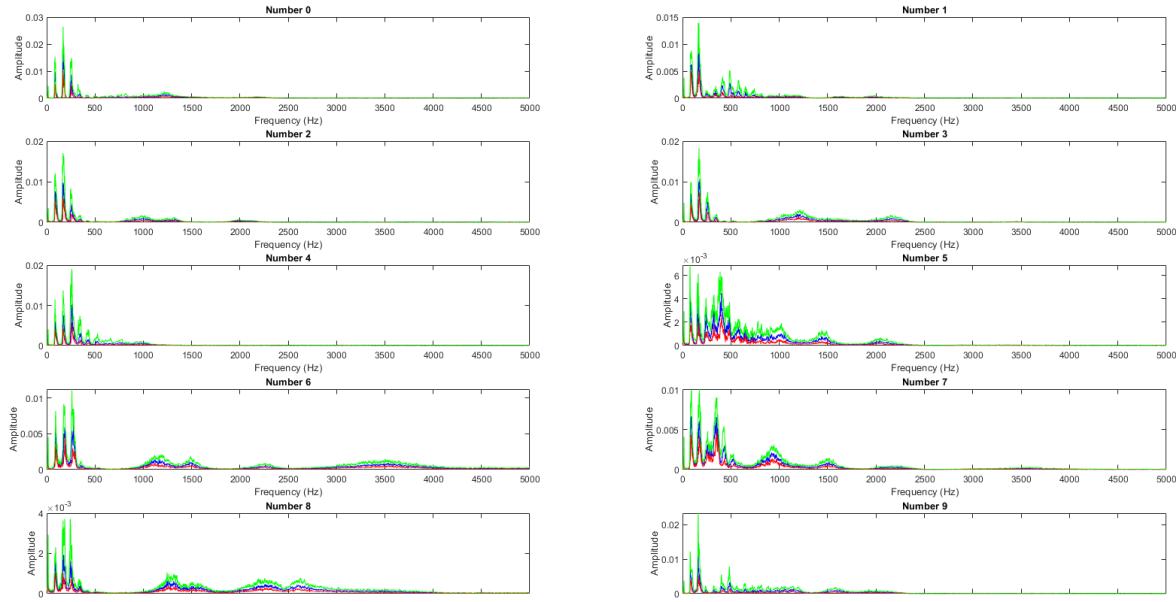


Figura 9 – Representação da janela Blackman

- média
- primeiro quartil
- terceiro quartil

Como se pode observar, os valores dos gráficos não diferem significativamente. Isto mostra que, apesar de as janelas diminuírem o spectral leakage e os lobos laterais, essa mudança ainda é pouco notável nos valores da mediana e dos quartis.

2.7 - Identificação de possíveis características espectrais que permitam diferenciação entre dígitos

Foram escolhidas 6 características espectrais, calculadas com base nos ficheiros fornecidos, após a obtenção da matriz com os valores das amplitudes espectrais.

Média das amplitudes espectrais: Foi calculado o valor médio das amplitudes espectrais para cada áudio (ponto [2.5](#)), através da função `mean()`.

Mediana dos picos: A função `findpeaks()` devolve os picos, ou seja, valores e localização em que uma amplitude é maior que ambos os seus vizinhos diretos. Para o caso apenas precisamos dos valores, e ao calcular a mediana destes, do pico de cada áudio, obtemos a matriz com esses valores

Spectral Skewness: Valor que traduz a simetria da distribuição das frequências. É calculada a média e o desvio padrão dos valores das amplitudes para se obter o valor da simetria através da fórmula: $E[(X - \mu)^3] / \sigma^3$.

Spectral edge frequency: Consiste no valor que delimita uma certa percentagem de frequências menor que o próprio, no caso 90% das frequências. Primeiro é obtido a *power spectral density* que consiste na forma como a potência do sinal está distribuída pelas frequências, através da função `pwelch()`. Depois é calculada e normalizada a soma cumulativa dos valores, e por fim obtido o valor da *spectral edge frequency* através da função `interp1()`.

Spectral Centroid: Como se fosse o ‘centro de massa’ do espectro, calculado pela média ponderada das frequências presentes no sinal.

Spectral Spread: Forma como estão dispersos os valores em torno do *centroid*. É calculado o *centroid*, de forma referida anteriormente, e a dispersão através da fórmula: $\text{sqrt}(\text{E}[(X - \mu)^2])$.

O plot é feito na função `plotScatterplot` que recebe as matrizes e labels como argumentos

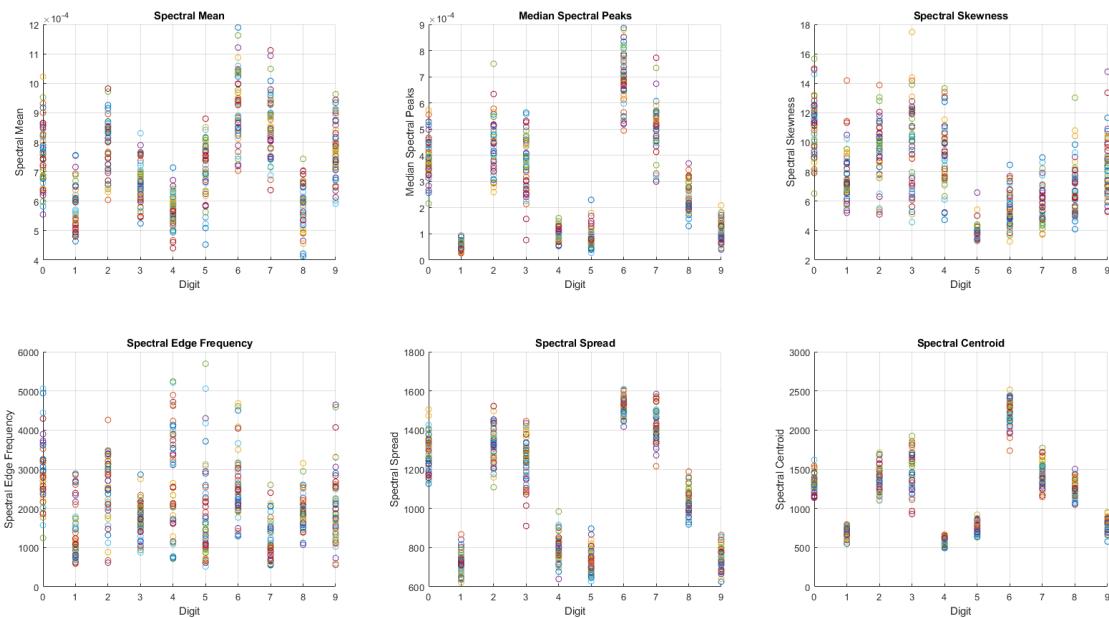


Figura 10 - scatterplot das diferentes características espectrais que permitam diferenciação entre dígitos

2.8 - Seleção das características que melhor diferenciam os dígitos

Como 3 melhores características selecionámos: **Média das amplitudes espectrais**, **Spectral Skewness** e **Spectral Spread**. A dispersão nestes dígitos é maior e mais evidente que nas restantes características. (Spectral Centroid não diverge muito de Spectral Spread dada a elevada informação mútua entre ambos).

No que toca à **spectral skewness**, apesar de os dígitos 2,3, e 4 terem valores similares, facilmente se faz a distinção nos valores de 5-9 e 0.

Quanto à **média das amplitudes espectrais**, os dígitos 1,4,8 e 0,9 apresentam valores semelhantes, respetivamente, mas existe alguma dispersão no que toca aos valores dos outros dígitos.

Por fim, os valores de **spectral spread**, mostram semelhança nos dígitos 1,4,5,9 e 0,2, mas fácil distinção nos valores 3,6,7,8.

O plot é feito na função `plotBoxplot` que recebe as matrizes e labels como argumentos, bem como usando `scatter3`.

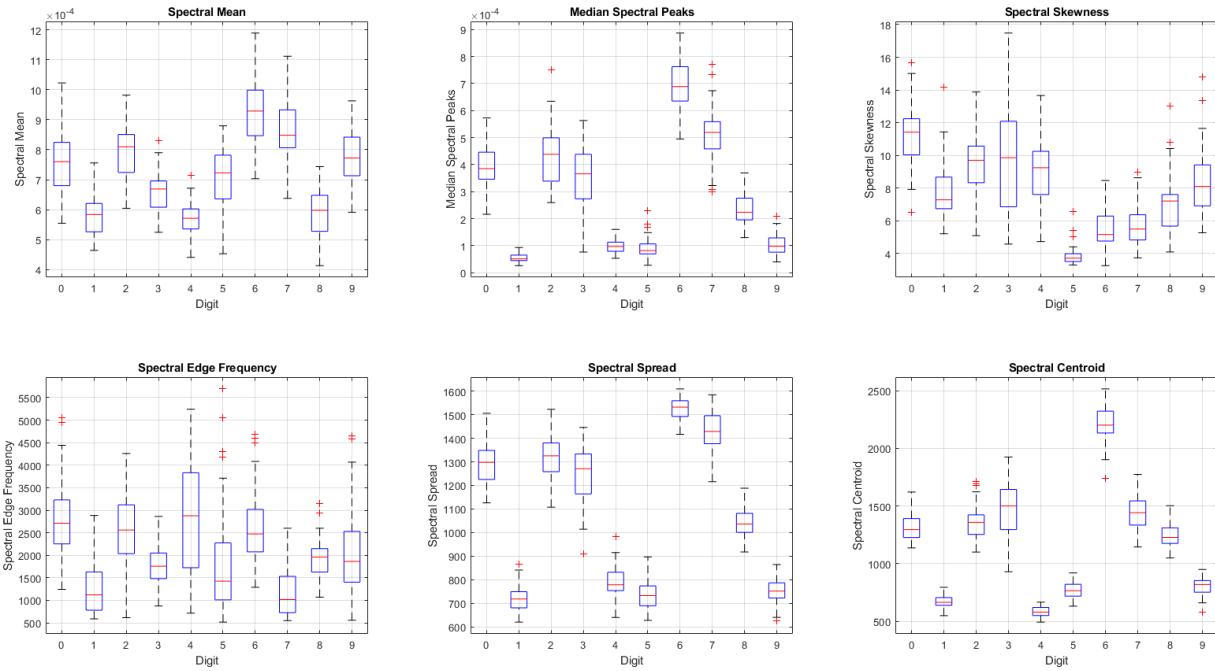


Figura 11 – representação em boxplot das características para selecionar as que melhor diferenciam os dígitos

