UNIVERSIDADE D
COIMBRA

Faculdade de Ciências e Tecnologias

Análise e Transformação de Dados - Licenciatura em Engenharia Informática

Identificação de Dígitos Através de Características Extraídas de Sinais de Áudio

Trabalho Realizado por:
Nuno Batista e Francisco Lapa Silva

Março, 2024

Dados usados: 16

META 1

1/2 - Importação dos sinais de áudio e representações visuais

De modo a importar os sinais de áudio com a função `audioread`, foi criada a função `getFeatures(exampleID, plt)`, que recebe como argumentos `exampleID`, que indica o índice das amostras a analisar e plt, que decide se os dígitos analisados devem ou não ser *plotted* (são *plotted* caso `plt == 1`). `getFeatures` é chamada, pela primeira vez, com o argumento `plt = 1`, ou seja, com o intuito de criar um *plot* de cada um dos sinais.

Dentro dessa função, um primeiro *for loop* itera pelos dígitos de 0 a 9, a cada iteração, a função `audioread` guarda os valores das amplitudes de cada um dos dígitos na variável `y` e o número de elementos de `y` em `rows`, deste modo, é possível concluir qual dos 10 sinais de áudio tem o maior número de amostras e guardar esse número na variável `maxRows` que, posteriormente, será usada para homogeneizar a duração dos sinais.

```
Ficheiro: getFeatures.mlx
% Find the duration of the longest audio
maxRows = 0;
for i = 0:9
      [y, ~] = audioread(sprintf("samples/%d_16_%d.wav", i, exampleID));
      [rows, ~] = size(y);

      if(maxRows < rows)
            maxRows = rows;
      end
end
```

De seguida, inicia outro *for loop* que também itera por todos os dígitos com o intuito de fazer os *plots*, no entanto, para efeitos de melhor análise posterior, este *plot* exclui o silêncio inicial de cada um dos exemplos e adiciona silêncio ao final para todos os sinais terem a mesma duração. Este processo é feito através da análise da energia em janelas de tempo. A energia de cada janela é guardada no array frameEnergy, que é preenchido pela função `getFrameEnergy`, de acordo com a fórmula da energia.

$$E = \sum_{n=-\infty}^{\infty} |y[n]|^2$$

```
Ficheiro: getFrameEnergy.mlx
function [frameEnergy] = getFrameEnergy(y, frameSamples, numFrames)
% Calculate frame energy, the sum of squares of the samples in the frame
```

```
(not the integral because we are in the discrete domain)
      frameEnergy = zeros(numFrames, 1);
      % Iterate through every frame
      for j = 1:numFrames
            % The frame range is from the (j-1)th*frameSamples frame to
the (j+1)th*frameSamples frame
            frame = y((j - 1)*frameSamples + 1:j*frameSamples);
            % Get the frame's energy
            frameEnergy(j) = sum(frame .^ 2);
      end
end
```

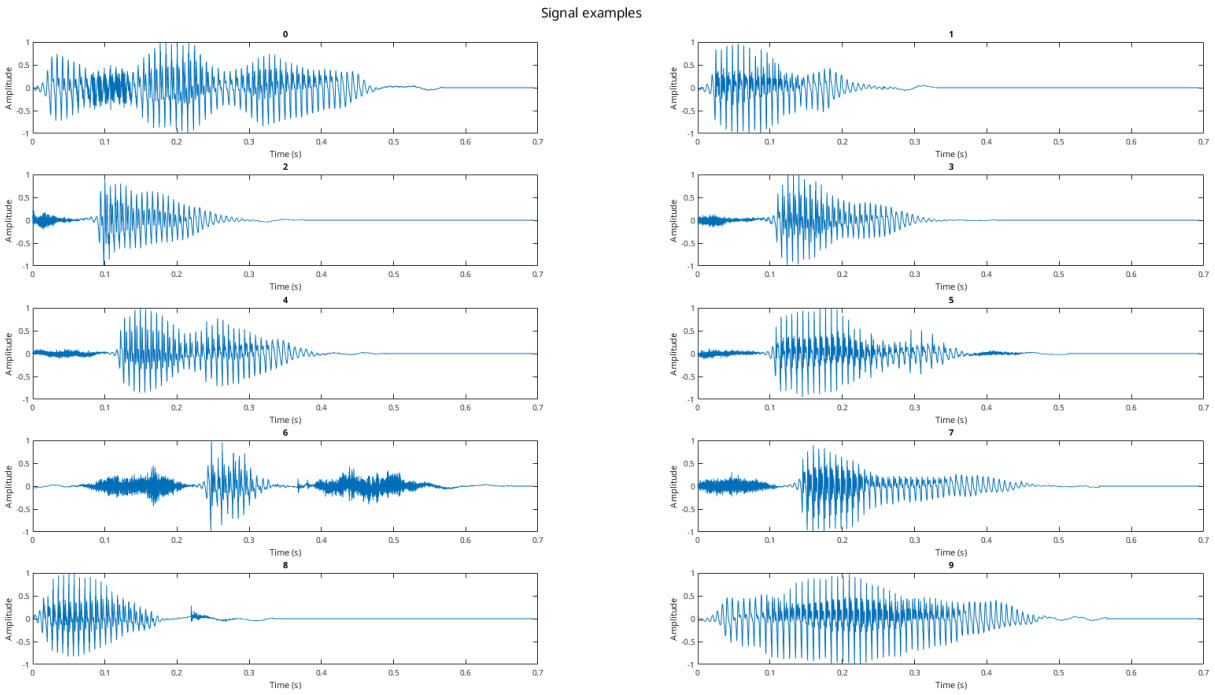
Basta, então, definir um *threshold* de energia e cortar todos as amostras antes da primeira que cumpre a restrição do *threshold* definido, bem como concatenar no final do sinal um vetor coluna cheio de elementos de valor 0 cujo número de elementos é igual à diferença entre o número de amostras do sinal de maior duração e o número de amostras do sinal atual (após o corte).

```
Ficheiro: getFetures.mlx
% frameEnergy is an array with the energy value of every frame
frameEnergy = getFrameEnergy(y, frameSamples, numFrames);

energyThreshold = 0.5;

% Find first index of the first frame with energy above threshold
startFrame = find(frameEnergy > energyThreshold, 1);
% Get the TIME index of the first frame with energy above threshold
startSample = (startFrame - 1) * frameSamples + 1;

% Trim y in order to remove low energy values
y = y(startSample:end);
% Add silence to the end of y
[curRows, ~] = size(y);
concatY = zeros(maxRows - curRows, 1);
y = [y; concatY + 0.5];
```

