

Projeto ATD

David Silva - 2020217642

Eduardo Carneiro - 2020240332

1.1

Inicialmente, importamos o sinal dos acelerômetros que medem os 3 eixos (X,Y,Z) das experiências correspondentes à PL1, ou seja, da experiência 1 à experiência 8 (user1 a user4). Além disso, temos de importar as labels e as activity_labels (que possuem, essencialmente os tempos que as atividades se realizaram numa certa experiência) para uma matriz. A informação das experiências e dos tempos em que certa atividade decorreu numa determinada experiência será guardada recorrendo a Map Containers.

```
formatSpec = '%f %f %f';

%vai buscar a path da pasta
files = dir(pwd);

%%leitura de dados
%guardamos as experiencias num dicionario em que a chave é o numero da experiencia
n_experiencia={};
n_utilizador=[];

data_exp={};

contador=1;
contador_utilizador=1;

for i = 1:numel(files)
    %todos os ficheiros que iniciam por acc (tem os dados do acelerometro de cada
    %experiencia)
    if contains(files(i).name, 'acc')
        file=fopen(files(i).name,'r');
        data=fscanf(file,formatSpec,[3 Inf])';
        n_utilizador = [n_utilizador contador_utilizador];
        n_experiencia = [n_experiencia contador];
        data_exp=[data_exp data];
        if mod(contador,2)==0
            contador_utilizador=contador_utilizador+1;
        end
        contador=contador+1;
    end
end

experiencias=containers.Map(n_experiencia,data_exp);

% para aceder aos dados da experiencia:
% experiencias(1) --- 1 e o numero da experiencia

%Import Activity Labels
fileLabels = fopen('labels.txt','r');
fileActivityLabels = fopen('activity_labels.txt','r');
```

```

labels = fscanf(fileLabels,'%d %d %d %d',[5 Inf]);
activityLabels = textscan(fileActivityLabels,'%d%s');

%obter os intervalos de tempo em que certa atividade se realizou
activity_frame={};
for h=1:12
    frames_activity={};
    frames={};
    for i=1:8
        fr=get_activity_file(i,n_utilizador(i),h,labels);
        frames=[frames fr];
    end

    frames_activity=containers.Map([1:8],frames);
    activity_frame=[activity_frame;containers.Map(h,frames_activity)];
end

%para aceder às frames de cada atividade em cada experiencia
%activity_frame(1).values({1}) frames atividade 1 da experiencia 1

```

2

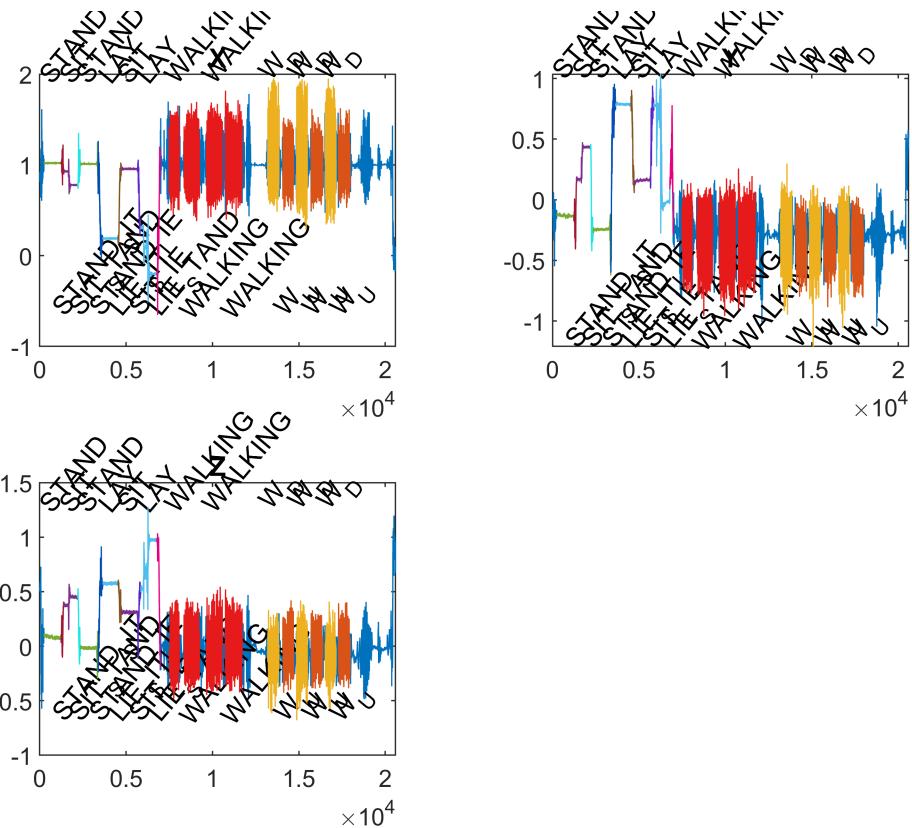
De forma a representar as medições dos acelerômetros são feitos 3 gráficos por experiência, em que cada gráfico representa cada eixo do acelerómetro.

```

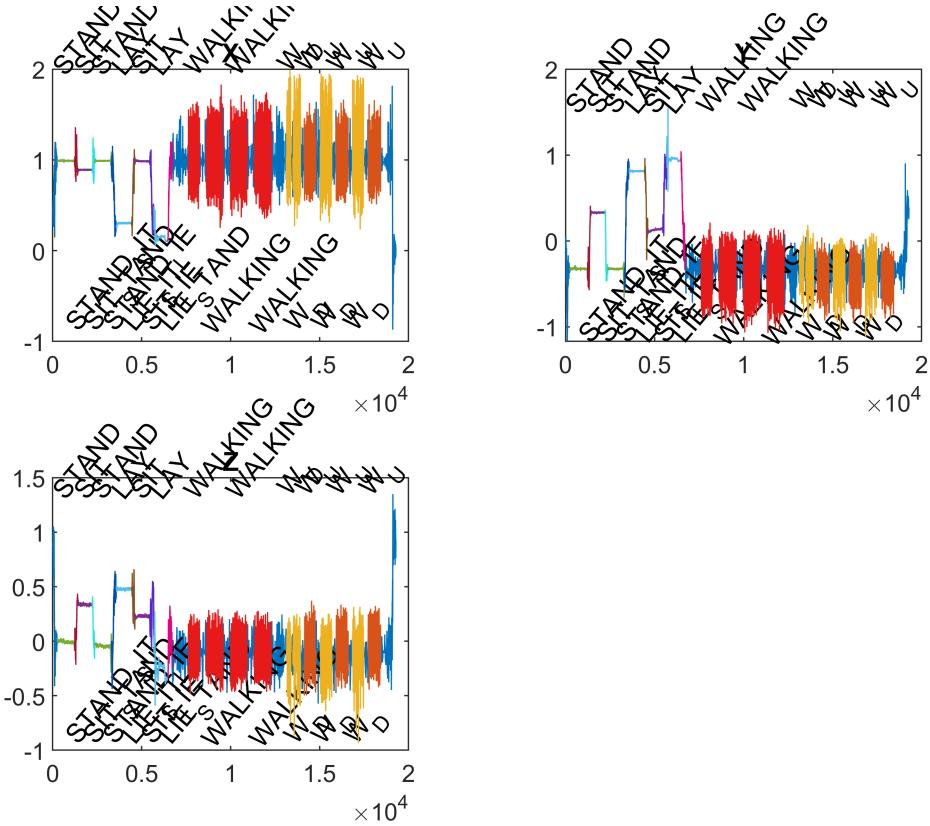
% print all graphics
for i=1:experiencias.Count
    fprintf("EXPERIENCIA %d\n",i)
    figure('Name',sprintf("EXPERIENCIA %d ",i))
    print_with_labels(experiencias(i),labels,activityLabels,i);
end