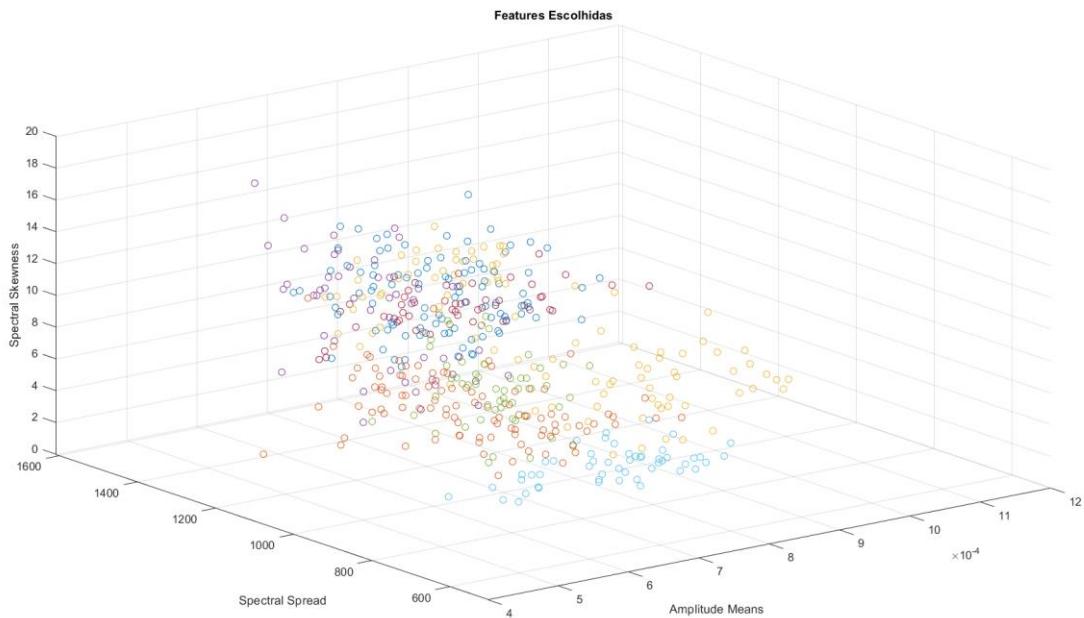


Figura 12 - scatterplot3d das 3 melhores características da meta II (cada dígito tem uma cor diferente)

Meta III

3.9 – Cálculo e visualização da STFT (*Short-time Fourier Transform*)

Nesta meta vão ser calculadas e selecionadas características de potencias em diferentes janelas tempo-frequênciá.

Para o cálculo da STFT, começamos por fazer pré-processamento dos áudios ao fazer a remoção de silencio inicial e final bem como normalização dos áudios. Aliás não foi adicionado silencio no final, visto que tornava todos os valores a 0, o que criava uma “faixa” indesejada na espectrograma:

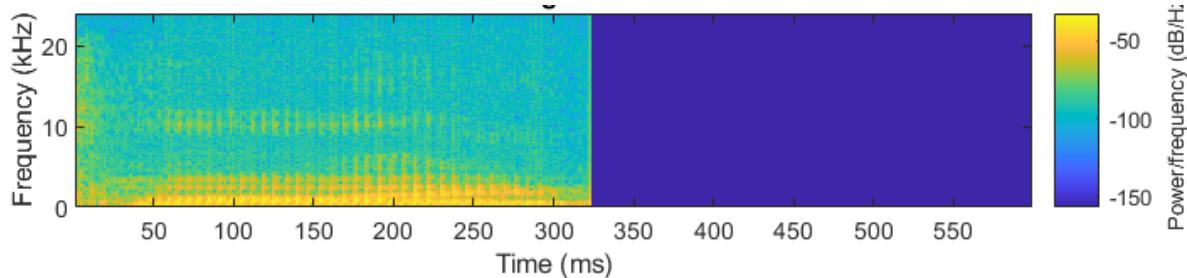


Figura 13 – espectrograma com remoção de silencio

Utilizando a função `spectrogram()` do MATLAB, obtemos uma matriz de magnitudes, na qual as linhas representam janelas de frequênciá, e as colunas de janelas de tempo.

Após testar vários valores de `windowSize` para o número de overlaped samples e número de samples (`nfft`), obtivemos os seguintes exemplos de resultados:

Valor de `nfft` baixo:

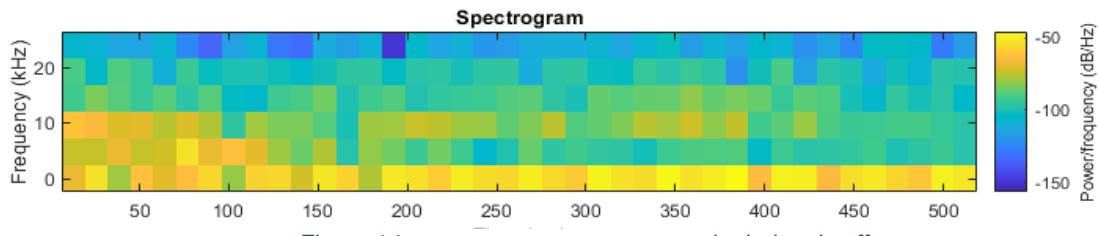


Figura 14 – espectrograma com um valor baixo de nfft

Valor de overlap alto:

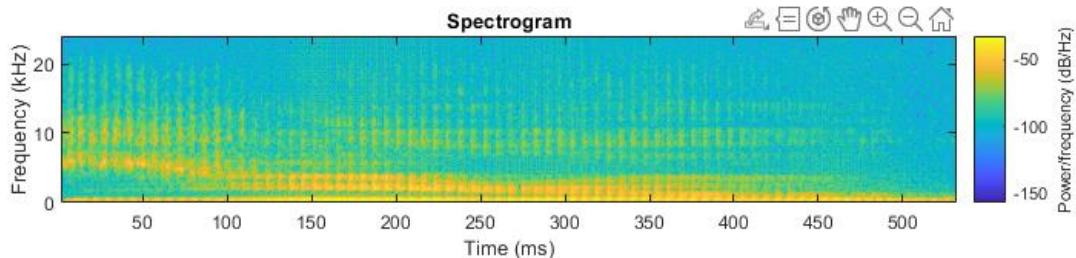


Figura 15 – espectrograma com um alto valor de overlap

Usámos sempre janelas de Hamming pois, após testar com as janelas possíveis da meta II, esta foi a que obteve resultados mais coerentes com o esperado.

Como valores finais escolhidos, optamos por definir o valor do overlap como metade do valor da janela, obtendo transições suaves sem necessitar de grande esforço computacional para o seu cálculo. O que resultou nos espectrogramas seguintes:

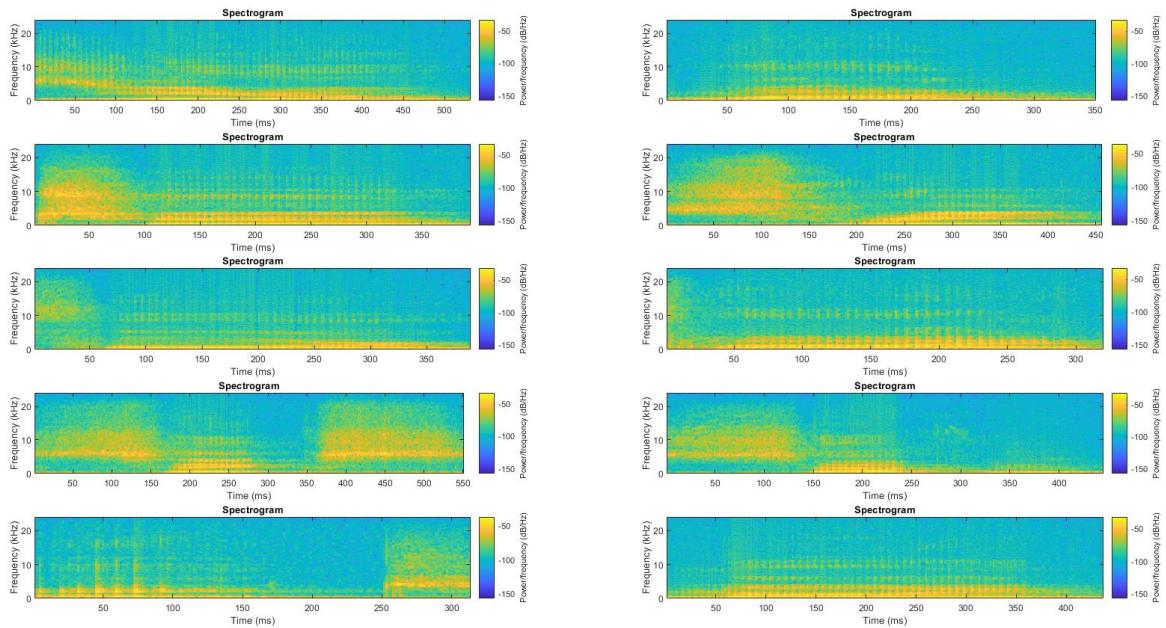


Figura 16 – espectrogramas finais para cada digito

3.10 – Cálculo e display de diferentes características de potencia:

Para esta meta selecionámos 11 características de potência:

- Mean Power Frequency Band
- Mean Power Time Band
- Spectral Band Energy per Time Band
- Spectral Contrast per Time Band
- Spectral RollOff per Time Band
- Spectral Flatness per Time Band
- Peak Power per Time Band
- Power per Time Window
- Spectral Flux per Time Band
- Power Standard Deviation per Time Band
- Frequency Standard Deviation per Time Band

Descrição breve e representação gráfica (em 3 dimensões) das características:

Mean Power Frequency Band – média ponderada dos componentes de frequência do sinal, calculada com a média da energia por banda de tempo.