Finalmente, resta fazer o próprio *plot* do dígito na iteração atual. Obtendo assim o resultado final.

```
if plt == 1
      % Ts is the sampling period
      Ts = 1 / Fs;
      % t is the time vector
      t = (0:maxRows-1);
      % t is the time vector in seconds (starts at the first frame that
exceeds the energy threshold)
```

```
        t = (t .* Ts);
        subplot(5, 2, i + 1);
        plot(t, y');
        xlabel('Time (s)');
        ylabel('Amplitude');
        label = sprintf("%d", i);
        title(label);
end
```



3 - Análise de características temporais

A função `getFeatures` devolve a matriz 10x5 `resultFeatures`, que contém os resultados das análises de 5 características diferentes de cada um dos 10 dígitos (`resultFeatures(i + 1, x)` é o valor da característica x do dígito i). As características escolhidas foram:

1 - Energia total

2 - Desvio Padrão

3 - Amplitude máxima

4 - Taxa de cruzamento por zero

5 - Duração em segundos

Para calcular a energia total, basta utilizar a fórmula anteriormente referida, mas desta vez, para todos os elementos de y.

```
resultFeatures(i + 1, 1) = sum(abs(y) .^ 2)
```

É de referir que no início da análise de cada dígito, a amplitude é normalizada de modo a todos os valores entrarem no raio [-1, 1] tornando assim o valor do desvio padrão muito reduzido, portanto, o cálculo desta característica é realizado pela função `std` antes da normalização.

```
resultFeatures(i + 1, 2) = std(y);
```

O mesmo acontece com a amplitude máxima, que é dada pelo valor máximo de `y`.

```
resultFeatures(i + 1, 3) = max(y);
```

A taxa de cruzamento por zero é calculada pela função `zerocrossrate`, é importante ter em conta que é por causa da análise desta característica que foi tomada a decisão de fazer uma normalização no raio [-1 1] e não [0 1].

```
resultFeatures(i + 1, 4) = zerocrossrate(y);
```

Para calcular a duração, é necessário remover o silêncio no final do sinal, para isso, procura-se a última janela com um valor de energia superior ao do *threshold* definido e divide-se a diferença entre a posição da última e da primeira amostra que cumprem a restrição do *threshold* pela taxa de amostragem `Fs`.