```

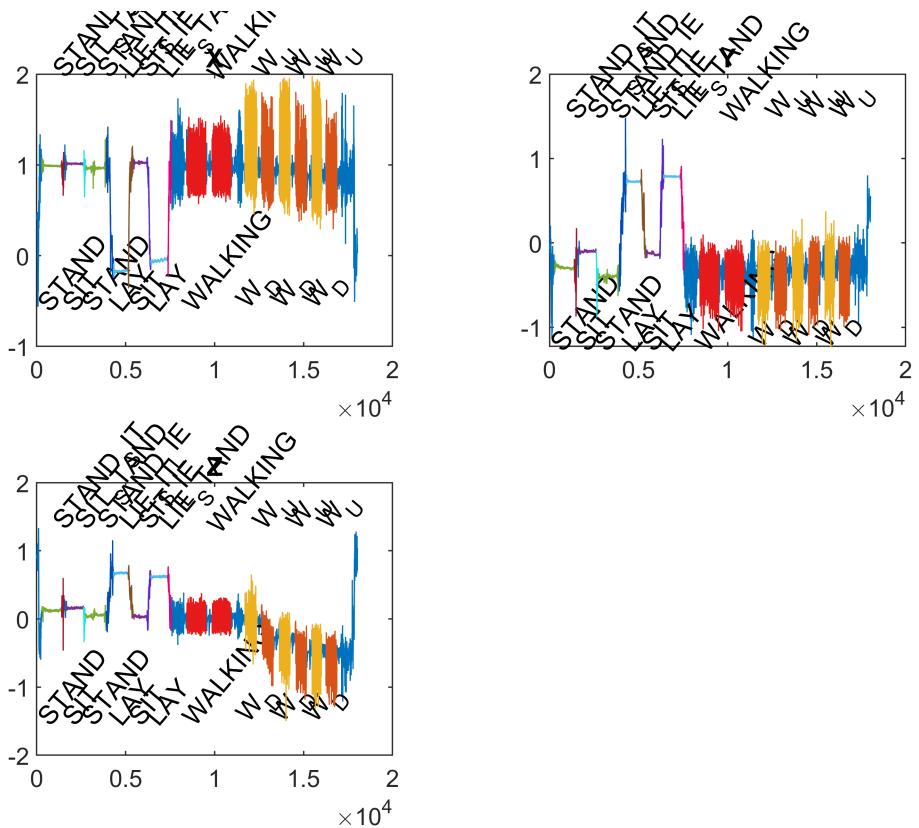
EXPERIENCIA 1



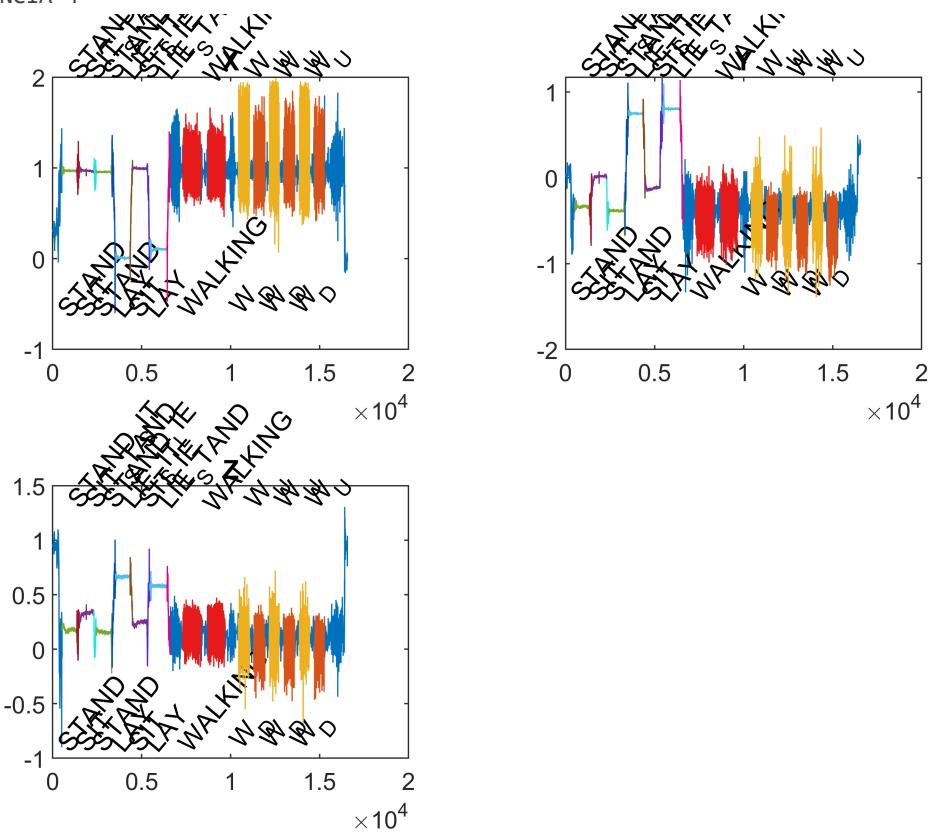
EXPERIENCIA 2



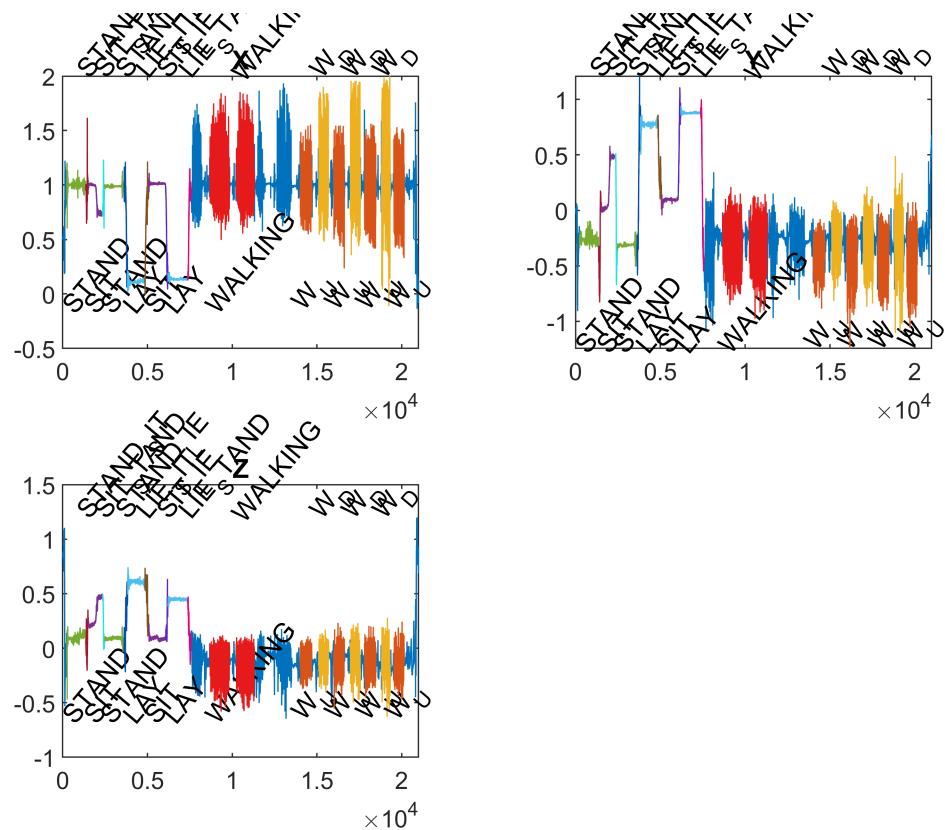
EXPERIENCIA 3



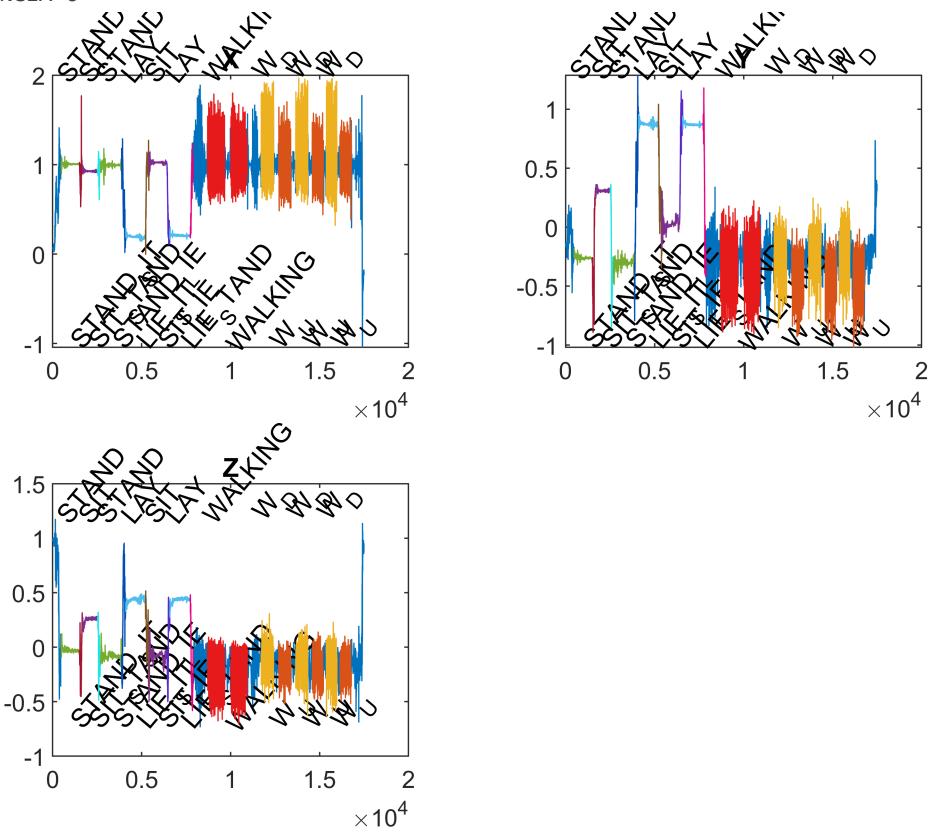
EXPERIENCIA 4



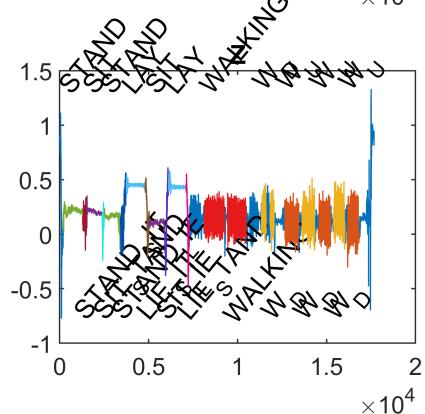
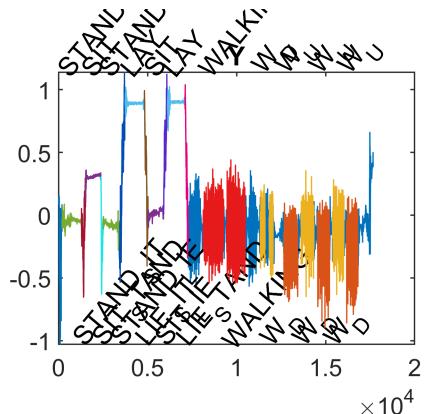
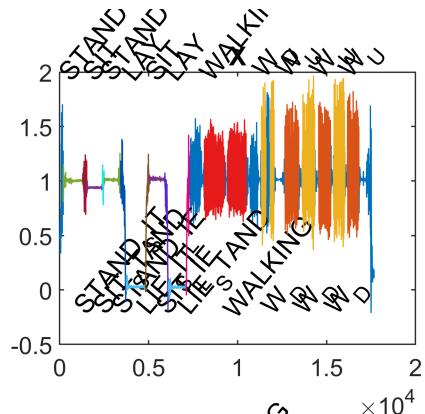
EXPERIENCIA 5



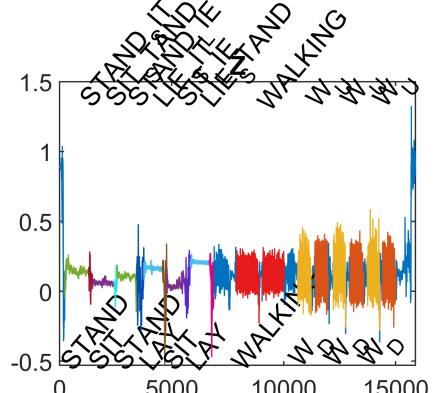
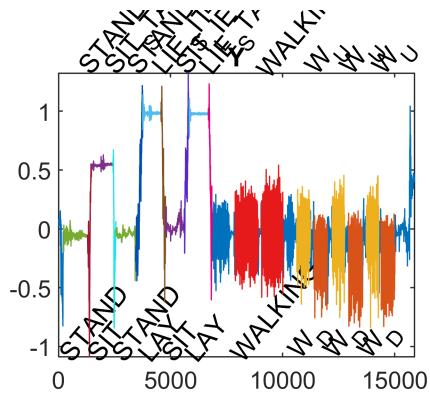
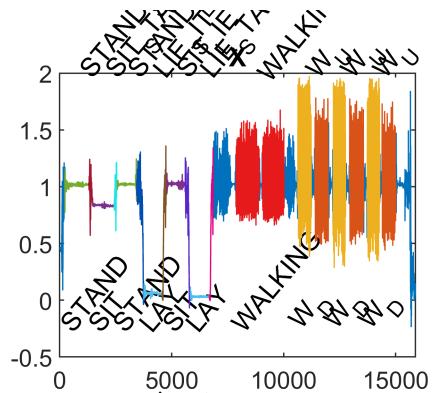
EXPERIENCIA 6



EXPERIENCIA 7



EXPERIENCIA 8



De seguida, calculamos a DFT de cada atividade em cada experiência e em cada eixo. Os valores do módulo da DFT correspondem à magnitude de certas frequências. Assim, é-nos possível obter as frequências relevantes de cada atividade em cada eixo, estabelecendo um threshold a partir do qual consideramos uma frequência como relevante (neste caso, considerámos relevantes qualquer frequência com magnitude superior a 80% do valor da maior magnitude). Posteriormente filtramos as frequências relevantes de cada atividade para que seja possível identificar cada atividade.