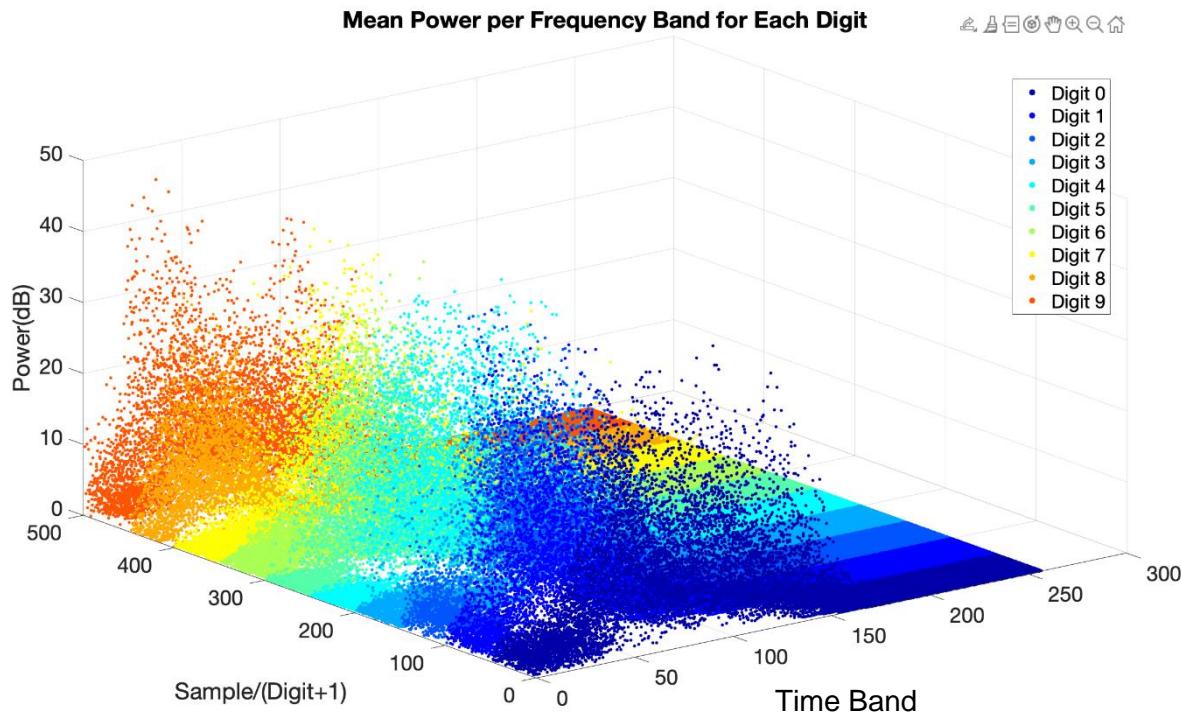


Figura 17 - Plot3D dos valores MPFB de cada áudio, para cada dígito

Mean Power Time Band – Potência média por cada janela de tempo, e é dada pela média do cálculo da energia.

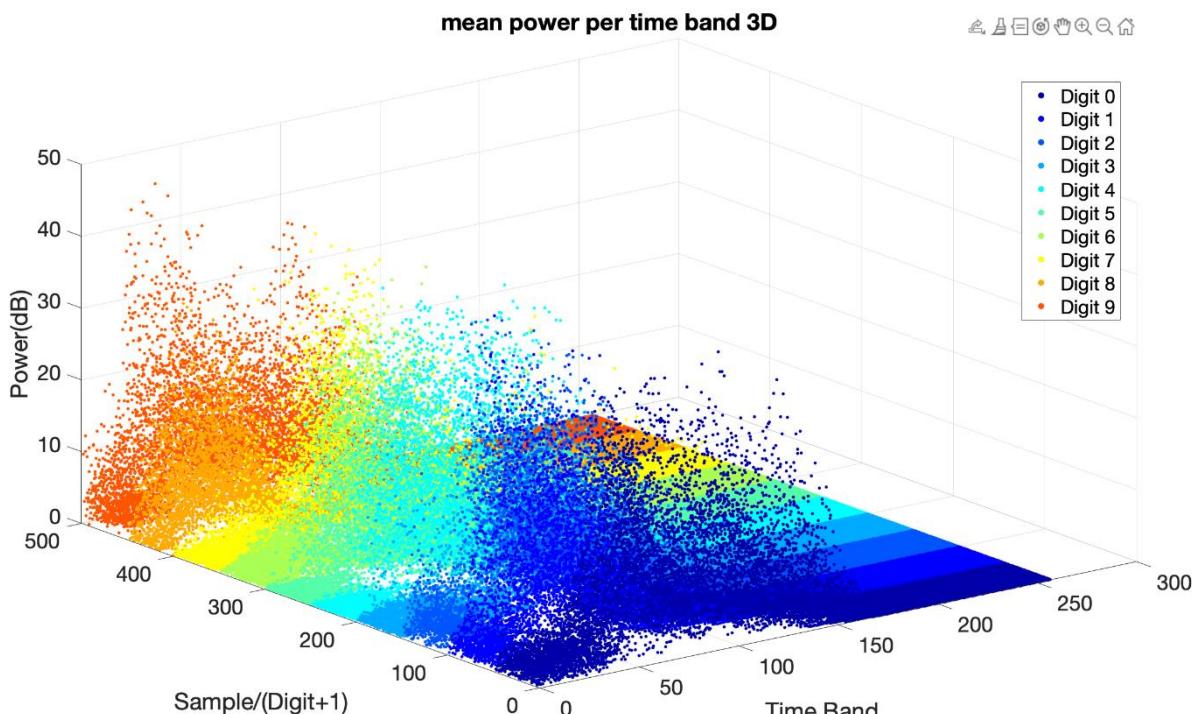


Figura 18 – Plot dos valores MPTB de cada áudio, para cada dígito

Spectral Band Energy per Time Band – Energy de um sinal para um certo valor de frequência e janela de tempo.

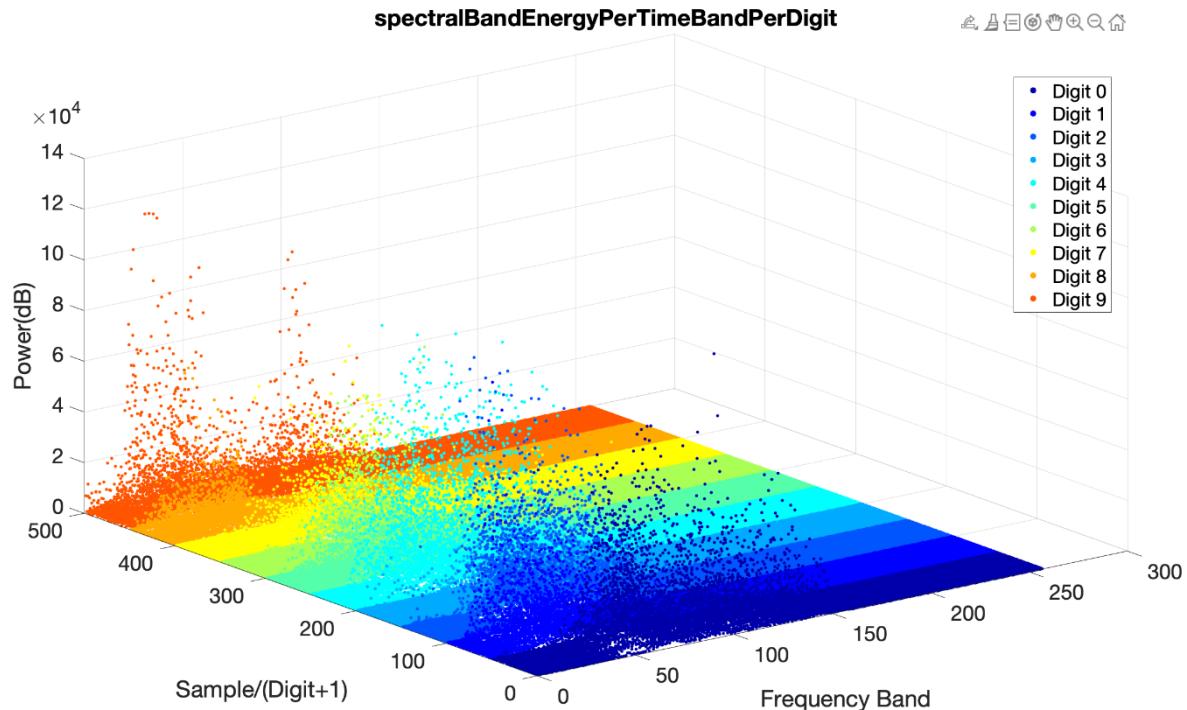


Figura 19 - Plot3D dos valores SBTB de cada áudio, para cada dígito

Spectral Contrast per Time Band – Mede a diferença entre os picos mais prominentes e os vales menos prominentes. Calculada com a função `max()` com `spectrum` para o calculo dos picos, e com `1/spectrum` para o calculo dos vales.