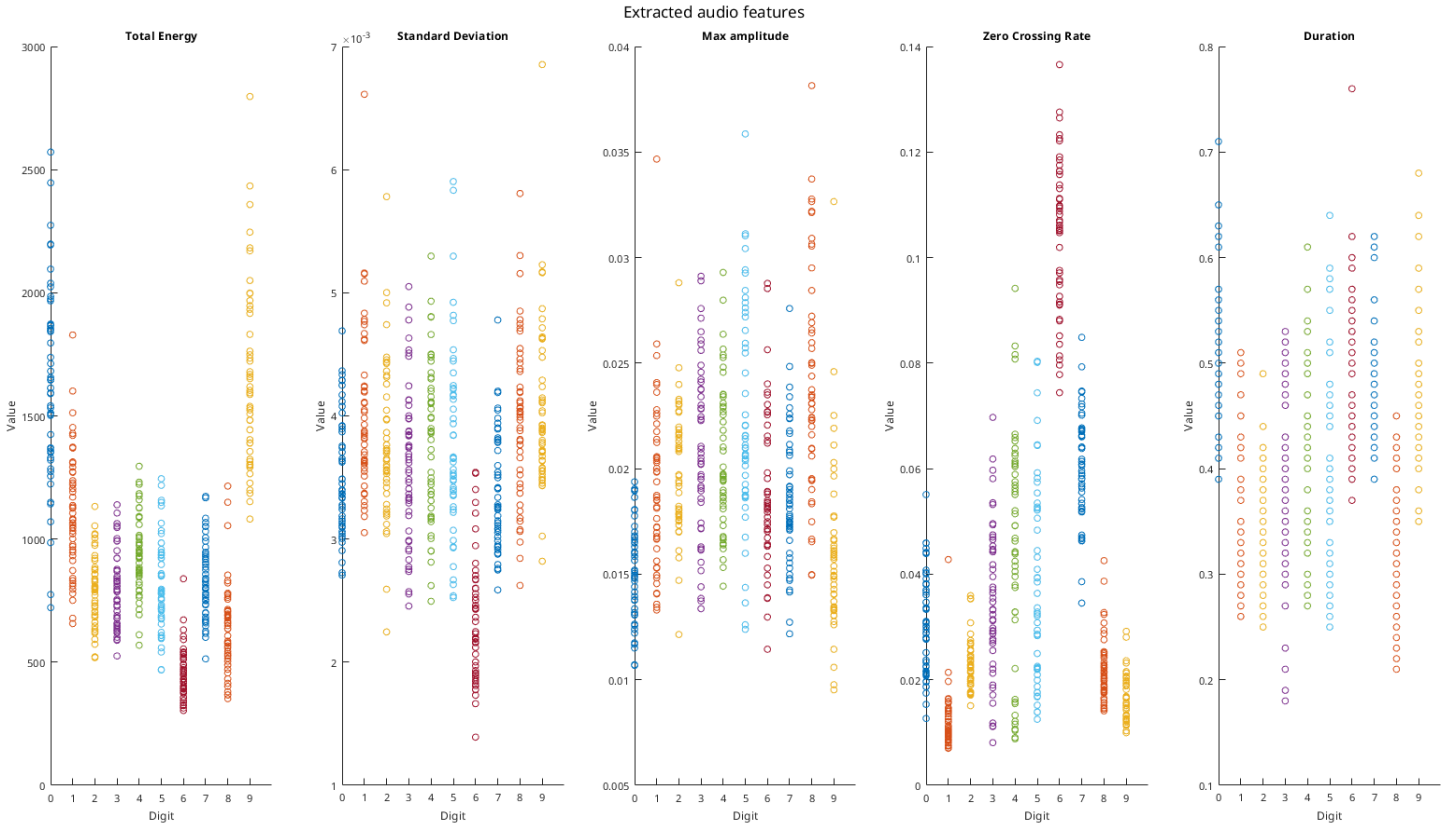
```
Ficheiro: getFeatures.mlx
energyThreshold = 0.1;
% Find first index of the first frame with energy above threshold
startFrame = find(frameEnergy > energyThreshold, 1);
% Get the TIME index of the first frame with energy above threshold
startSample = (startFrame - 1) * frameSamples + 1;

% Find the index of the last frame with energy above threshold
endFrame = find(frameEnergy > energyThreshold, 1, 'last');
% Get the TIME index of the last frame with energy above threshold
endSample = endFrame * frameSamples;

% Store the audio duration without silences in seconds
resultFeatures(i + 1, 5) = (endSample - startSample) / Fs;
```

Evidentemente, `resultFeatures` tem apenas as características de um conjunto de dígitos, no entanto, há 50 conjuntos, portanto, na função main, um *for loop* chama esta função com os IDs dos

conjuntos restantes, ou seja, de 1 a 49 e guarda essas características na matriz 3D `audioFeatures`, onde `audioFeatures(x, y, z)` é o valor da característica `z` do dígito `y-1` do conjunto `x-1`.