A caracterização das atividades é feita recorrendo a intervalos de 2 Hz. Seguidamente, são selecionados os intervalos que contêm mais elementos (isto é, os intervalos onde temos mais frequências, portanto as que melhor caracterizam a atividade) e é calculada a média e o desvio padrão para cada um desses intervalos. Estes valores são-nos úteis posteriormente, aquando da tentativa de identificação de cada atividade.

```

freq_filtered={};
for i=1:activity_frame.Count
    x=[];
    y=[];
    z=[];
    for j=1:experiencias.Count
        [auxx,auxy,auxz] = DFT_activity(experiencias(j),cell2mat(activity_frame(i).values({j})),0,0);
        x=[x auxx];
        y=[y auxy];
        z=[z auxz];
    end
    [xf,yf,zf,med_x,med_y,med_z] = get_freqs_filtered(x,y,z);

    freq=containers.Map({'xf' 'yf' 'zf' 'med_x' 'med_y' 'med_z'}, {xf yf zf med_x med_y med_z});

    freq_filtered=[freq_filtered;containers.Map(i,freq)];
end

%acceder as características relevantes de cada atividade
%freq_filtered(3).values({'xf','yf','zf','med_x','med_y','med_z'})

for i=1:activity_frame.Count
    fprintf("Atividade %d\n",i);
    freq_filtered(i).values({'xf','yf','zf','med_x','med_y','med_z'})
end

```

Atividade 1
ans = 1×6 cell

	1	2	3	4	5	6
1	[1.6618,1.8...	[1.6618,1.8...	[0.8486,1.7...	1.7612	1.9858	1.2989

Atividade 2
ans = 1×6 cell

	1	2	3	4	5	6
1	[1.435,1.68...	[1.2582,1.7...	[0.5554,1.2...	1.5596	1.4917	0.8814

Atividade 3
ans = 1×6 cell

	1	2	3	4	5	6
1	[1.5463,1.8...]	[1.3948,1.8993;2.2433,...]	[0.6161,1.2...]	1.773	3.1768	1.0139

Atividade 4

ans = 1x6 cell

	1	2	3	4	5	6
1	[0,1.2971;0,7....]	[0,0.9038]	[0.1128,0.9...]	4.2551	0.813	0.7746

Atividade 5

ans = 1x6 cell

	1	2	3	4	5	6
1	[2.3888,3.4103;4.3579,...]	[0.1259,1.1...]	[0.1581,0.4...]	3.9268	1.0405	0.3205

Atividade 6

ans = 1x6 cell

	1	2	3	4	5
1	[0,1.0266]	[0.0225,1.2725;2.2983,3.3178;4.42...]	[0.0321,0.9...]	0.9835	3.3237

Atividade 7

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.1567,1.2...]	[0.2347,0.3...]	[0.1833,0.7...]	0.9094	0.2823	0.4533

Atividade 8

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.3806,0.8...]	[0.1866,0.4...]	[0.3071,0.6...]	0.6163	0.3064	0.4547

Atividade 9

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.2319,0.3...]	[0.2418,0.3...]	[0.2418,0.3...]	0.2756	0.2843	0.2838

Atividade 10

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.1885,0.3...]	[0.2361,0.3...]	[0.1851,0.4...]	0.253	0.3043	0.3196

Atividade 11

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.1373,0.3...]	[0.1562,0.2...]	[0.1286,0.4...]	0.2202	0.2168	0.2693

Atividade 12

ans = 1x6 cell

	1	2	3	4	5	6
1	[0.2114,0.3...]	[0.2158,0.3...]	[0.1656,0.5...]	0.2599	0.2931	0.3498

3.3

Para calcular o número de passos nas 3 atividades dinâmicas, calculamos a média das frequências no eixo do X e dividimos 60 pelo inverso da frequência (número de passos por minuto 60/periodo).

Decidimos escolher o eixo de X porque de acordo com o enunciado, é o eixo que está na vertical, isto é, numa atividade como caminhar, deverá manter uma frequência relativamente constante, devido ao movimento cima/baixo e baixo/cima.

```

numero_passos=[];
for i=1:3
    %cell2mat convert cell num array convencional
    numero_passos=[numero_passos calculate_steps(cell2mat(freq_filtered(i).values({'xf'})))];
end

for i=1:3
    fprintf("Atividade %d",i);
    fprintf("%f passos por minuto ",numero_passos(i));
end

```

```

Atividade 1
99.710770 passos por minuto
Atividade 2
86.102641 passos por minuto
Atividade 3
92.779434 passos por minuto

```

3.4

Para cada experiência tentámos identificar a categoria das atividades presentes, e o seu grau de sensibilidade e especificidade, isto é, analizar o previsto e o obtido.

Através dos resultados obtidos, é possível perceber que a função está a funcionar com bastante precisão, identificando corretamente praticamente todos os casos. Temos valores de 1 ou muito perto de 1 tanto para a sensibilidade como para a especificidade, isto é, tem boa precisão a identificar casos positivos corretamente bem como casos negativos corretamente.

```

identificar_classes={};
for i=1:8
    [se,sp]=try_identify_class(cell2mat(experiencias.values({i})),labels,i,0);
    d=containers.Map({'sens' 'spec'}, {se,sp});
    identificar_classes=[identificar_classes;containers.Map(i,d)];
end
%obter specificity e sensivity
%identificar_classes(1).values({'sens','spec'})
for i=1:8
    fprintf("Experiencia %d\n",i);
    identificar_classes(i).values({'sens','spec'})
end

```

```

Experiencia 1
ans = 1x2 cell

```

	1	2
1	1	1

```

Experiencia 2
ans = 1x2 cell

```

	1	2
1	0.9565	0.9756

```

Experiencia 3
ans = 1x2 cell

```

	1	2
1	0.95	0.9714

Experiencia 4

ans = 1x2 cell

	1	2
1	0.95	0.9697

Experiencia 5

ans = 1x2 cell

	1	2
1	0.9524	0.9714

Experiencia 6

ans = 1x2 cell

	1	2
1	1	1

Experiencia 7

ans = 1x2 cell

	1	2
1	0.9524	0.973

Experiencia 8

ans = 1x2 cell

	1	2
1	0.95	0.9677

3.5

De seguida, vamos tentar identificar as atividades na experiência e obter o número de atividades reais e o número de atividades previstas. Para tal, utilizámos a mesma função do exercício anterior, calculando a DFT da atividade a identificar e identificando a categoria da atividade. Seguidamente, baseado-nos nas médias e nos valores dos intervalos das frequências relevantes, estimámos valores aproximados para cada atividade. Posteriormente, para identificar a atividade, já sabendo a categoria e com a frequência media obtida pela DFT do segmento, averiguamos de qual atividade se aproxima mais. Esta função apresenta claramente margem para melhorias embora detete corretamente algumas atividades, ainda falha em bastantes delas. Para além disso, não nos foi possível distinguir entre as 4 últimas atividades de transição (Sit_to_lie, Lie_to_sit, Stand_to_lie e Lie_to_stand), devido à proximidade dos intervalos das frequências nos 3 eixos.

```

identificar=[];
for i=1:8
    [real,prev]=try_identify(cell2mat(experiencias.values({i})),labels,i);
    d=containers.Map({'real' 'prev'},[real prev]);
    identificar=[identificar ;containers.Map(i,d)];
end
%obter specificity e sensivity
%identificar(1).values({'real','prev'})
for i=1:8
    fprintf("Experiencia %d\n",i);
    cell2mat(identificar(i).values({'real'}))
    cell2mat(identificar(i).values({'prev'}))
end

```

```

Experiencia 1
ans = 1x12
    4     3     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    4     3     3     5     1     0     0     1     5     0     0     0
Experiencia 2
ans = 1x12
    4     3     4     2     2     2     1     1     1     1     1     1
ans = 1x12
    4     3     5     2     1     2     2     2     2     0     0     0
Experiencia 3
ans = 1x12
    2     3     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    2     5     0     4     2     1     0     2     4     0     0     0
Experiencia 4
ans = 1x12
    2     3     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    3     2     3     3     2     2     0     1     4     0     0     0
Experiencia 5
ans = 1x12
    2     4     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    1     2     5     4     1     2     1     1     4     0     0     0
Experiencia 6
ans = 1x12
    2     3     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    1     2     5     4     0     2     1     1     4     0     0     0
Experiencia 7
ans = 1x12
    2     3     4     2     2     2     1     1     1     1     1     1
ans = 1x12
    0     3     5     4     1     2     2     0     4     0     0     0
Experiencia 8
ans = 1x12
    2     3     3     2     2     2     1     1     1     1     1     1
ans = 1x12
    0     1     6     4     1     2     1     1     4     0     0     0

```

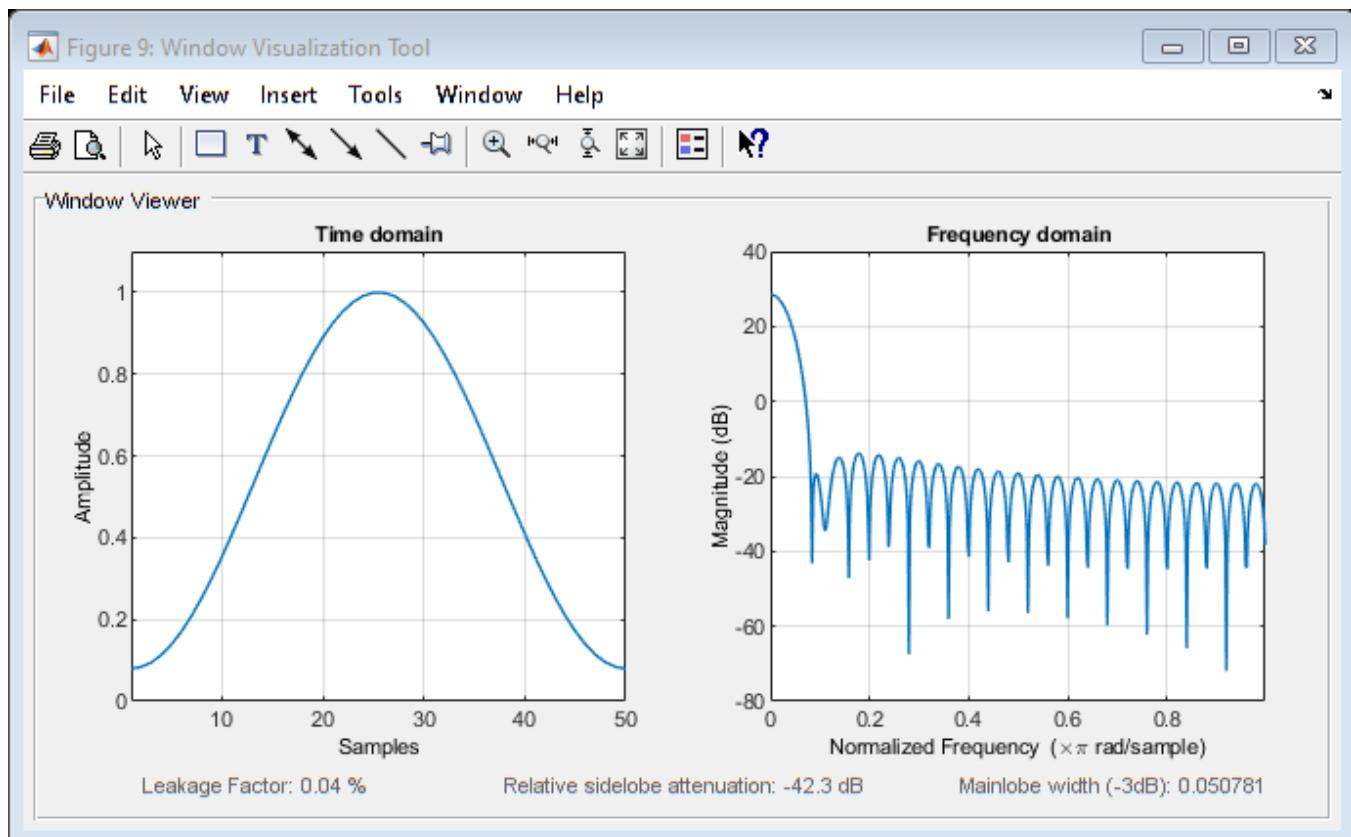
4.