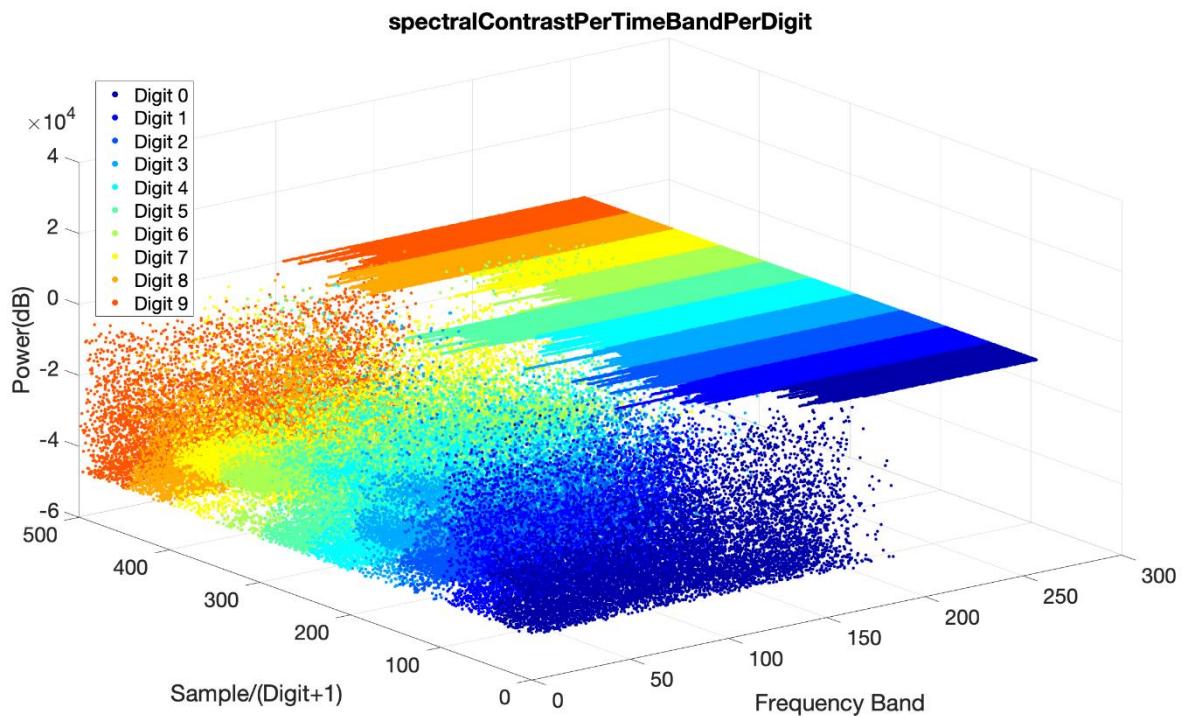


Figura 20 - Plot3D dos valores SCTB de cada áudio, para cada dígito

Spectral Rolloff per Time Band – Frequência para a qual uma certa de percentagem (no caso 85%) das frequências é menor, calculada ao procurar a partir de que ponto na soma cumulativa, o valor excede os 85%

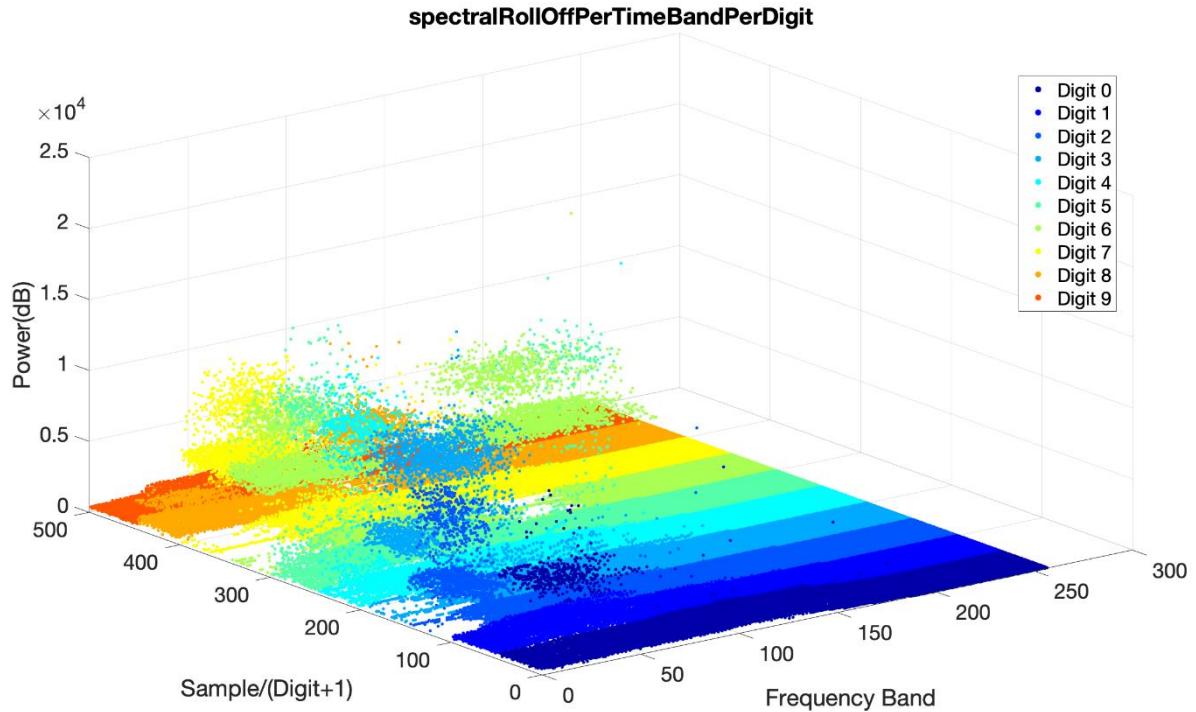


Figura 21 - Plot3D dos valores SROTB de cada áudio, para cada dígito

Spectral Flatness – (ou Entropia de Wiener) É a média geométrica sobre a média aritmética:

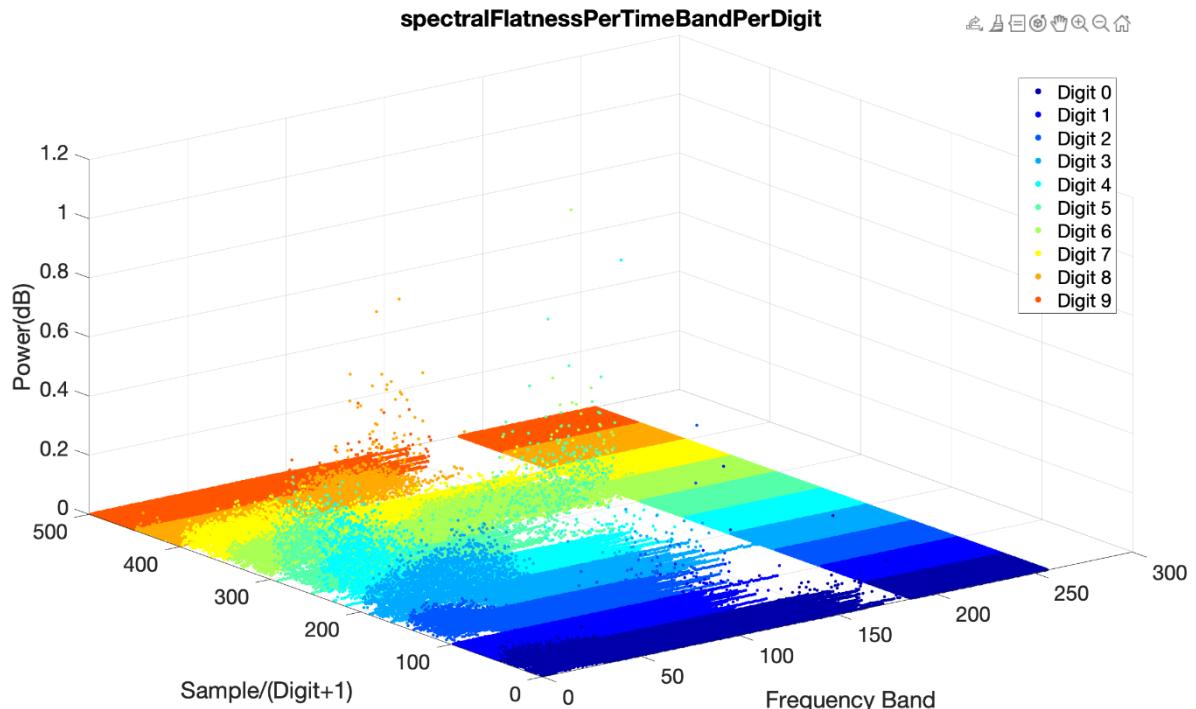


Figura 22 - Plot3D dos valores SFTB de cada áudio, para cada dígito

Peak Power per Time Band – Valor máximo de potência, por intervalo de tempo.