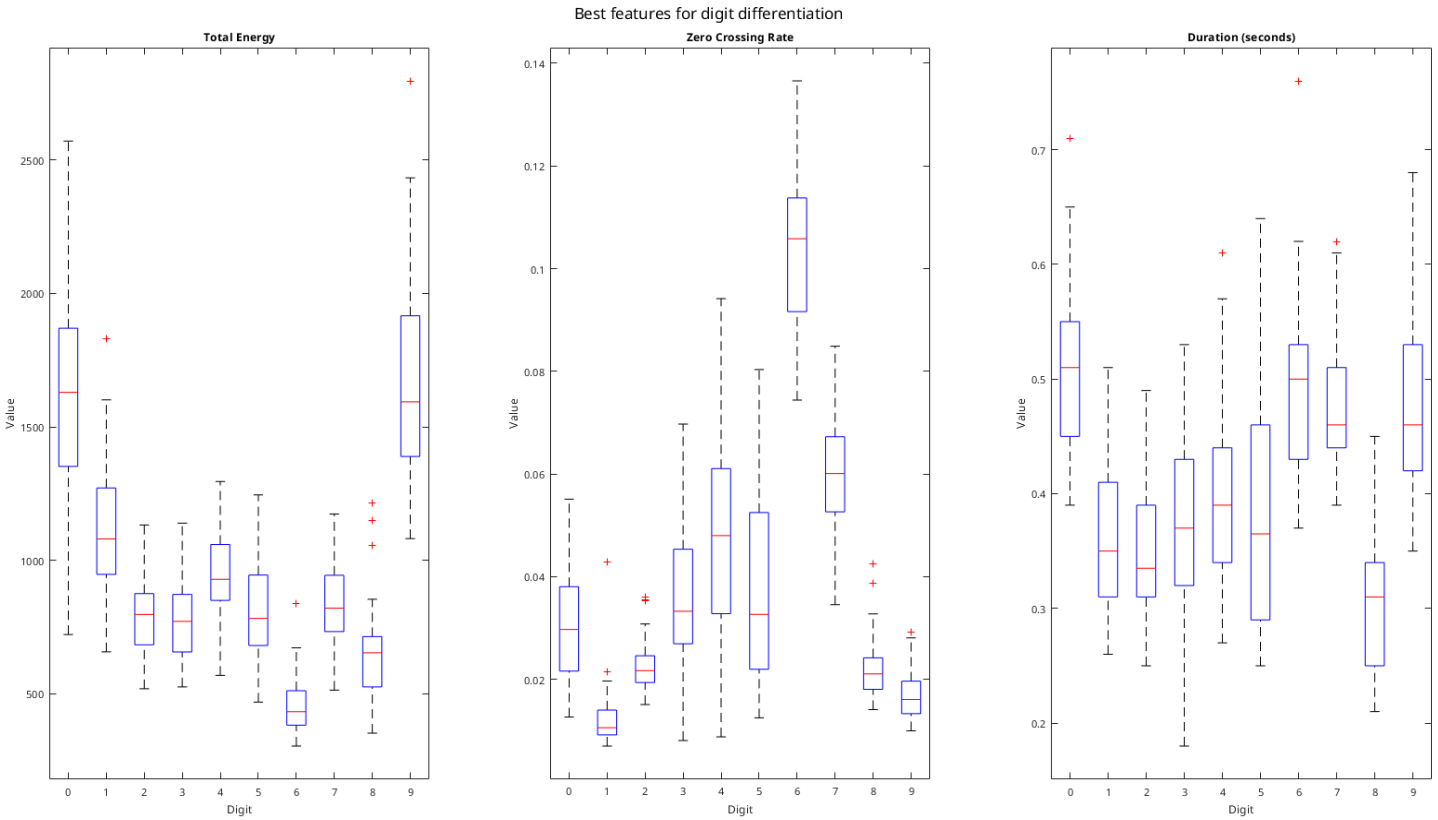
```
% i + 1 because MATLAB uses 1-based indexing but the audio file names
use 0-based indexing
for i = 1:49
      audioFeatures(i+1, :, :) = getFeatures(i, 0);
end
```

Por último, resta apresentar os resultados visualmente, obtendo o seguinte resultado com *scatter plots* 2D.



4 - Identificação das melhores características temporais para discriminação de dígitos

Tendo em conta os resultados obtidos no ponto anterior, a energia total, a taxa de cruzamento por zero e a duração foram consideradas as três melhores características para descriminação dos dígitos, para representar estas três características visualmente, usam-se *boxplots* 2D, obtendo assim o seguinte resultado.



Estas foram escolhidas pois, como os valores entre diferentes dígitos são, tendencialmente, mais diferentes, ao analisar estas mesmas características de uma dada amostra de um dado dígito, com base nos valores obtidos, é possível ter uma melhor ideia de qual dígito se trata.

Por exemplo, quando a o valor da energia total está por volta dos 2000 J, há uma probabilidade mais elevada de se tratar do dígito 0 ou 9, por outro lado, se a taxa de cruzamento por zero estiver por volta dos 0.1, é provável que estejamos perante o dígito 6, entre outros exemplos.