4.1

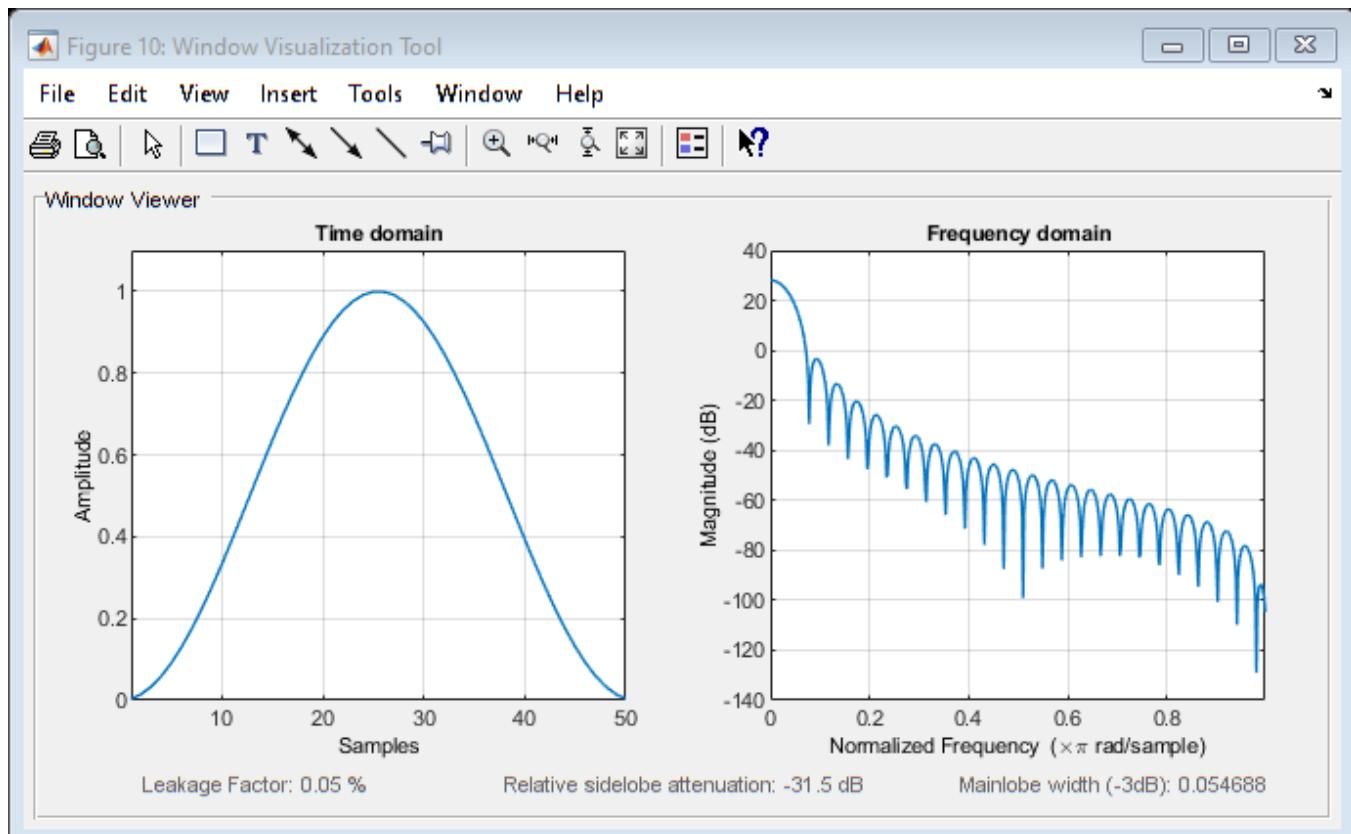
Aqui aplicámos a STFT (aplicámos a DFT numa janela deslizante) ao eixo Z da experiencia 2, com uma sobreposição de 50% em relação ao tamanho da janela, e damos plot a todas as dfts sobrepostas relativas a cada tipo de janela (utilizámos janelas de Hamming, Hann, Blackman e retangular).

```
%ficheiro escolhido: user1 experiencia 2
test_windows_stft(experiencias(2), cell2mat( activity_frame(1).values({2})) )
```

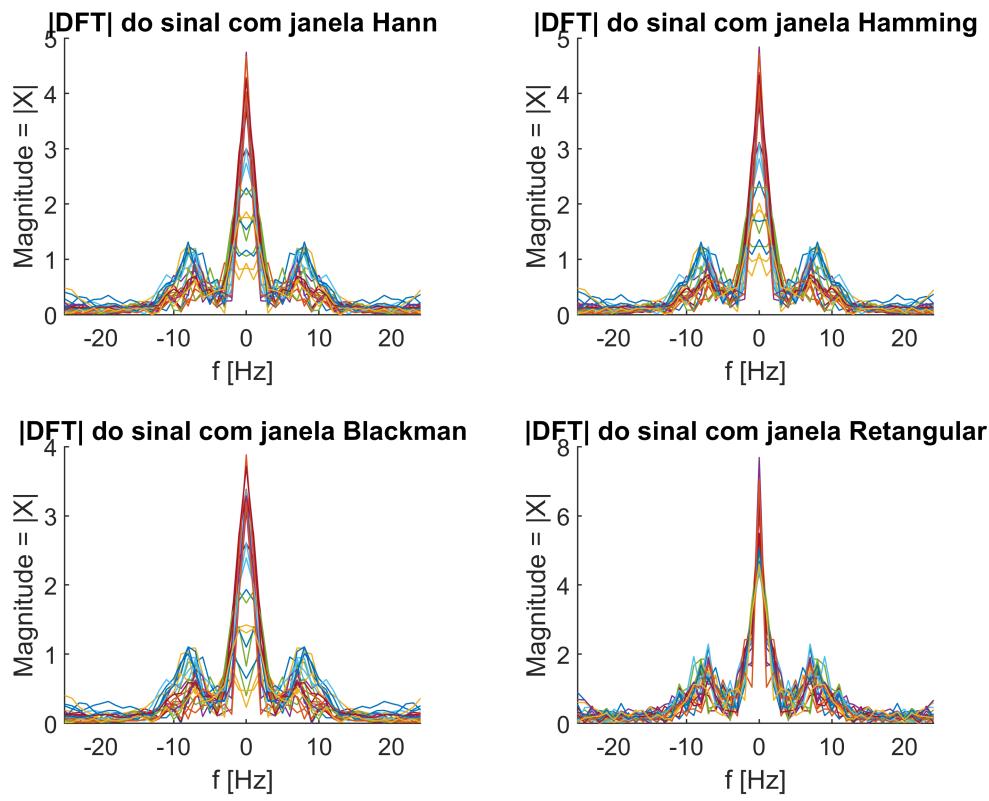
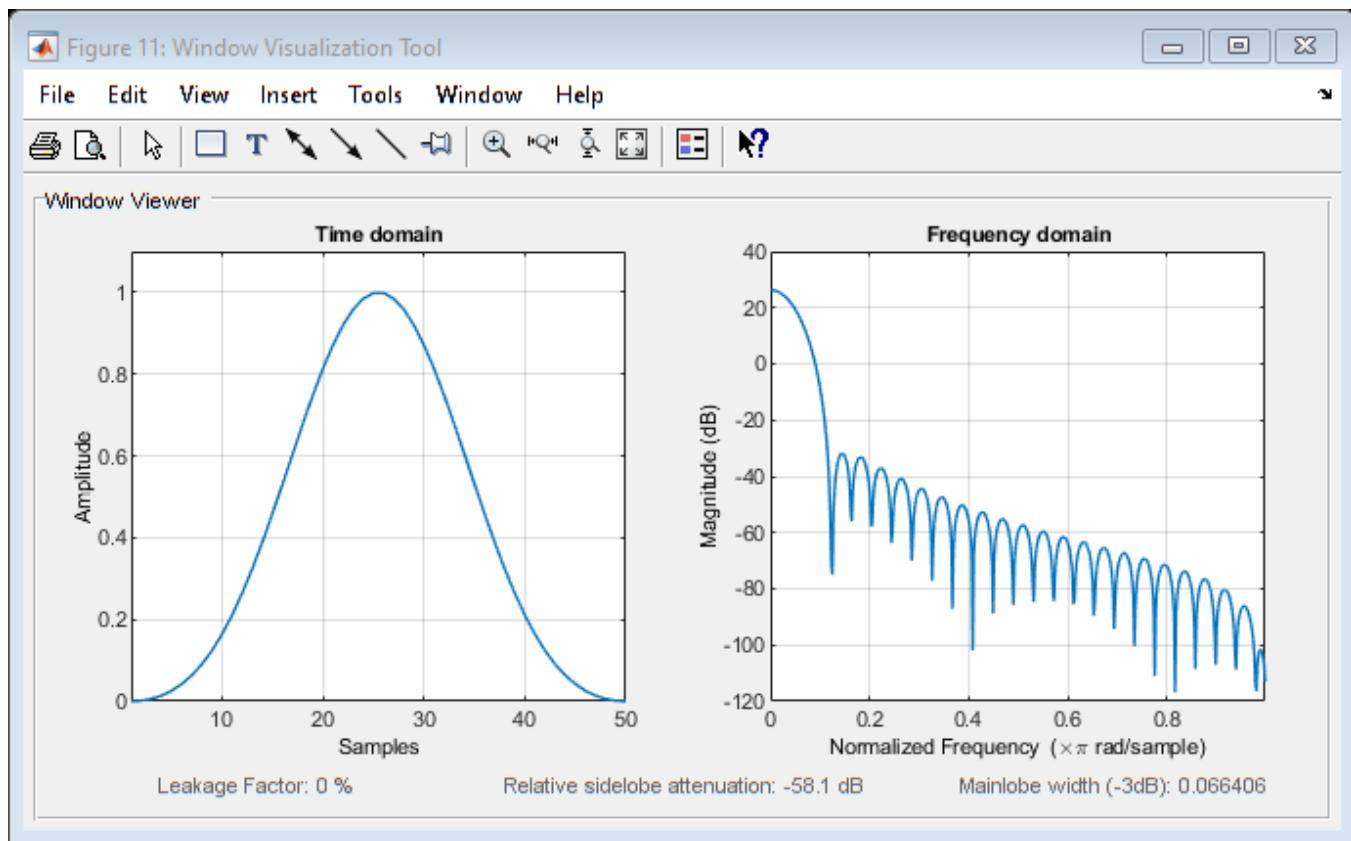
Janela Hamming



Janela Hann



Janela Blackman



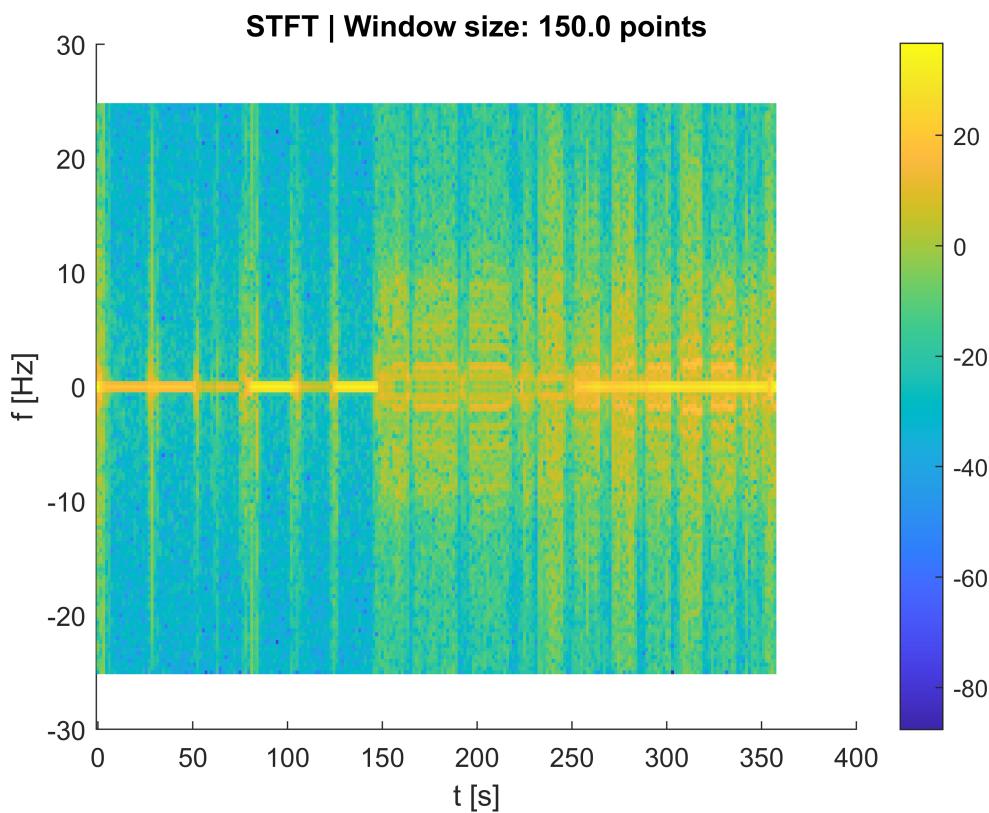
Após a análise dos gráficos das DFTs bem como das janelas, considerámos que a janela mais adequada seria a de Hann, pois mantém um espectro grande de frequências no centro, mas cortando as frequências nas

"pontas", isto é, após o cálculo da DFT, vamos ter menos descontinuidades, geradas pela natureza da DFT, diminuindo assim frequências indesejadas. Apesar disto, nesta aplicação tanto a janela de Hamming como a de Blackman seriam igualmente válidas.