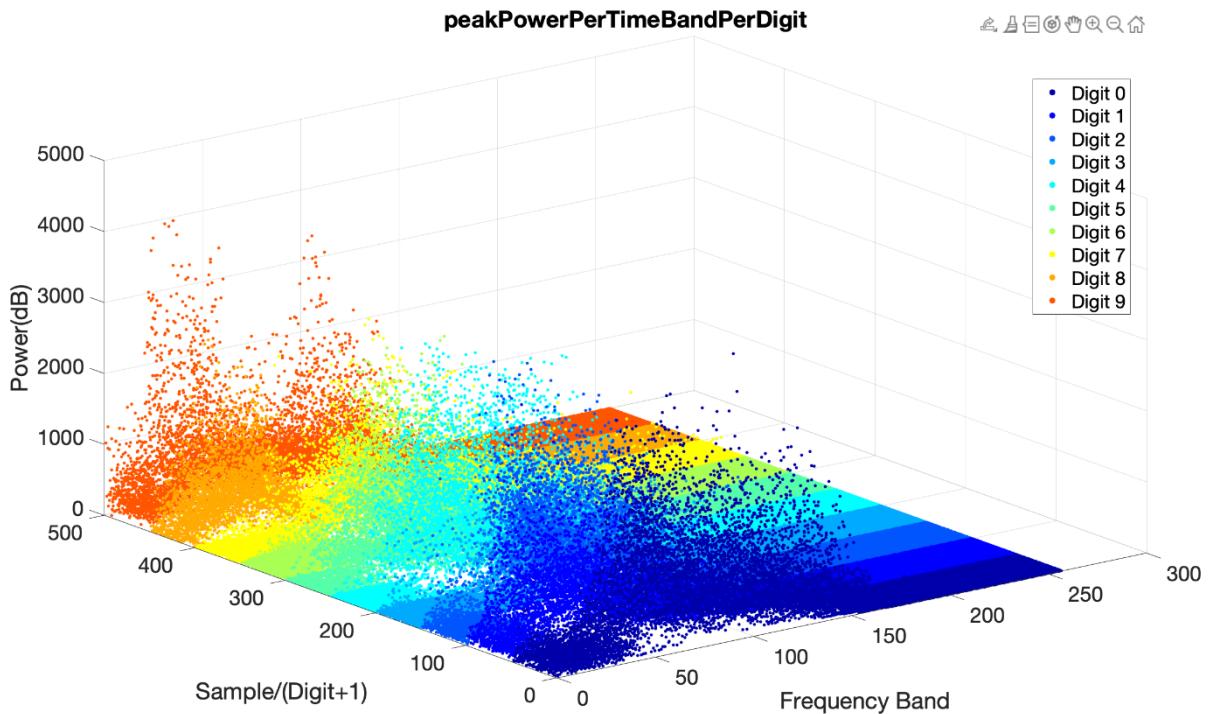


Figura 23 - Plot3D dos valores PPTB de cada áudio, para cada dígito

Power per Time band – Valor da potência total, por intervalo de tempo.

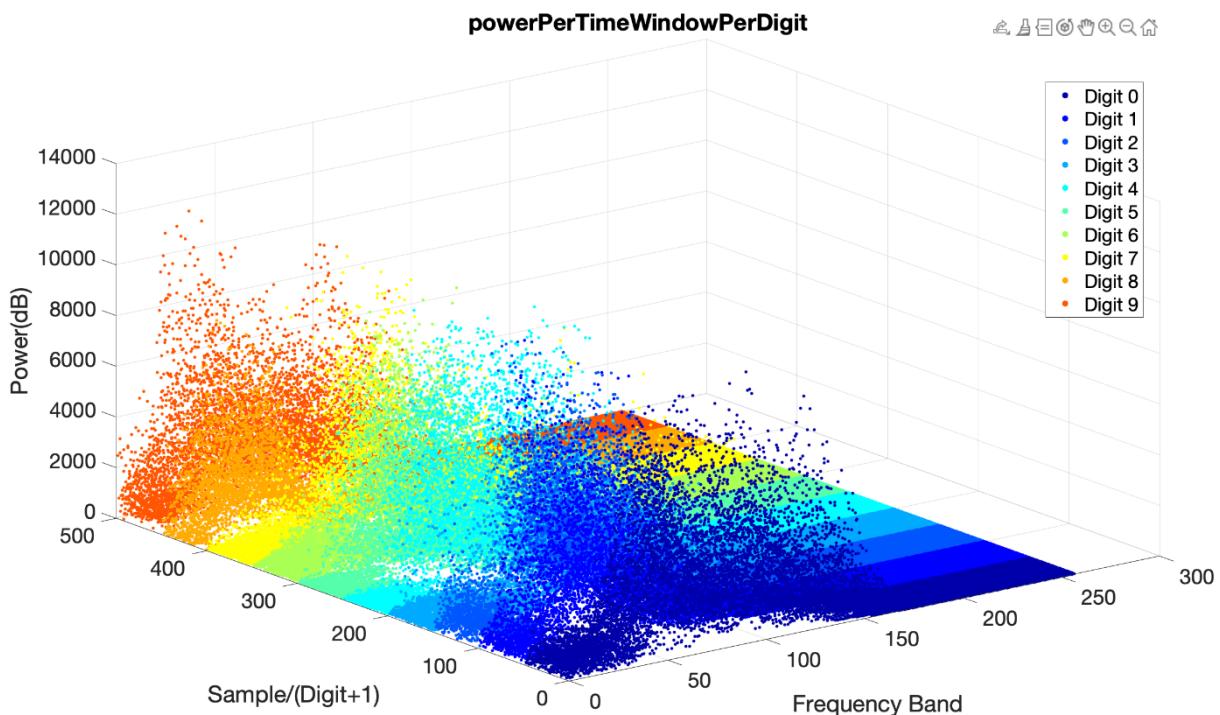


Figura 24 - Plot3D dos valores PPTW de cada áudio, para cada dígito

Spectral Flux per Time Band – Forma como variam os valores de potência, por intervalo de tempo

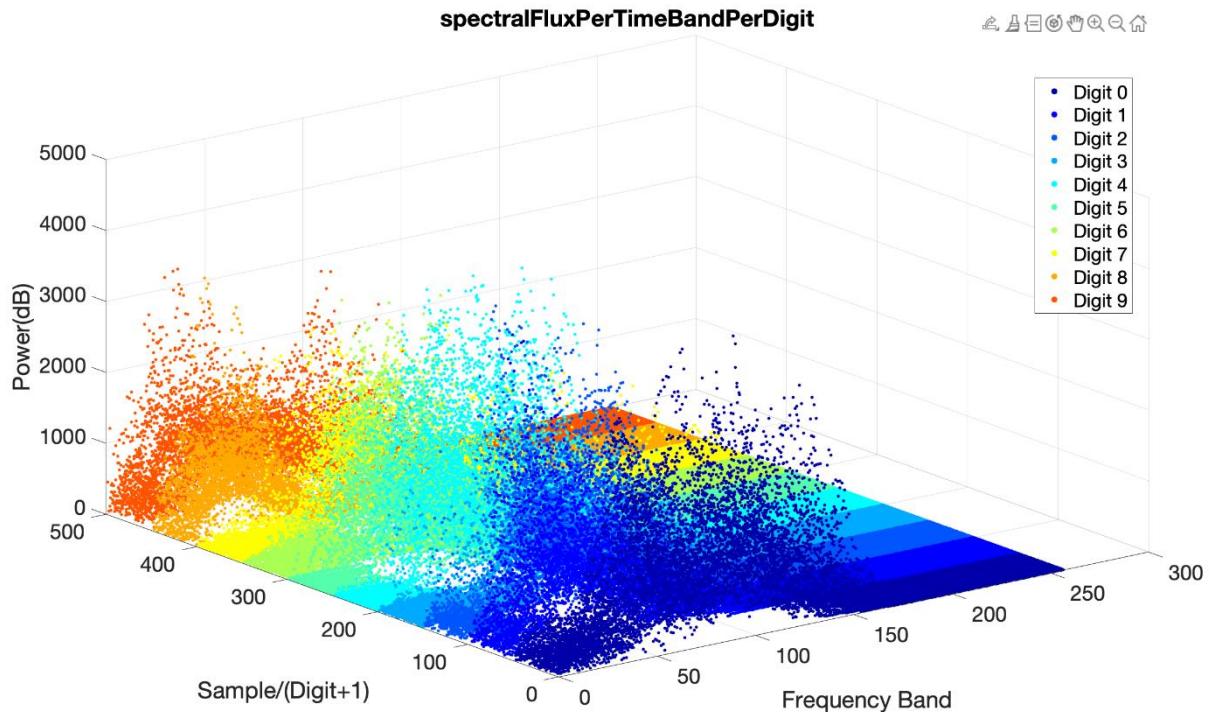


Figura 25 - Plot3D dos valores SFTB de cada áudio, para cada dígito

Power Standard Deviation per Time Band – Valor do desvio padrão da potência, por intervalo de tempo