META 2

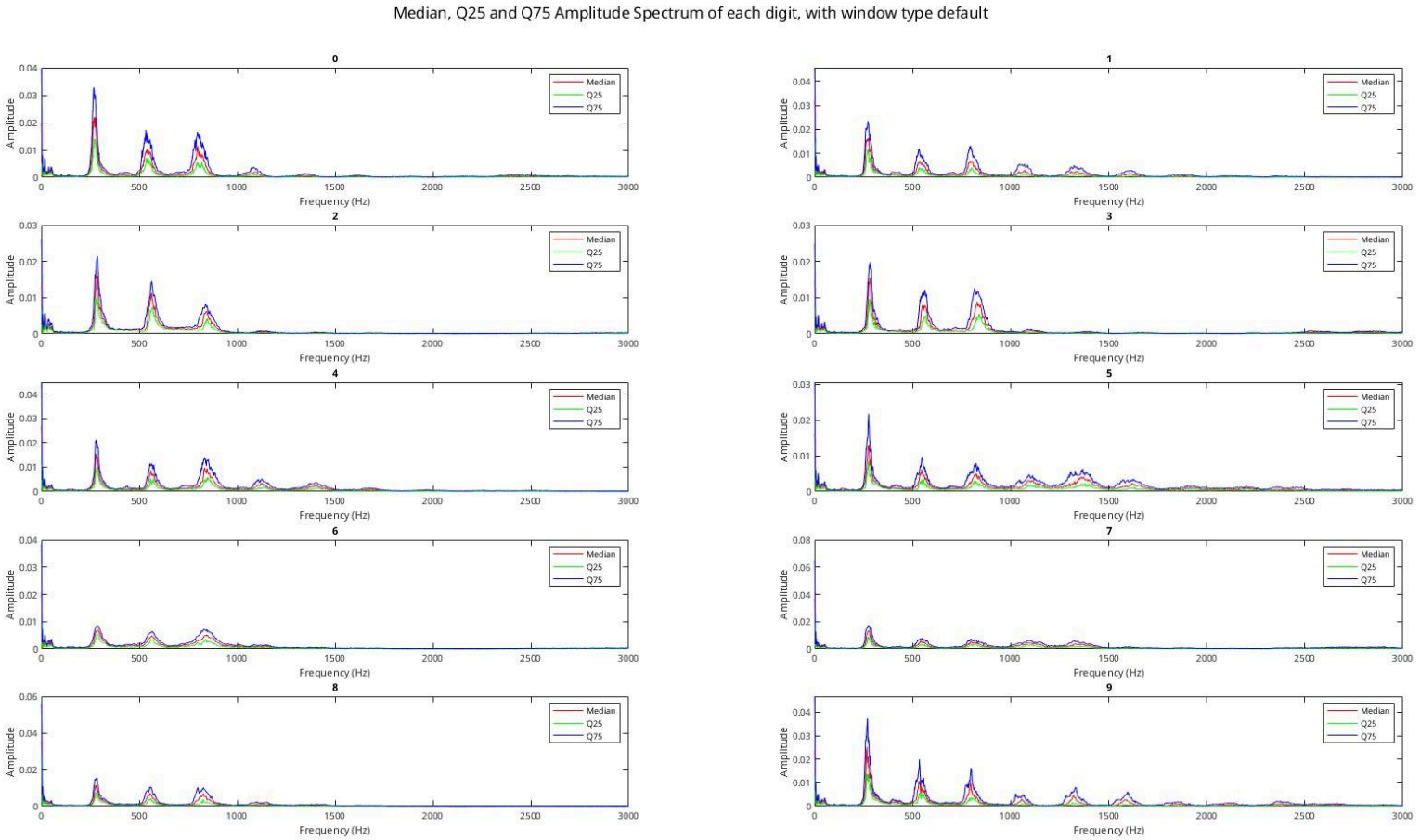
1 - Série de Fourier dos sinais de áudio

Nesta meta, serão analisadas algumas características espectrais dos sinais de áudio, para obter os espectros de amplitudes de cada dígito, vamos recorrer à mediana, ao primeiro e ao terceiro quartil dos valores absolutos dos coeficientes da série complexa de fourier dos 50 exemplos de cada dígito, normalizados pelo número de amostras. Ou seja, os c_m dados por:

$$c_m = \sum_{n=0}^{N-1} y[n] e^{-i2\pi kn/N}$$

A normalização é importante pois quantas mais amostras existirem, mais "contribuições" para uma dada frequência podem haver, ao normalizar pelo número de amostras, está, essencialmente, a ser feita uma média das contribuições de cada amostra.

Apresentando os resultados visualmente, obtém-se a seguinte figura (é de notar que não é apresentado o eixo das frequências completo para efeitos de melhor visualização)

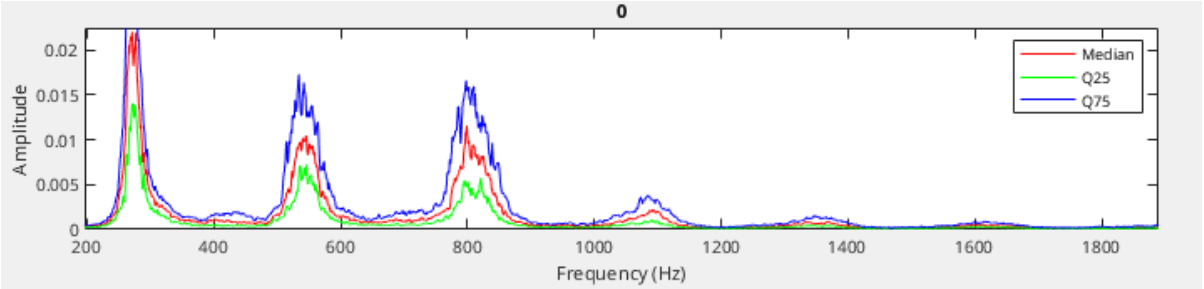


2 - Aplicação de diferentes tipos de janela

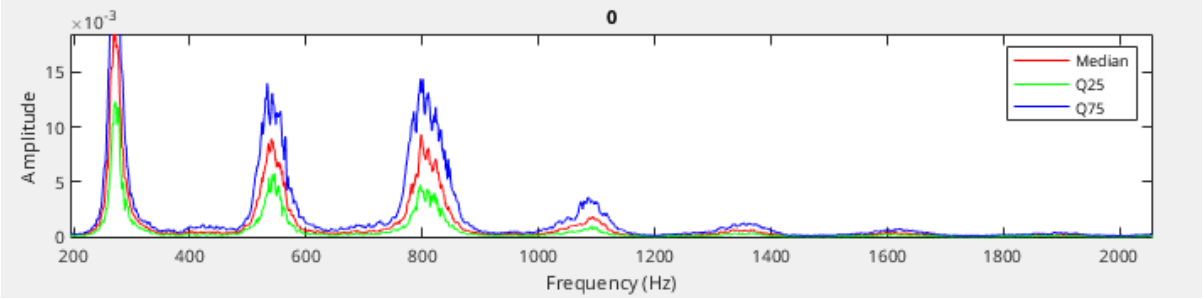
Em teoria, as transformadas de fourier devem ser calculadas no intervalo]-∞, +∞[, no entanto, os computadores, só processam um número finito de pontos, o que é equivalente a calcular a FT de um sinal multiplicado por uma função retangular, que se traduz, no domínio da frequência, para a convolução do sinal com uma função retangular, deste modo, os sinais acabam por ser cortados nas suas extremidades, acabando sempre por causar algum *spectral leakage* e altos lobos laterais.