4.2

Aplicação da STFT "à mão", isto é, aplicação da DFT a janelas de tamanho fixo, segmentando um sinal.

```
%testar a STFT num ficheiro
fs = 50;
experiencia_eixos = experiencias(3);
calculate_stft(experiencia_eixos(:,3) ,150,fs,1);
```



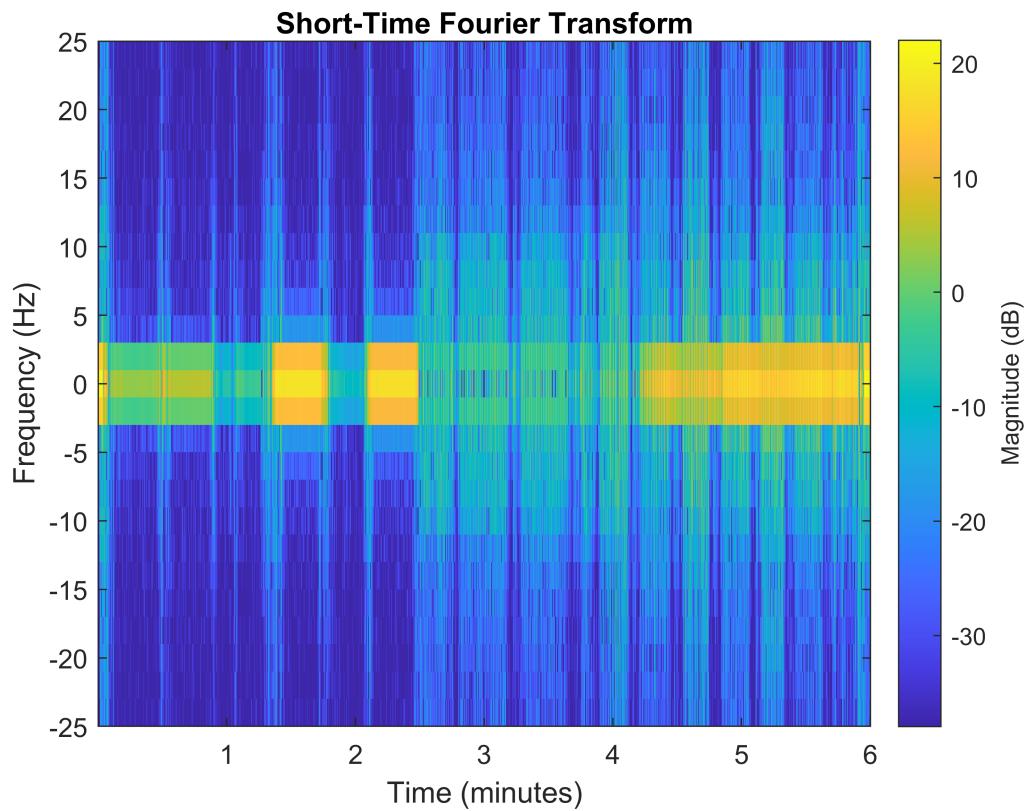
4.3

Cálculo da STFT com janela de Hann, com sobreposição de 50% da janela, para todo o ficheiro 3, no eixo Z.

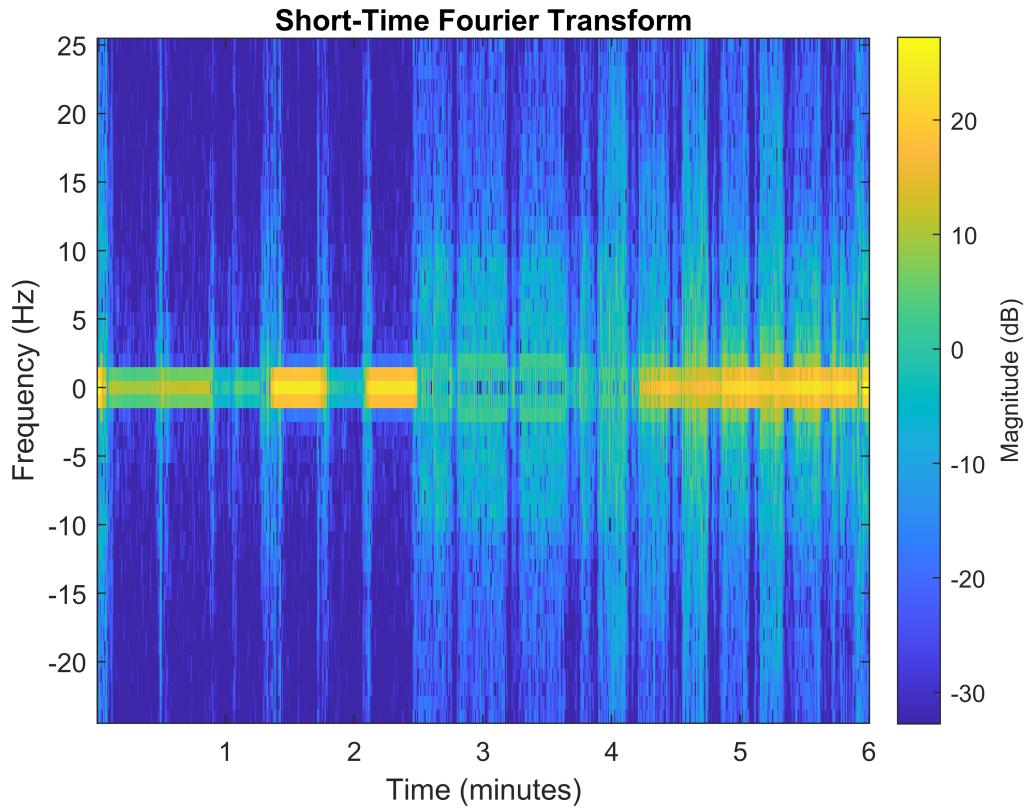
Calculámos o menor tamanho de uma atividade, e tivemos em conta esse valor (73 pontos) para testar os tamanhos das janelas da STFT.

Testámos vários tamanhos de janelas para estudar o seu impacto nas frequências obtidas. Ao diminuir o tamanho da janela, a resolução em frequência diminui. Posto isto, é necessário encontrar um equilíbrio entre resolução em frequência e resolução temporal. Neste caso, consideramos que uma janela adequada seria com tamanho a variar entre 100 e 150 pontos.

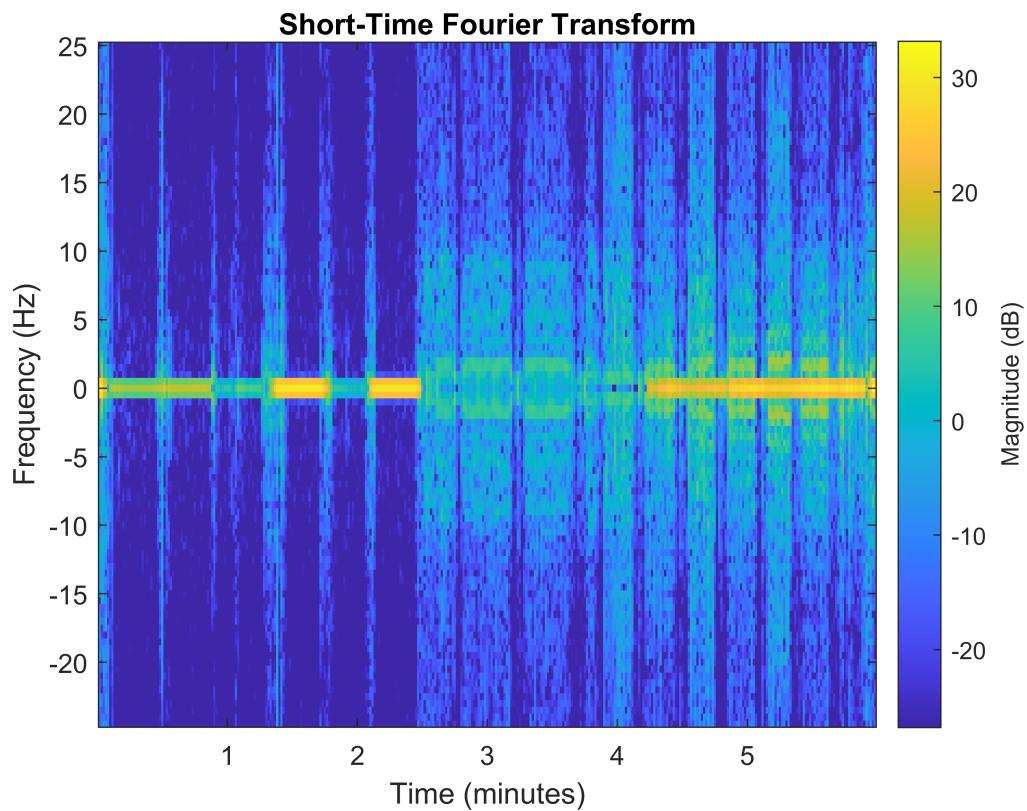
```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(25),'OverlapLength',13)
```