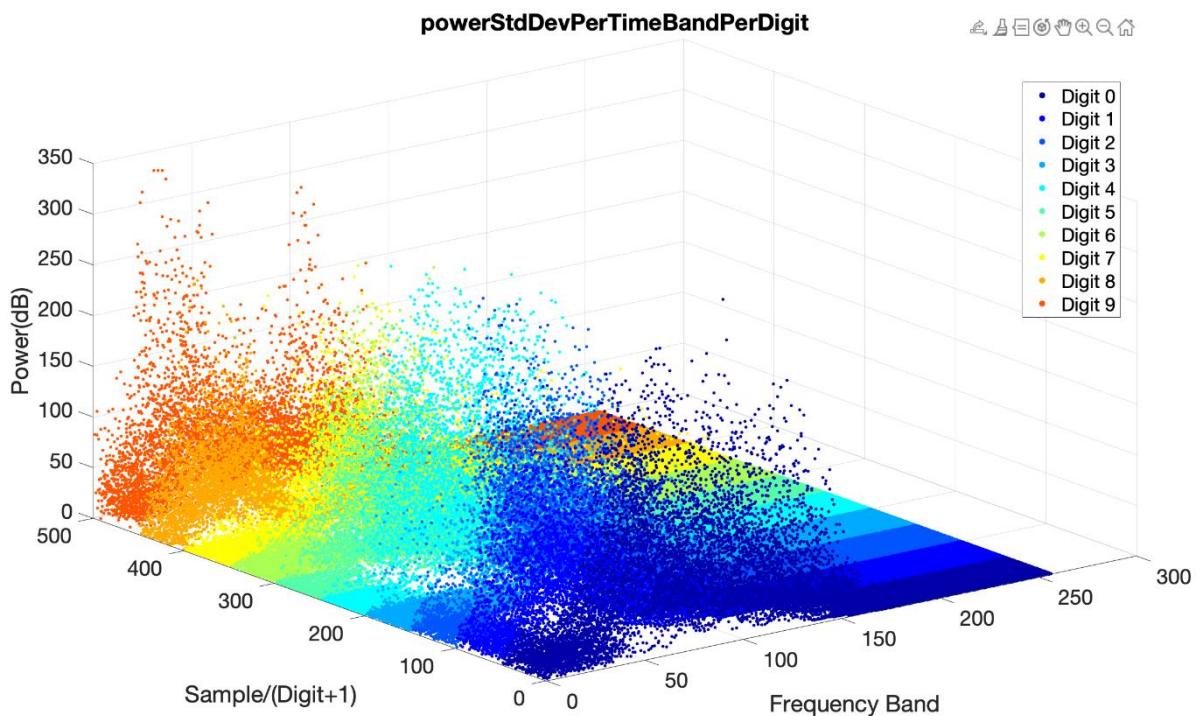


Figura 26 – Plot3D dos valores PStdTB de cada áudio, para cada dígito

Frequency Standard Deviation per Time Band – Valor do desvio padrão da frequência, por intervalo de tempo.

Plot3D dos valores de cada áudio, para cada dígito:

`frequencyStdDevPerTimeBandPerDigit`

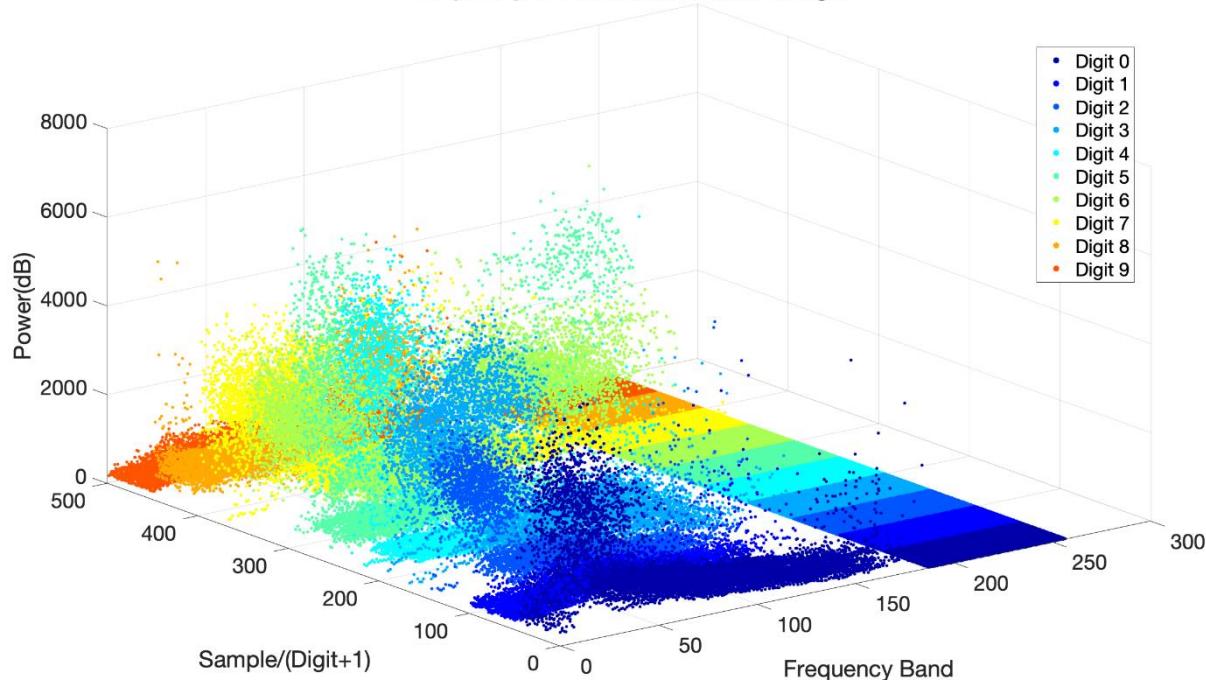


Figura 27- Plot3D dos valores FStdDTB de cada áudio, para cada dígito

Fizemos também gráficos 2D para todas as características, têm, porém, menor relevância para a nossa escolha dos cálculos do boxplot, sendo apenas bons para ver em que janelas de tempo há uma maior discrepância de valores para cada dígito:

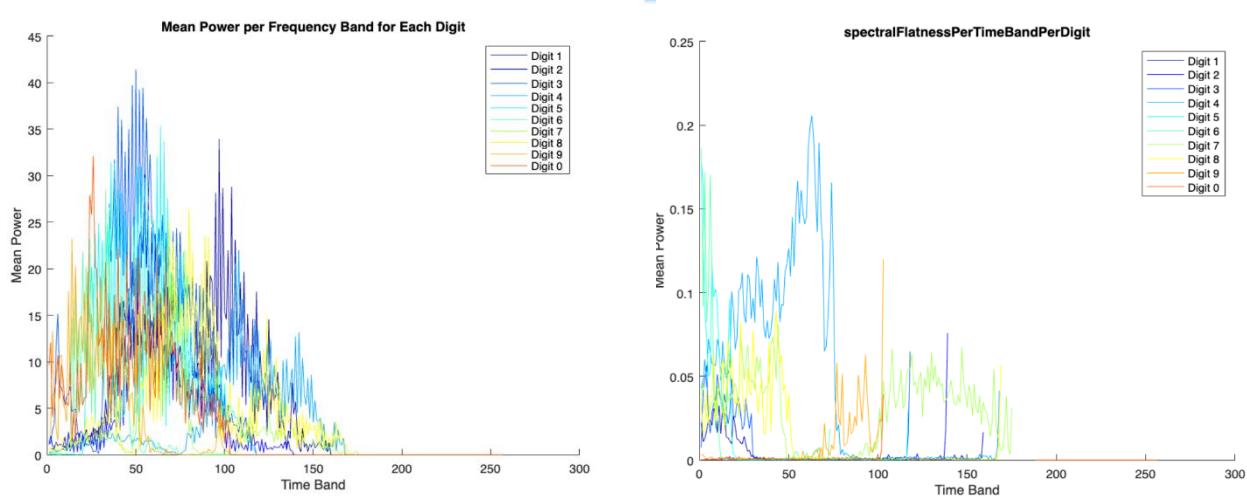


Figura 28 - exemplos gráficos 2d para algumas características (mean power per FB e spectral flatness per TB)

3.11– Escolha das três melhores características tempo-frequência

Com os gráficos 2D e 3D feitos, tínhamos dois métodos de reunir as características com um valor absoluto dos áudios de cada dígito:

-Registávamos a janela temporal do gráfico 2d em que se visualizasse uma maior discrepância dos valores da característica, por dígito, e fazímos nessa janela de tempo a média, mediana ou standard deviation de cada característica para todos os áudios, pegando depois nesses valores absolutos e fazendo o boxplot com estes.

-Ou retirávamos uma característica absoluta de cada áudio para cada característica (por exemplo std, mediana ou média) e fazímos com esses valores os boxplots para cada dígito, para cada característica.

Escolhemos a segunda opção porque, após fazer alguns testes, verificamos que era a que obtinha melhores resultados. Porém o primeiro método poderá ser uma boa opção para aumentar a percentagem de acerto de dígitos, na meta 4.

Os boxplots obtidos para as características, usando este método são:

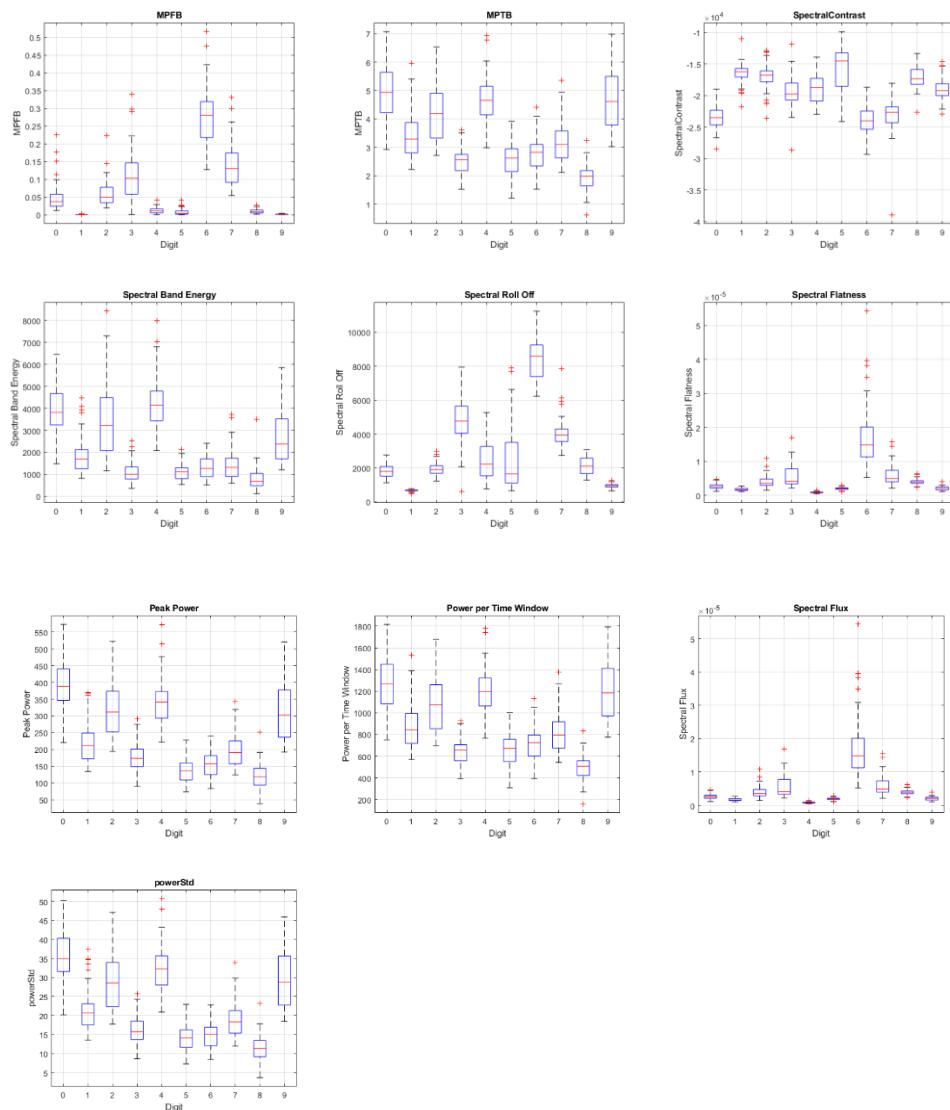


Figura 29 – Boxplots das características da meta III

Após a analise dos intervalos e agrupamento dos dígitos em cada boxplot, identificamos que as melhores características são a **Mean Power per Time Band** (MPTB), **Power per Time Window** e **Spectral Roll Off**. Estas são as características em que os intervalo de valores para cada dígito difere mais, logo, será mais fácil distingui-los.

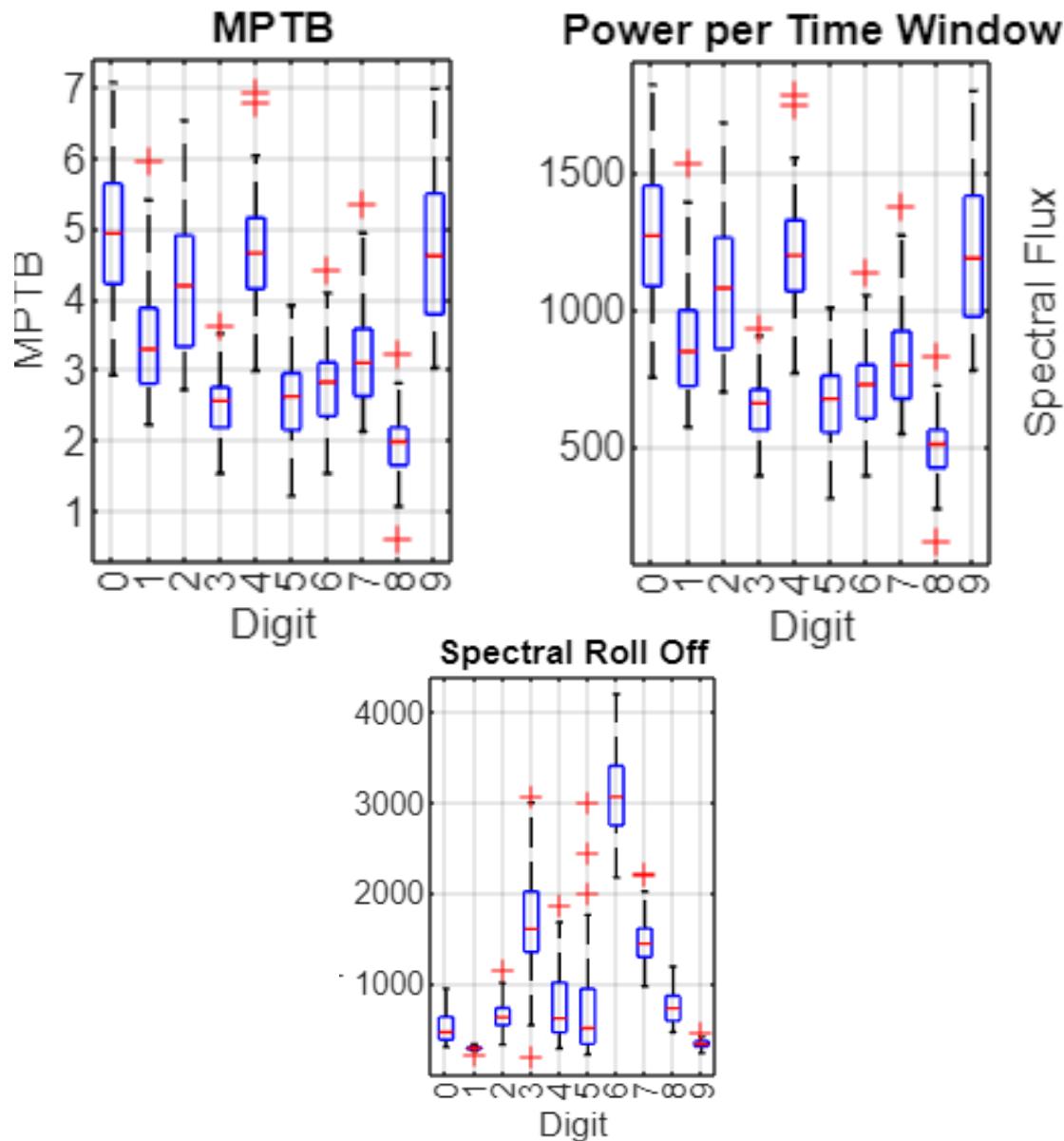


Figura 30 – Boxplots das 3 melhores características, por dígito

Valores das características, por dígito

No que toca à **mean power per time band**, é possível distinguir o dígito 8 por ter valores diferentes dos demais. É também possível distinguir os conjuntos de dígitos {1,7} dos {0,2,4,9} e {3,5,6,7}. Os valores das medianas para cada dígito são: 0 - 4.9; 1 - 3.3; 2 - 4.2; 3 - 2.6; 4 - 4.7; 5 - 2.7; 6 - 2.9; 7 - 3.1; 8 - 1.9; 9 - 4.7;

No que toca ao **Spectral Roll-Off**, é possível distinguir os dígitos {0,6,7} dos {1,2,5,8} e {3,4,9} por terem valores distintos. Os valores das medianas para cada dígito são: 0 - 498; 1 - 300; 2 - 1200; 3 - 1700; 4 - 700; 5 - 500; 6 - 3100; 7 - 1400; 8 - 750; 9 - 350;

No que toca à **Power per time window**, é possível distinguir o dígito 6 por ter valores diferentes dos demais. É também possível distinguir os conjuntos de dígitos {3,7} dos {1,9} e {0,2,4,5,8}. Os valores das medianas para cada dígito são: 0 - 1250; 1 - 900; 2 - 1100; 3 - 650; 4 - 1200; 5 - 700; 6 - 750; 7 - 800; 8 - 460; 9 - 1190;

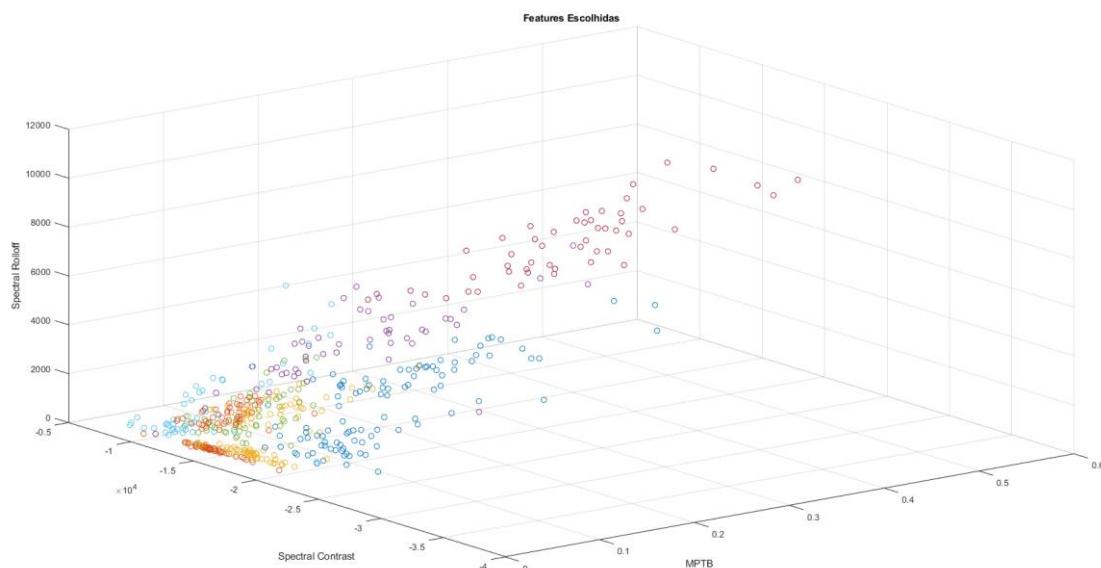


Figura 31 – Scatterplot3D das 3 melhores características da meta III (cada dígito tem uma cor diferente)

Meta IV

4.12 - Construção das regras de decisão para a distinção dos dígitos, usando as 3 melhores características de cada meta