A aplicação de funções de janelas tem como objetivo o controle destes efeitos. As três janelas escolhidas para comparar foram as de Hamming, Hann e Blackman, obtendo os seguintes resultados para o dígito 0:

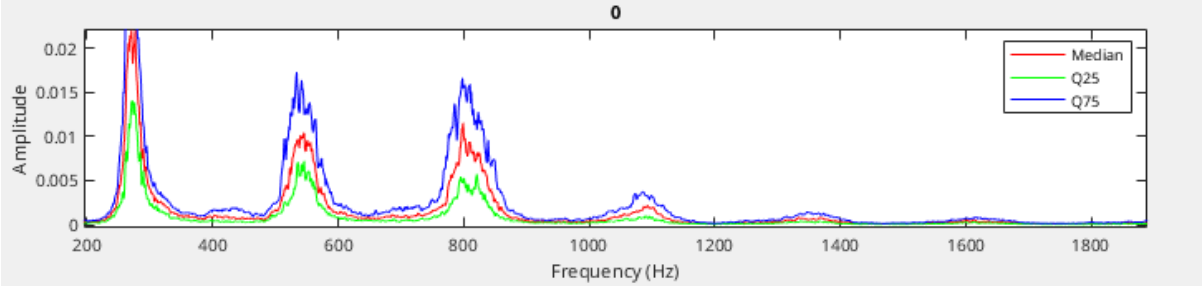
Sem janela



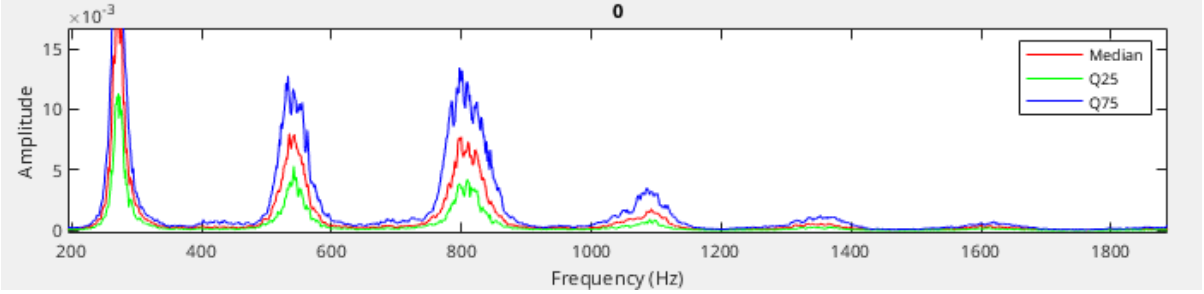
Hamming



Hann



Blackman



Apesar de, em teoria, as janelas diminuírem o *spectral leakage* e os lobos laterais. Nesta fase, esses efeitos ainda são muito pouco notáveis e as representações visuais não diferem significativamente.

3 - Análise de características espectrais

Bem como na meta anterior, nesta foram escolhidas algumas características espectrais a analisar, sendo essas:

1 - Média espectral

2 - Fluxo espectral

3 - Desvio padrão espectral

4 - Mediana dos picos espectrais

5 - Variância espectral

6 - Entropia de Wiener

A média, o desvio padrão e a variância espectral são calculadas usando as suas respectivas funções do MATLAB,

O fluxo espectral é dado pela soma dos valores absolutos das diferenças entre valores consecutivos, ou seja, sendo $F\{y\}_n$ o coeficiente $n$da série de fourier complexa:

$$\sum_{n=0}^{N-1} |\mathcal{F}\{y\}_n - \mathcal{F}\{y\}_{n-1}|$$

Podemos assim, usar as funções `diff`, `abs` e `sum`.