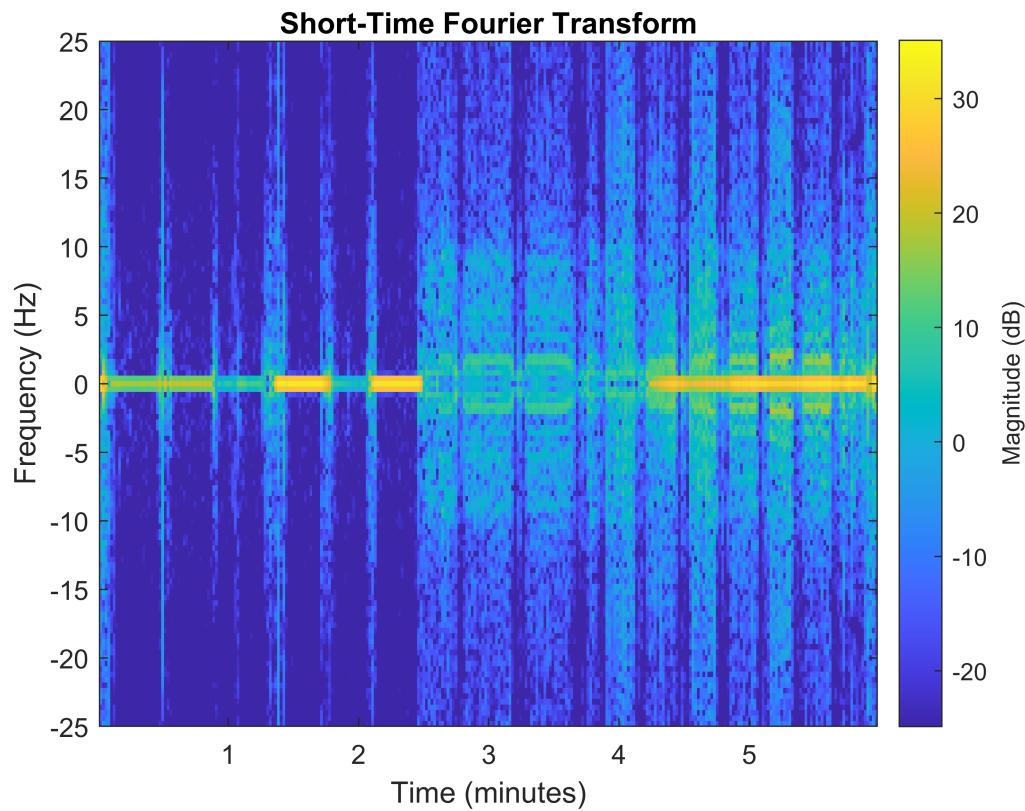
```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(50),'OverlapLength',25)
```



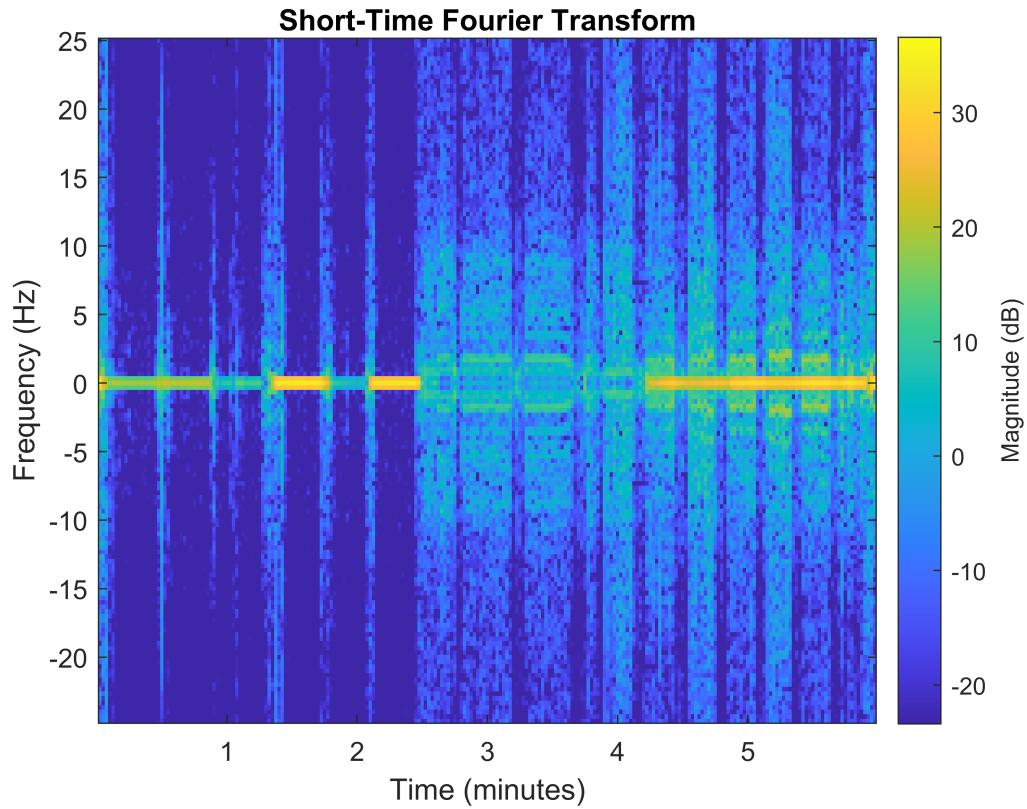
```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(100), 'OverlapLength',50)
```



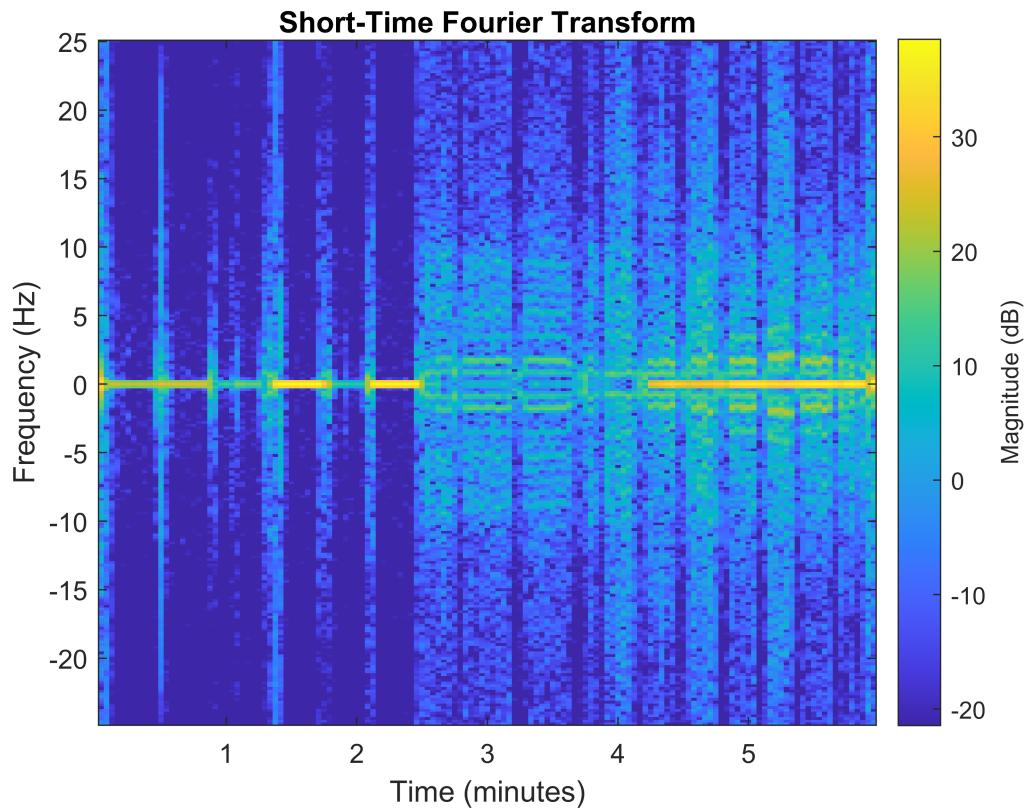
```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(125),'OverlapLength',63)
```



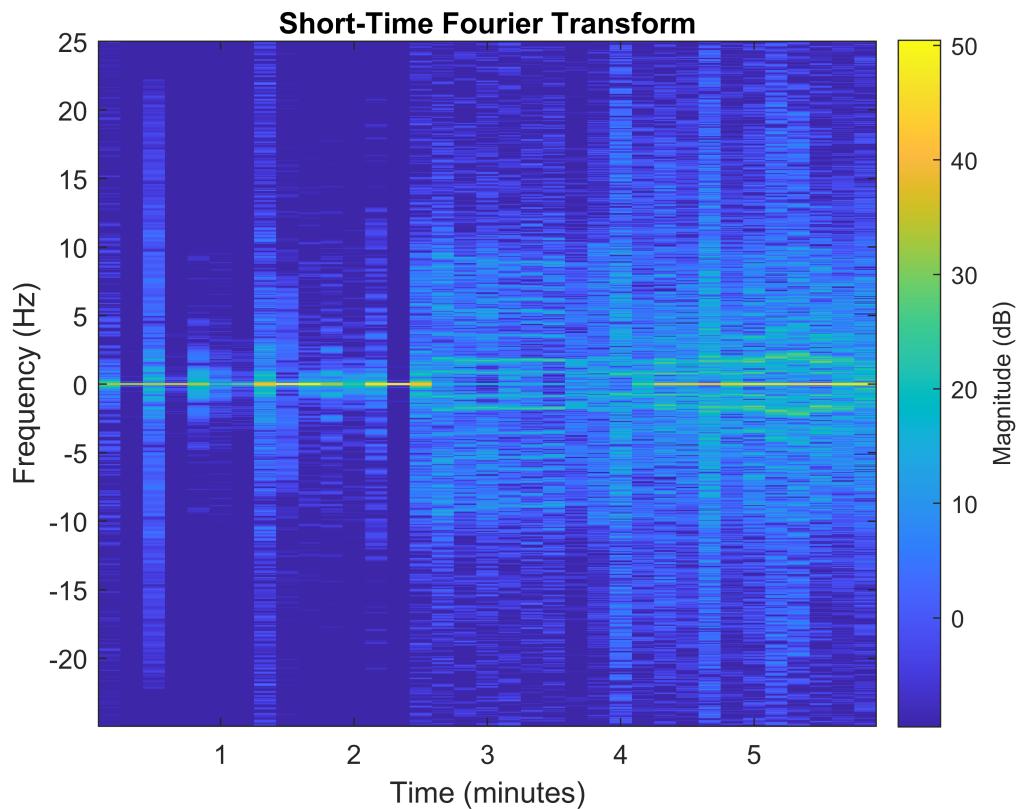
```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(150),'OverlapLength',75)
```



```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(250), 'OverlapLength',125)
```



```
stft( experiencia_eixos(:,3) ,fs,'Window',hann(1000),'OverlapLength',500)
```



Conclusão

Ao longo deste trabalho desenvolvemos competências relacionadas com análise e tratamento de dados, através do uso de ferramentas como, por exemplo, transformadas de Fourier.

Além disso, aprofundámos também as nossas competências no domínio do MATLAB.

Por fim, achamos que os resultados obtidos poderiam ser melhorados aprofundando ainda mais a análise dos dados.