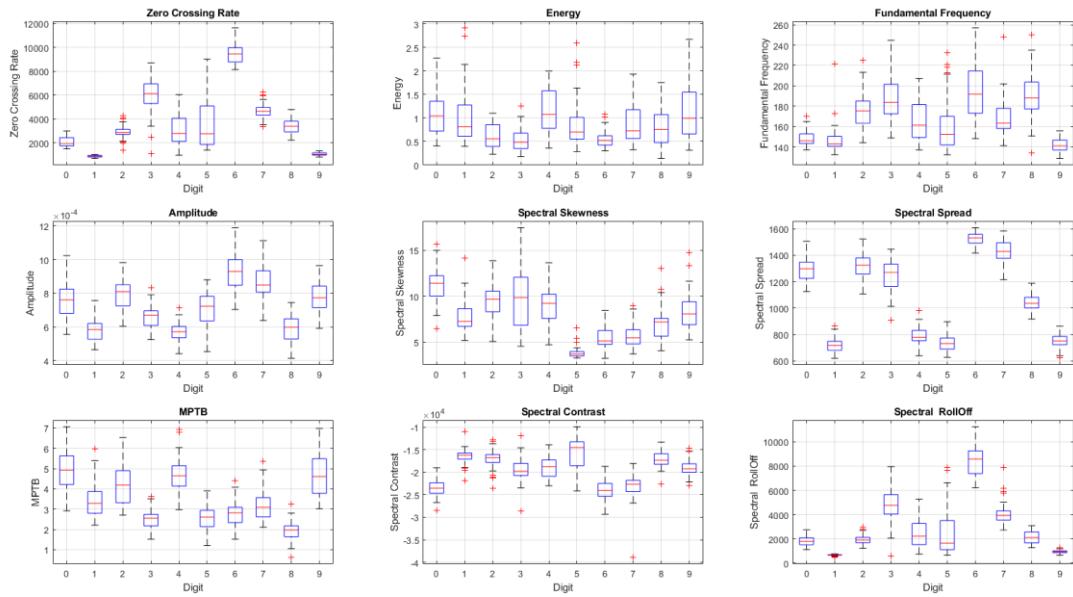


Figura 32 – Boxplots das 9 melhores características (1^a linha -> Meta I; 2^a linha -> Meta II; 3^a linha -> Meta III)

Primeiro identificámos quais os intervalos em que os dígitos estão mais próximos em cada característica (Esses intervalos já foram identificados nos pontos 1.4 para a primeira meta, 2.8 para a segunda e 3.11 para a terceira)

Definimos então um intervalo para cada conjunto de valores, este é constituído pelo valor mais baixo da característica presente no conjunto até ao valor mais alto (identificados na tabela 1):

Digitos por intervalo			
Meta 1			
Zero Crossing Rate			
{3}	{6}	{1,9}	{0,2,4,5,7,8}
[8302 – 6947]	[8781 – 9969]	[683 – 1320]	[2990 – 5083]
Energy			
{2,3,6}	{0,4,9}	{1,5,7,8}	
[0.345 - 0.855]	[0.656 - 1.57]	[0.473 - 1.27]	
Fundamental Frequency			
{0,1,9}	{4,5,7}	{2,3,6,8}	
[137 - 152.7]	[141.9 - 181.6]	[163.2 – 214]	
Meta 2			
Spectral Mean			
{1,4,8}	{0,2,3,5,9}	{6,7}	
[5.27 - 7.4] x10^-4	[6,35 - 8,42] x10^-4	[8,06 - 9,99] x10^-4	
Spectral Skewness			
{5}	{0}	{2,3,4,9}	{1,8,9}
[3,51 - 3,99]	[10,03 - 12,23]	[5,86 - 12,08]	[5,67 - 9,41]
[4,76 - 6,36]			
Spectral Spread			
{0,2,3}	{6,7}	{8}	{1,4,5,9}
[1164 – 1380]	[1376 – 1559]	[1000 – 1081]	[680 – 831]
Meta 3			
Mean Power Per Time Band			
{8}	{0,2,4,9}	{3,5,6,7}	{1,7}
[1,64 - 2,17]	[3,32 - 5,67]	[2,13 - 3,57]	[2,62 - 3,86]
Spectral Contrast			
{0,6,7}	{1,2,5,8}	{3,4,9}	
[-25352 , -21782]	[-18541 , -13217]	[-20888 , -17266]	
Spectral Roll-Off			
{ 3,7}	{6}	{1,9}	{0,2,4,5,8}
[3557 – 7950]	[7387 – 9260]	[500 – 1010]	[1110-3508]

Tabela 1 – Intervalos de valores das 9 melhores características

Calculámos os valores para cada uma das 9 características, para cada áudio, e usámos a função **ClosesInterval** para decidir qual o intervalo indicado mais próximo. Esta funciona ao verificar quais dos intervalos o valor pertence e mede a distância do valor ao extremo inferior destes. O que apresentar uma menor distância é escolhido. Se o valor for um *outlier*, ou seja, não pertence a nenhum intervalo definido, é procurado o que tenha limite inferior mais próximo deste ponto, passando esse a ser o intervalo de valores escolhido. Dependendo do intervalo escolhido é dado uma pontuação aos dígitos associados a esse intervalo. Quantos mais dígitos associados ao intervalo menor a pontuação escolhida, visto que isto significa que é pior a diferenciar dígitos. Chegado ao final é escolhido o dígito com pontuação máxima. Caso haja mais do que um dígito com a mesma pontuação máxima, é escolhido aleatoriamente entre esses

4.13 – Comparação com os dígitos reais e cálculo da percentagem de acerto

- Para o dígito 0 a percentagem de acerto foi 88%, sendo este apenas confundido com o dígito 7, 12% das vezes;
- Para o dígito 1 a percentagem de acerto foi 50%, isto deve-se a muitos destes dígitos serem confundidos com o 9;
- Para o dígito 2 a percentagem de acerto foi 48%, visto que este é confundido com diversos dígitos diferentes;
- Para o dígito 3 a percentagem de acerto foi 74%, sendo que 26% dos áudios são confundidos com o dígito 8 e outros números aleatórios;
- Para o dígito 4 a percentagem de acerto foi 52%, visto que este é confundido com diversos dígitos diferentes;
- Para o dígito 5 a percentagem de acerto foi 40%, visto que este é confundido com diversos dígitos diferentes;
- Para o dígito 6 a percentagem de acerto foi 96%, sendo este é adivinhado corretamente quase todas as vezes;
- Para o dígito 7 a percentagem de acerto foi 82%, sendo que apenas 18% dos áudios são confundidos com outros dígitos;
- Para o dígito 8 a percentagem de acerto foi 98%, sendo este corretamente adivinhado praticamente todas as vezes;
- Para o dígito 9 a percentagem de acerto foi 92%, sendo este corretamente adivinhado praticamente todas as vezes

É de notar que estas percentagens, bem como a taxa de acerto total, podem sofrer flutuações, isto deve-se maioritariamente aos casos em que mais do que um dígito apresenta a maior pontuação, sendo feita assim uma escolha aleatória entre os dígitos com a pontuação mais alta.

A percentagem de acerto total para os 500 áudios é **72.2%** dos dígitos adivinhados corretamente.

O principal problema ocorreu na identificação correta dos dígitos 1,2,3,4 e 5, visto que, nestes casos, outro dígito muitas vezes sobrepõe-se na triagem e seleção, devido à semelhança de valores nos conjuntos de algumas características e dígitos.

4.14 – Plot3D com as 3 melhores características

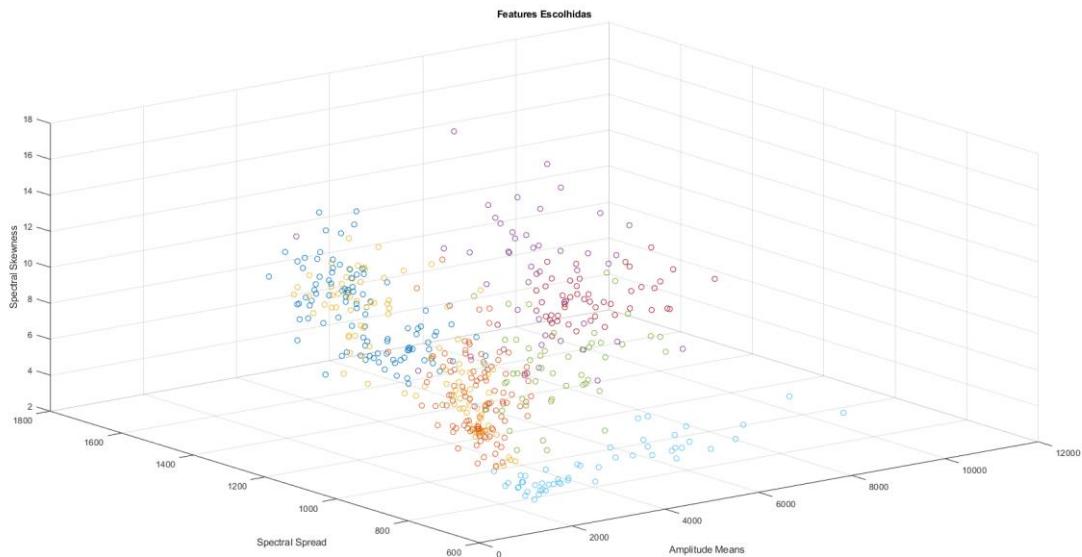


Figura 33 – ScatterPlot3D das 3 melhores características

Neste gráfico é possível identificar “nuvens” de cores diferentes, representando cada dígito. Isto mostra a eficácia destas características para identificar os dígitos diferentes, a partir dos valores de cada uma. As “nuvens” que estão sobrepostas, ou mais próximas, indicam uma maior dificuldade na distinção dos dígitos (o que se traduz nos dígitos terem menor taxa de acerto). Pelo contrário, as “nuvens” mais isoladas representam os dígitos mais facilmente identificáveis a partir destas características, devido aos seus intervalos de valores diferentes dos restantes.