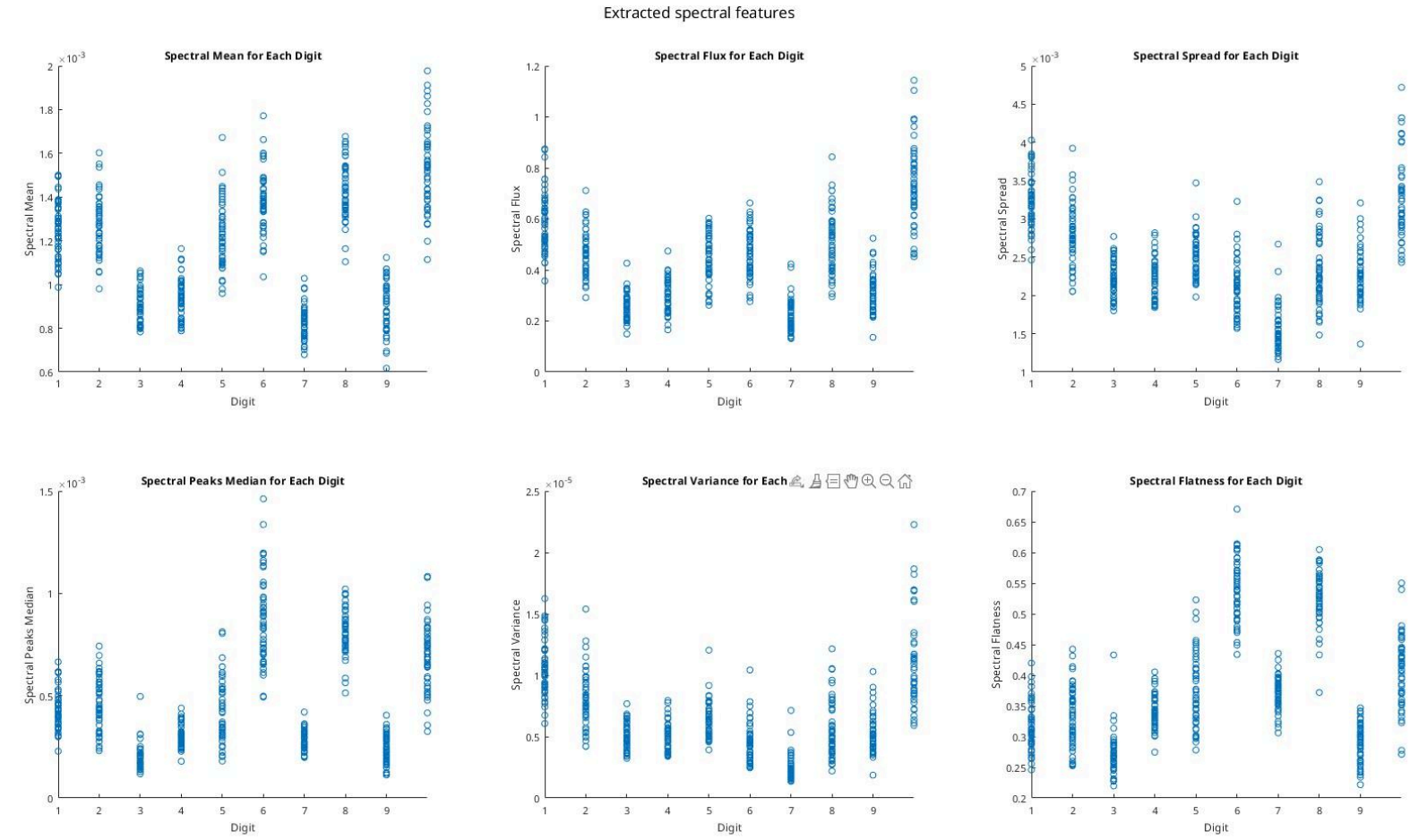
Para encontrar a mediana dos picos espectrais, utilizou-se a função `findpeaks` para receber um array com os picos locais de cada um dos exemplos de áudio, seguido da função `median`, para tornar os dados menos dimensionais.

A entropia de Wiener ou *spectral flatness*, por sua vez, descreve o quanto um som é tonal, em oposição a apenas barulho e é dada pela média geométrica sobre a média aritmética, ou seja:

$$\frac{\sqrt[N]{\prod_{n=0}^{N-1} \mathcal{F}\{y\}_n}}{\frac{1}{N}\sum_{n=0}^{N-1} \mathcal{F}\{y\}_n}$$