A função que se segue serve para encontrar o(s) tempo(s) de início e de fim em que uma certa atividade decorreu naquela experiência.

```
function fin = get_activity_file(experience,user,activity,labels)
fin = [];
i = 1;
%find correct file lines for the user given
while (labels(2,i) ~= user)
    i = i+1;
end
while( labels(1,i) ~= experience)
    i = i+1;
end
%get frames for the activity requested
```

```

while( labels(2,i) == user && labels(1,i) == experiencia)

    if( labels(3,i) == activity)
        fin = [fin ; [labels(4,i) , labels(5,i)] ];
    end
    i = i+1;
end
end

```

A função que se segue serve para fazer os 3 gráficos que representam cada eixo para cada experiência. Além disso, é marcado com cores distintas a atividade que se realizou nos diferentes intervalos de tempo, como também a sua identificação em texto.

```

function [] = print_with_labels(file_data,labels,activityLabels,experiencia)
colors = [ [0.9 0.1 0.1] ; [0.8500 0.3250 0.0980] ; [0.9290 0.6940 0.1250] ;[0.4940 0.1840
i = 1;
j = i;
x_vals = [];
activity = {};
%Get Labels
while(labels(1,i) ~= experiencia)
    i = i+1;
end
j = i;

%X
nexttile
plot([1:length(file_data)],file_data(:,1))
hold on
title("X")
min_y = min(file_data(:,1));
max_y = max(file_data(:,1));
while (labels(1,j) == experiencia)
    if mod(j,2) == 0
        l = text( labels(4,j) , min_y , activityLabels{2}{labels(3,j)} );
    else
        l = text( labels(4,j) , max_y, activityLabels{2}{labels(3,j)} );
    end
    set(l,'Rotation',50);
    hold on
    plot( [labels(4,j):labels(5,j)] , file_data( labels(4,j):labels(5,j) , 1 ) , 'Color',
j = j+1;
end
j = i;

%Y
nexttile
plot([1:length(file_data)],file_data(:,2))
title("Y")
min_y = min(file_data(:,2));
max_y = max(file_data(:,2));
while (labels(1,j) == experiencia)
    if mod(j,2) == 0

```

```

        l = text( labels(4,j) , min_y , activityLabels{2}{labels(3,j)} );
    else
        l = text( labels(4,j) , max_y, activityLabels{2}{labels(3,j)} );
    end
    set(l,'Rotation',50);
    hold on
    plot( [labels(4,j):labels(5,j)] , file_data( labels(4,j):labels(5,j) , 2) , 'Color',
    j = j+1;
end
j = i;

%Z
nexttile
plot([1:length(file_data)],file_data(:,3))
title("Z")
min_y = min(file_data(:,3));
max_y = max(file_data(:,3));
while (labels(1,j) == experiencia)

    if mod(j,2) == 0
        l = text( labels(4,j) , min_y , activityLabels{2}{labels(3,j)} );
    else
        l = text( labels(4,j) , max_y, activityLabels{2}{labels(3,j)} );
    end
    set(l,'Rotation',50);
    hold on
    plot( [labels(4,j):labels(5,j)] , file_data( labels(4,j):labels(5,j) , 3) , 'Color',
    j = j+1;
end
j = i;
end

```

A função que se apresenta de seguida serve para calcular a DFT de uma atividade numa experiência, obtendo então as frequências relevantes em cada eixo.

```

%%Calculo DFT de cada atividade para cada experiencia
function[x,y,z] = DFT_activity(file,times,plot_magnitude,plot_freqs)
fs = 50; %Hz, sampling frequency
x = [];
y = [];
z = [];
%Cm_x = [];
%Cm_y = [];
%Cm_z = [];

[a b] = size(times);

%if activity only appears once in the file
if a == 1
    flag_line = 1;
else
    flag_line = 0;
end

```

```

for i = 1:a
    if flag_line == 1
        dft_x = fftshift( fft( file(times(1):times(2) ,1) ) );
        dft_y = fftshift( fft( file(times(1):times(2) ,2) ) );
        dft_z = fftshift( fft( file(times(1):times(2) ,3) ) );
    else
        dft_x = fftshift( fft( file(times(i,1):times(i,2) ,1) ) );
        dft_y = fftshift( fft( file(times(i,1):times(i,2) ,2) ) );
        dft_z = fftshift( fft( file(times(i,1):times(i,2) ,3) ) );
    end

N = length(dft_x);
fq = [-fs/2:fs/N:fs/2-fs/N];

%get positive values
m_x = abs(dft_x);
m_y = abs(dft_y);
m_z = abs(dft_z);

%clean
m_x( m_x<0.001)=0;
m_y( m_y<0.001)=0;
m_z( m_z<0.001 )=0;

m_x( abs(fq) <0.15) = 0;
m_y( abs(fq) <0.15) = 0;
m_z( abs(fq) <0.15) = 0;

%Cm can be used to identify activities
%Cm_x = [Cm_x (m_x/N)'];
%Cm_y = [Cm_y (m_y/N)'];
%Cm_z = [Cm_z (m_z/N)'];

threshold_x = 0.8*max(m_x);
threshold_y = 0.8*max(m_y);
threshold_z = 0.8*max(m_z);

if plot_magnitude == 1
    figure();

    %plot DFTS MAGNITUDE
    nexttile;
    yline(threshold_x);
    hold on;
    plot(fq,m_x)
    title('|DFT| do sinal eixo X');
    ylabel('Magnitude = |X|');
    xlabel('f [Hz]');

    nexttile;
    yline(threshold_y);
    hold on;
    plot(fq,m_y)

```

```

        title('|DFT| do sinal eixo Y');
        ylabel('Magnitude = |X|');
        xlabel('f [Hz]');

        nexttile;
        yline(threshold_z);
        hold on;
        plot(fq,m_z)
        title('|DFT| do sinal eixo Z');
        ylabel('Magnitude = |X|');
        xlabel('f [Hz]');

    end

%find relevant frequencies for each axis
%X
[pks,locs] = findpeaks(m_x,'MinPeakHeight',threshold_x);
f_relevant_x = fq(locs);
x = [x,f_relevant_x( f_relevant_x>0 )];

%Y
[pks,locs] = findpeaks(m_y,'MinPeakHeight',threshold_y);
f_relevant_y = fq(locs);
y = [y,f_relevant_y( f_relevant_y>0 )];

%Z
[pks,locs] = findpeaks(m_z,'MinPeakHeight',threshold_z);
f_relevant_z = fq(locs);
z = [z,f_relevant_z( f_relevant_z>0 )];

if plot_freqs == 1

    figure();
    m_x_zeros = m_x;
    for k =1:length(fq)
        if(~ismember(fq(k),f_relevant_x))
            m_x_zeros(k) = 0;
        end
    end
    %m_x_zeros( m_x_zeros < threshold_x) = 0;

    m_y_zeros = m_y;
    for k =1:length(fq)
        if(~ismember(fq(k),f_relevant_y))
            m_y_zeros(k) = 0;
        end
    end
    %m_y_zeros( m_y_zeros < threshold_y) = 0;

    m_z_zeros = m_z;
    for k =1:length(fq)
        if(~ismember(fq(k),f_relevant_z))
            m_z_zeros(k) = 0;
        end
    end

```

```

    end
    %m_z_zeros( m_z_zeros < threshold_z) = 0;

    %plot relevant freqs
    nexttile;
    hold on;
    plot(fq,m_x_zeros)
    title('Frequencias relevantes do sinal eixo X');
    ylabel('Magnitude = |X|');
    xlabel('f [Hz]');

    nexttile;
    hold on;
    plot(fq,m_y_zeros)
    title('Frequencias relevantes do sinal eixo Y');
    ylabel('Magnitude = |X|');
    xlabel('f [Hz]');

    nexttile;
    hold on;
    plot(fq,m_z_zeros)
    title('Frequencias relevantes do sinal eixo Z');
    ylabel('Magnitude = |X|');
    xlabel('f [Hz]');
end
end

```

Nesta função recebemos todas as frequências relevantes de uma atividade. Seguidamente, filtramos estas frequências, agrupando em intervalos de 2Hz, e guardando apenas os intervalos com maior número de frequências. Para além disso, retornamos também a média das frequências relevantes de cada eixo.

```

%%Obter freq_relevantes e carateristicas para cada experiencia
function[xf,yf,zf,med_x,med_y,med_z] = get_freqs_filtered(x,y,z)

xf = [];
yf = [];
zf = [];
step = 2;

med_x = mean(x);
med_y = mean(y);
med_z = mean(z);

%  

for k = 0:step:max(x)

    intervalo = x( x>=k & x <k+step );

    desvio_pad = std(intervalo);
    med = mean( intervalo );
    nr_elem = length( intervalo );

```

```

    xf = [xf;[med-desvio_pad,med+desvio_pad,nr_elem]];
end

%filter the frequencias by the groups that have the most values
%criar intervalo com base na media e no desvio padrao
%meter a 0 caso o limite inferior seja inferior a 0
number_max_freqs = max(xf(:,3));
xf = xf( xf(:,3) >= 0.5*number_max_freqs , 1:2 );
xf( : , xf(:,1) < 0) = 0;

%Y
for k = 0:step:max(y)

    intervalo = y( y>=k & y <k+step );

    desvio_pad = std(intervalo);
    med = mean( intervalo );
    nr_elem = length( intervalo );

    yf = [yf;[med-desvio_pad,med+desvio_pad,nr_elem]];
end

number_max_freqs = max(yf(:,3));
yf = yf( yf(:,3) >= 0.5*number_max_freqs , 1:2 );
yf( : , yf(:,1) < 0) = 0;

%Z
for k = 0:step:max(z)
    intervalo = z( z>=k & z <k+step );

    desvio_pad = std(intervalo);
    med = mean( intervalo );
    nr_elem = length( intervalo );

    zf = [zf;[med-desvio_pad,med+desvio_pad,nr_elem]];
end

number_max_freqs = max(zf(:,3));
zf = zf( zf(:,3) >= 0.5*number_max_freqs , 1:2 );
zf( : , zf(:,1) < 0) = 0;
end

```

A função que se segue serve para prever o numero de categorias de atividades (dinâmica, estática e transição) presentes numa experiência, e assim comparar com os resultados reais com os previstos, obtendo a especificidade e a sensibilidade.

```

function [sensitivity,specificity] = try_identify_class(file, labels, experience, plotit)

%plot do ficheiro no eixo x
if plotit
    plot([1:length(file)],file(:,1))
    hold on;

```

```

    colors = ['r' 'c' 'm'];
end

act = labels(:, labels(1,:) == experience);

nr_real_estaticas = 0;
nr_real_dinamicas= 0;
nr_real_transicao= 0;

nr_prev_estaticas = 0;
nr_prev_dinamicas= 0;
nr_prev_transicao= 0;

nr_verdadeiros_negativos = 0;

for i = 1:length(act)
    %loop por todas as atividades do desta experiencia
    correct_activity = act(3,i);
    activity_size = act(5,i) - act(4,i);

    %atualizar o nr real da atividade
    if correct_activity <= 3
        nr_real_dinamicas = nr_real_dinamicas + 1;
    elseif correct_activity <= 6
        nr_real_estaticas= nr_real_estaticas+ 1;
    else
        nr_real_transicao= nr_real_transicao+ 1;
    end

    %prever
    %calculate dft and get frequencies
    [c] = get_class(file,[act(4,i) act(5,i)]);

    if c == "estatica" | c == "laying"
        %estatica
        nr_prev_estaticas = nr_prev_estaticas + 1 ;

        if correct_activity > 3 & correct_activity < 6
            nr_verdadeiros_negativos = nr_verdadeiros_negativos + 2;
        end

        if plotit
            plot([act(4,i):act(5,i)], file( act(4,i):act(5,i) , 1), colors(1) );
        end

    elseif c == "dinamica"
        %dinamica
        nr_prev_dinamicas = nr_prev_dinamicas + 1;

        if correct_activity <= 3
            nr_verdadeiros_negativos = nr_verdadeiros_negativos + 2;
        end

        if plotit

```

```

        plot([act(4,i):act(5,i)], file( act(4,i):act(5,i) , 1), colors(2) );
    end

elseif c == "transicao"
%transicao
nr_prev_transicao = nr_prev_transicao + 1;

if correct_activity > 6
    nr_verdadeiros_negativos = nr_verdadeiros_negativos + 2;
end

if plotit
    plot([act(4,i):act(5,i)], file( act(4,i):act(5,i) , 1), colors(3) );
end

end
end

%averiguar sensibilidade e especificidade
nr_verdadeiros_positivos = 0;
nr_falsos_negativos = 0;
nr_falsos_positivos = 0;

test = nr_real_dinamicas - nr_prev_dinamicas;

if test > 0
    nr_falsos_negativos = nr_falsos_negativos + test;
    nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_prev_dinamicas;

else
    nr_falsos_positivos = nr_falsos_positivos + abs(test);
    nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_real_dinamicas;
end

%disp(nr_real_dinamicas);
%disp(nr_prev_dinamicas);

test = nr_real_estaticas - nr_prev_estaticas;

if test > 0
    nr_falsos_negativos = nr_falsos_negativos + test;
    nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_prev_estaticas;

else
    nr_falsos_positivos = nr_falsos_positivos + abs(test);
    nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_real_estaticas;
end

%disp(nr_real_estaticas);
%disp(nr_prev_estaticas);

test = nr_real_transicao - nr_prev_transicao;

if test > 0

```

```

nr_falsos_negativos = nr_falsos_negativos + test;
nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_prev_transicao;

else
    nr_falsos_positivos = nr_falsos_positivos + abs(test);
    nr_verdadeiros_positivos = nr_verdadeiros_positivos + nr_real_transicao;
end

%disp(nr_real_transicao);
%disp(nr_prev_transicao);

sensitivity = nr_verdadeiros_positivos / (nr_verdadeiros_positivos + nr_falsos_negativos);
specificity = nr_verdadeiros_negativos / (nr_verdadeiros_negativos + nr_falsos_positivos);
end

```

A função que se segue serve para prever a categoria de um certo intervalo de tempo (transição, estática, dinâmica). Assim, é calculada a DFT para esse intervalo de tempo, recorrendo à função desenvolvida anteriormente (DFT_activity), posteriormente calculamos a média dos 3 eixos e analizando os valores obtidos para cada atividade, consideramos alguns limites para considerar a categoria de uma atividade (se a media de X for maior do que 1.4 e menor do que 2 então consideramos que é dinâmica, se a média de X for maior do que 2 consideramos que é estática , se a media de y for maior do que 1.3 consideramos que é estática, se a media de X for menor do que 1.4 e se a média de Y for menor do que 1.3, então verificamos pelo tamanho da atividade, se for menor do que 550 consideramos que é de transição, caso contrário, é estática).

```

function[class,med_x,med_y,med_z] = get_class(file, times)
activity_size = times(2) - times(1);
%prever
%calculate dft and get frequencies
intervalo = [times(1) times(2)];
[x,y,z] = DFT_activity(file,intervalo,0,0);

med_x = mean(x);
med_y = mean(y);
med_z = mean(z);

if med_x > 1.4
    %dinamica ou estatica
    if med_x > 2
        %Estatica - Standing ou Sit
        class = "estatica";
    else
        %alguma dinamica
        class = "dinamica";
    end
elseif med_y > 1.3
    %Laying - Estatica
    class = "laying";
else
    % frequencia x pequena, frequencia em y pequena
    % pode ser estatica ou de transicao
    % vamos comparar através do tamanho da atividade
    if activity_size < 550

```

```

        %transicao, atividade rápida
        class = "transicao";
    else
        %estatica - sit
        class = "estatica";
    end
end
end

```

A função seguinte serve para calcular o número de passos para cada uma das 3 atividades dinâmicas (WALKING, WALKING_UP,WALKING_DOWN). Assim, fazemos a média das frequências relevantes de uma das 3 atividades e o número de passos por minuto é a divisão de 1 minuto por 1 período ($60/(1/\text{mean(freqs_relevantes})$).

```

function[num_steps] = calculate_steps(freqs)
freqs_relevantes = [];
[a b] = size(freqs);

%Correr todos os intervalos das frequências relevantes (só estamos a
%receber 1 eixo aqui)
for i = 1:a
    get_freq = mean(freqs(i)); %fazer a média do intervalo
    freqs_relevantes = [ freqs_relevantes ; get_freq]; %adicionar na matriz de frequências

end
%Número de passos por minuto é 60 (segundos) a dividir pelo Período,
%isto é, o tempo de 1 ciclo completo.  $60/(1/\text{freq})$ 
num_steps = 60/(1/mean(freqs_relevantes));
end

```

A função que se segue serve para tentar identificar as atividades presentes em cada experiência, sendo retornado o número de atividades reais e o número de atividades previstas. Os limites impostos são fruto da análise de frequências reais.

```

function[nr_real_activities,nr_prev_activities] = try_identify(file,labels,experience)

act = labels(:, labels(1,:) == experience);
nr_real_activities = [0 0 0 0 0 0 0 0 0 0];
nr_prev_activities = [0 0 0 0 0 0 0 0 0 0];

for i = 1:length(act)
    %loop por todas as atividades desta experiência
    correct_activity = act(3,i);
    activity_size = act(5,i) - act(4,i);
    intervalo = [act(4,i) act(5,i)];

    %atualizar o nr real da atividade
    nr_real_activities(correct_activity) = nr_real_activities(correct_activity) + 1;

    %prever
    %calculate dft and get frequencies
    %PREVER ATIVIDADE

```

```

[prever_classe,med_x,med_y,med_z] = get_class(file,intervalo);

%README - ISTO NÃO ESTÁ COMPLETO E NÃO ESTÁ A FUNCIONAR MUITO BEM
if prever_classe == "laying"
    nr_prev_activities(6) = nr_prev_activities(6) + 1;

elseif prever_classe == "dinamica"
    %VERIFICAR QUAL DAS DINAMICAS É
    %nr_prev_activities(2) = nr_prev_activities(2) + 1;
    %disp(med_y)
    dif_walk = abs(1.74-med_y);
    dif_walkUp = abs(1.49-med_y);
    dif_walkDown = abs(1.64 -med_y);

    if med_y > 1.82
        %Walking down
        nr_prev_activities(3) = nr_prev_activities(3) + 1;

    elseif dif_walkUp < dif_walk & dif_walkUp < dif_walkDown
        %Walking Up
        nr_prev_activities(2) = nr_prev_activities(2) + 1;

    elseif dif_walk < dif_walkDown & dif_walk < dif_walkUp
        %Walking
        nr_prev_activities(1) = nr_prev_activities(1) + 1;

    else
        %Walk Down
        nr_prev_activities(3) = nr_prev_activities(3) + 1;
    end

elseif prever_classe == "estatica"
    %nas estaticas só sobra sitting e standing
    %amplitude baixa

    %nr_prev_activities(5) = nr_prev_activities(5) + 1;
    dif_sit = abs(4.2551 - med_x);
    dif_stand = abs(3.9268 - med_x);

    if med_x < 2.30
        %tem de ser sitting, no standing nunca temos frequencias
        %abaixo de 2.38
        nr_prev_activities(4) = nr_prev_activities(4) + 1;

    elseif med_x > 5.5
        %tem de ser sitting, no standing nunca temos frequencias
        %acima de 5.39
        nr_prev_activities(4) = nr_prev_activities(4) + 1;
    elseif dif_sit < dif_stand
        %se está no intervalo comum às duas atividades, vamos pelo
        %que está mais perto da média
        %neste caso, mais perto da media do sitting

```

```

        nr_prev_activities(4) = nr_prev_activities(4) + 1;
    else
        %standing
        nr_prev_activities(5) = nr_prev_activities(5) + 1;

    end

elseif prever_classe == "transicao"
    %TODO VERIFICAR QUAL DAS TRANSICAO É

    if med_x > 0.9
        %STAND_TO_SIT
        nr_prev_activities(7) = nr_prev_activities(7) + 1;

    elseif med_x > 0.35
        %7 ou 8
        dif_stand_to_sit = abs(med_x - 0.9094);
        dif_sit_to_stand = abs(med_x - 0.6163);

        if dif_stand_to_sit < dif_sit_to_stand
            %STAND_TO_SIT
            nr_prev_activities(7) = nr_prev_activities(7) + 1;

        else
            %SIT_TO_STAND
            nr_prev_activities(8) = nr_prev_activities(8) + 1;
        end

    else
        dif_stand_to_sit = abs(med_z - 0.4533);

        dif_rest = abs(med_z - 0.3056); %média das médias do eixo de z das 4 ultimas at

        if dif_stand_to_sit < dif_rest
            %STAND_TO_SIT
            nr_prev_activities(7) = nr_prev_activities(7) + 1;

        else
            %Se não for 7 ou 8, vai para a atividade 9
            nr_prev_activities(9) = nr_prev_activities(9) + 1;
        end
    end
end
end

```

Esta função calcula várias DFTs com uma janela deslizante duma atividade, e aplica vários tipos de janelas, com o objetivo de decidirmos qual a melhor para o propósito deste projeto.

Resumidamente, calcula a STFT mas apenas de uma atividade, com uma sobreposição de 50% do tamanho da janela.

```
function [] = test_windows_stft(file,times)
fs = 50; %Hz
frame_size = 1; %em segundos
N = length(file);
tam = round(fs * frame_size);
passo = round(tam/2);

%windows
hamm = hamming(tam);
hann = hanning(tam);
black = blackman(tam);

%Plot das janelas
disp("Janela Hamming")
wvtool(hamm);
disp("Janela Hann")
wvtool(hann);
disp("Janela Blackman")
wvtool(black);

figure();
%ATIVIDADE ESCOLHIDA: 1 - WALKING

if ( mod(tam,2) == 0 )
    f_frame=-fs/2:fs/tam:fs/2-fs/tam;
else
    f_frame=-fs/2+fs(2*tam):fs/tam:fs/2+fs/(2*tam);
end

%Calcular DFT com diferentes janelas para o segmento desta atividade,
%no eixo de Z

%hann
nexttile;
for i = times(1,1) : passo : times(1,2)-tam
    dft = abs(fftshift(fft( file( i:i+tam-1 , 3) .* hann )));

    hold on;
    plot(f_frame,dft)
end
title('|DFT| do sinal com janela Hann');
ylabel('Magnitude = |X|');
xlabel('f [Hz]');

%hamming
nexttile;
for i = times(1,1) : passo : times(1,2)-tam
    dft = abs(fftshift(fft( file( i:i+tam-1 , 3) .* hamm )));

    hold on;
```

```

    plot(f_frame,dft)
end
title('|DFT| do sinal com janela Hamming');
ylabel('Magnitude = |X|');
xlabel('f [Hz]');

%blackman
nexttile;
for i = times(1,1) : passo : times(1,2)-tam
    dft = abs(fftshift(fft( file( i:i+tam-1 , 3) .* black )));

    hold on;
    plot(f_frame,dft)
end
title('|DFT| do sinal com janela Blackman');
ylabel('Magnitude = |X|');
xlabel('f [Hz]');

%retangular
nexttile;
for i = times(1,1) : passo : times(1,2)-tam
    dft = abs(fftshift(fft( file( i:i+tam-1 , 3) )));

    hold on;
    plot(f_frame,dft)
end
title('|DFT| do sinal com janela Retangular');
ylabel('Magnitude = |X|');
xlabel('f [Hz]');

end

```

A seguinte função calcula a STFT para um determinado segmento, com um certo tamanho de janela e para uma certa frequência de amostragem. É aplicada uma sobreposição de 50% do tamanho da janela.

```

%4.2
function [stft,f_frame,time_frames] = calculate_stft(segment,win_size,fs,plotit)

stft = [];
time_frames = [];
n_frame = 0;
N = length(segment);

hann = hanning(win_size);

if ( mod(win_size,2) == 0 )
    f_frame = -fs/2:fs/win_size:fs/2-fs/win_size;
else
    f_frame = -fs/2+fs/(2*win_size):fs/win_size:fs/2-fs/(2*win_size);
end

passo = round(win_size/2);

```

```

for i = 1 : passo : N-win_size
    mag_dft = abs(fftshift(fft( segment(i:i+win_size-1) .* hann )));

    time_frames = [time_frames n_frame*passo/fs];
    n_frame = n_frame + 1;

    stft = [stft mag_dft];

end

if plotit
    figure();
    imagesc('XData',time_frames,'YData',f_frame,'CData',mag2db(stft))
    %surf(time_frames,f_frame,mag2db(stft))
    colorbar
    xlabel('t [s]')
    ylabel('f [Hz]')
    title(['STFT | Window size: ',num2str(win_size,'%.1f'), ' points'])
end

```