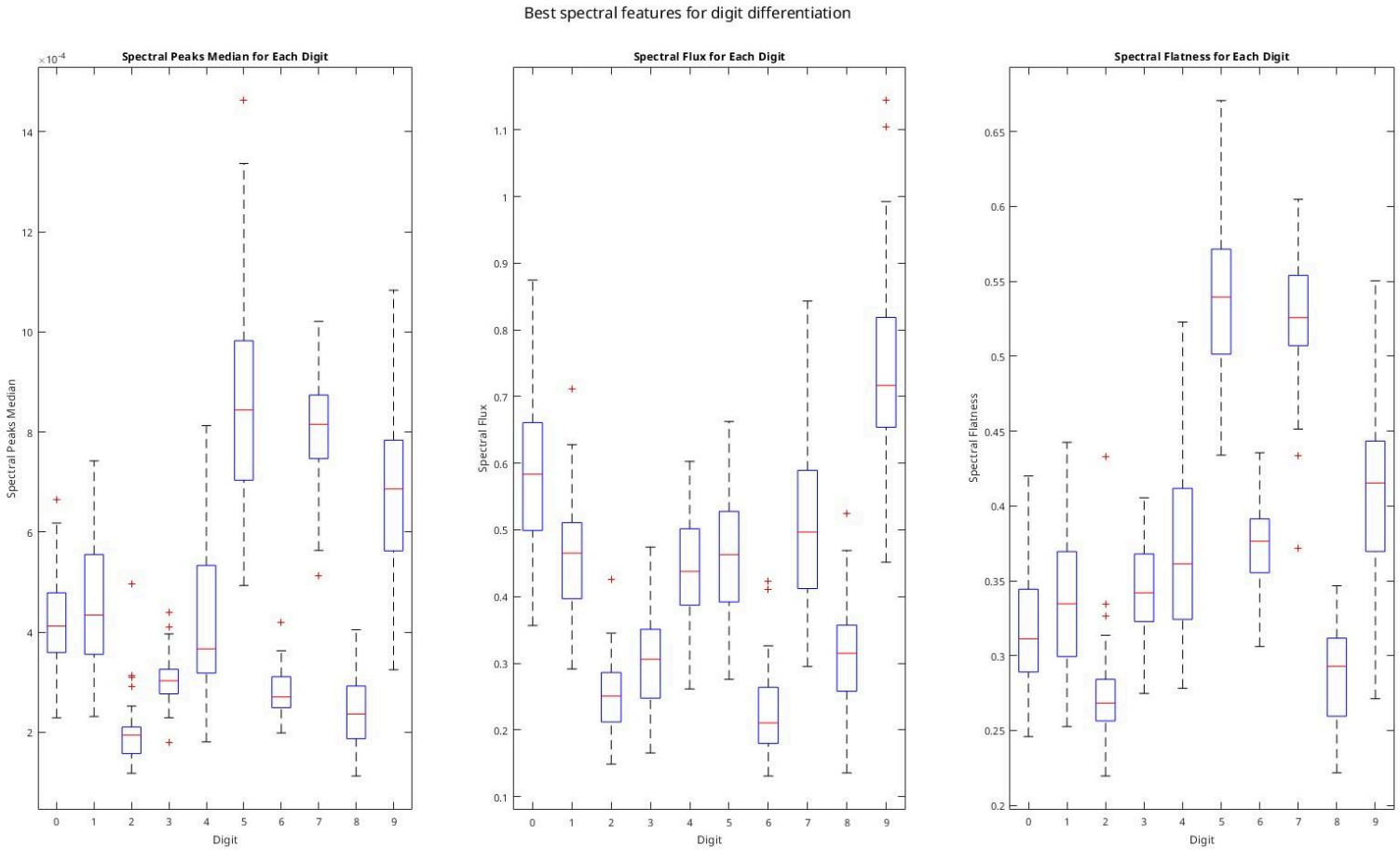
Ambas as médias podem ser calculadas pelas funções `geomean` e `mean`.

Representando visualmente todas estas características para cada dígito, obtemos o seguinte *scatterplot*:



4 - Identificação das melhores características espectrais para discriminação de dígitos

Tendo em conta os resultados obtidos no ponto anterior, a mediana dos picos espectrais, o fluxo espectral e a entropia de Wiener foram consideradas as três melhores características para descriminação dos dígitos, para representar estas três características visualmente, usam-se *boxplots* 2D, obtendo assim o seguinte resultado.



A probabilidade de o dígito ser "1" aumenta quando o valor da Mediana dos Picos Espectrais se aproxima de 4.1, o valor do Fluxo Espectral se aproxima de 0.6, e o valor da Planicidade Espectral se aproxima de 0.31.

É de esperar que a probabilidade de ser dígito "2" aumente quando o valor da Mediana dos Picos Espectrais se aproxima de 4.2, o valor do Fluxo Espectral se aproxima de 0.46, e o valor da Planicidade Espectral se aproxima de 0.34.

É provável que o dígito seja "3" quanto se o valor da Mediana dos Picos Espectrais rondar 2.0, o valor de Fluxo Espectral se aproximar de 0.25, e o valor de Planicidade Espectral se aproximar de 0.27.

Quando o valor da Mediana dos Picos Espectrais rondar os 3.5, o valor de Fluxo Espectral se aproximarem de 0.45, e o valor da Planicidade Espectral for próxima de 0.36, é provável que o dígito seja "4".

A probabilidade de o dígito ser "5", aumenta quando o valor da Mediana dos Picos Espectrais se aproxima de 8.1, o valor do Fluxo Espectral se aproxima de 0.46, e o valor da Planicidade Espectral se aproxima de 0.54.

É provável que o dígito seja "*6*" quanto se o valor da Mediana dos Picos Espectrais rondar 2.5, o valor de Fluxo Espectral se aproximar de 0.2, e o valor de Planicidade Espectral se aproximar de 0.37.

A probabilidade de o dígito ser "7" é maior quanto mais o valor da Mediana dos Picos Espectrais se aproxima de 8.0, e o valor de Fluxo Espectral está próximo de 0.5, e o valor da Planicidade Espectral se aproxima de 0.53.

É mais provável que o dígito seja "8" quanto mais próximo o valor da Mediana dos Picos Espectrais se aproxima de 2.1, o valor de Fluxo Espectral de 0.31, e o valor de Planicidade Espectral de 0.3.

A probabilidade do dígito ser "9" aumenta quanto mais o valor da Mediana dos Picos Espectrais se aproximar de 7.0, o valor de Fluxo Espectral se aproximar de 0.71, e o valor de Planicidade Espectral se aproximar de 0.43.