Universidade do Minho

Processamento digital de sinal

Deteção de segmentos de fala

PROCESSOS ESTOCÁSTICOS

Nuno Barroso Nº 50104

Mestrado Integrado em Engenharia de Eletrónica Industrial e Computadores

Índice

1. Introdução

O objectivo deste trabalho será analisar um ficheiro de áudio detetar os momentos correspondentes à fala e separá-los dos momentos de ruido. Para tal criou-se uma aplicação em *Matlab* composta por uma função de deteção de segmentos de fala e sua separação e uma interface gráfica aonde são mostrados os resultados que posteriormente serão analisados.

Através de processos estocásticos é possível calcular o modelo de ruído (variância, media). Tendo estes parâmetros consegue-se saber se a amostra faz parte do ruído ou da fala, através da curva de Gauss, verifica-se se a amostra se encontra dentro ou fora do *outlier*, se estiver fora é fala se estiver dentro é ruído.

2. Fundamentos teóricos

2.1. Ruido branco

O ruído branco é um tipo de ruído produzido pela combinação simultânea de sons de todas as frequências. Este e denominado de branco em analogia ao funcionamento da luz branca, dado que esta é obtida por meio da combinação simultânea de todas as frequências cromáticas. Por conter sons de todas as frequências, o ruído branco é frequentemente empregado para mascarar outros sons.

2.2. Processos estocásticos

Um processo estocástico é utilizado para analisar sinais em que não é possível saber do seu conteúdo previamente, como o problema sugerido. No entanto sabe-se o comportamento que apresentará, dai, pode-se descrever os sinais estocásticos através de um modelo probabilístico.

Podemos então considerar sinais estocásticos como sinais aleatórios, cujas amostras são independentes de todas as outras, pelo que sempre que repetimos as experiências e registamos o seu output, os valores das sucessivas amostras serão de um modo geral diferentes das amostras obtidas nas experiências anteriores.

No caso do ruido é possível saber os limites a partir dos quais um segmento do sinal é efetivamente ruído ou contém informação relevante, pois este apresenta as seguintes características:

- Distribuição normal de media igual a zero;
- Variância constante;
- Covariância nula;
- Simetria;
- É não correlacionado entre diferentes estantes;

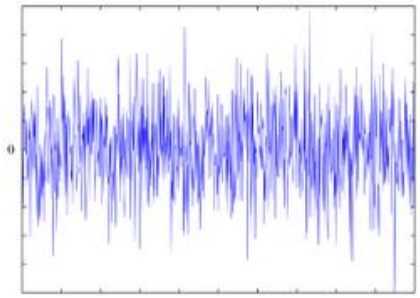


Figura 1 - Forma de onda do ruido branco

Para fazer a separação da informação relevante do ruido utiliza-se uma curva de *gauss,* verificando se a amostra se encontra dentro ou fora do "*outlier*" sendo fora informação útil e dentro ruido. A diferenciação entre ruido e informação útil tem em conta um *threshold* que indica a proporção mínima de amostras consideradas informação útil de um dado segmento do sinal.

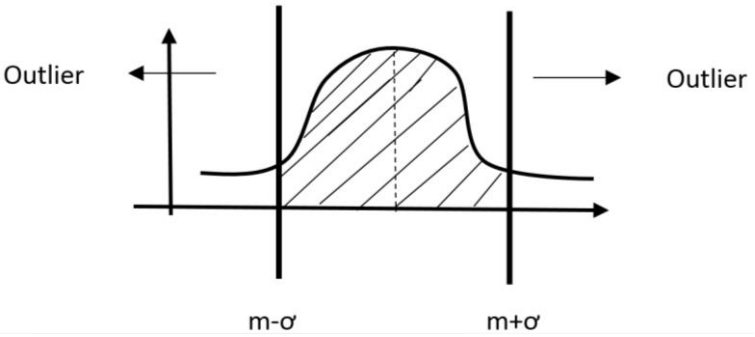


Figura 2 - modelo do ruido

Após a deteção e separação da informação útil do ruido pode-se calcular o SNR, *signal noise ratio,* fornecendo uma perspetiva da qualidade da amostra, este é nos dado por:

$$SNR = 20 \times \log(\frac{\mu amostra}{\mu ruido})$$

3. Procedimento

3.1. Metodologia

Para detetar e separar os segmentos de fala dos segmentos implementou-se o seguinte algoritmo:

- 1º fase – Amostrar o sinal para tal utilizou-se a função *wavrecord* que recebe como argumentos de entrada o numero de amostras e a frequência de amostragem e retorna o sinal amostrado.
- 2º fase – Calculou-se a media do sinal amostrado anteriormente através da função *mean,* a variância através da função *var* e o *threshold* através do produto de um dado tamanho da janela com uma dada proporção de informação útil.
- 3º fase – ciclo que percorre o sinal amostrado e verifica amostra a amostra se cada uma delas ruido ou fala e coloca num buffer de validação com o valor "1" se for fala ou a "0" se for ruido. A verificação se é ruido ou voz è efetuada através da comparação do módulo da diferença entre o valor da amostra e a média e dai se o resultado for maior que o desvio padrão é considerado como voz senão é considerado como ruido.

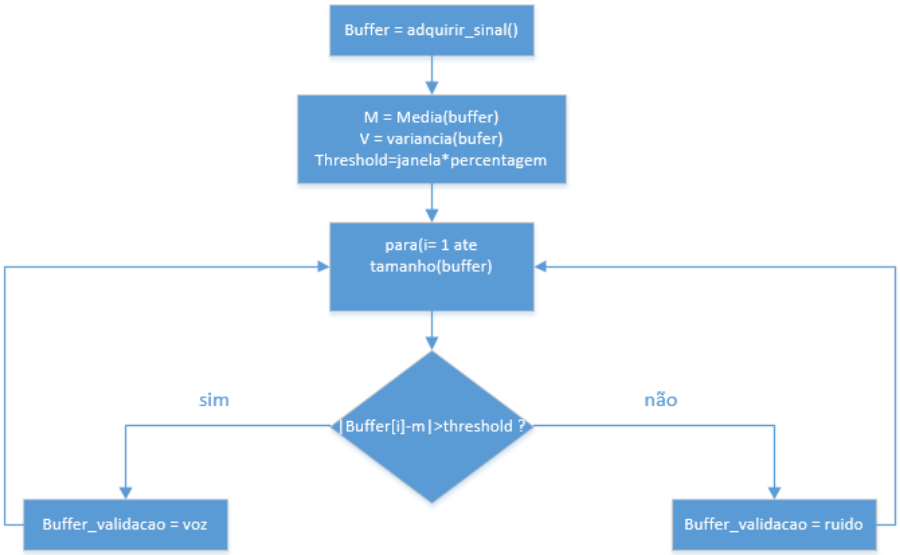


Figura 3- fluxograma do procedimento descrito anteriormente

- 4º fase – ciclo que percorre o buffer de validação anteriormente preenchido através de janelas fazendo o somatório dos valores presentes em cada janela, se o resultado deste somatório for superior ao *threshold* indica que essa janela

e de voz dai copia-se os valores correspondentes a essa janela do *buffer* do sinal amostrado para o *buffer* de voz senão copia-se para o *buffer* de ruido . Para se obter uma melhor precisão a janela corresponde a um segmento que contém a metade final da janela anterior, dai ser necessário verificar se a janela anterior foi para o mesmo *buffer* que a janela atual e adicionar a esse *buffer* de modo a não repetir informação.

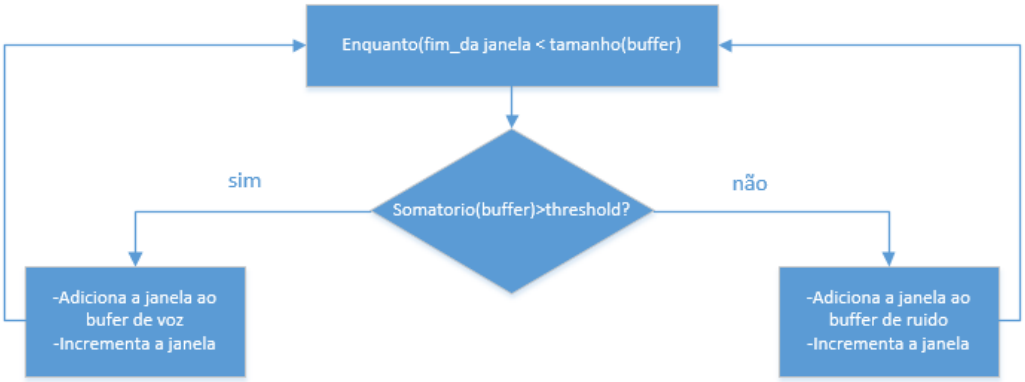


Figura 4 - fluxograma correspondente à fase acima descrita

3.2. Implementação

A implementação divide-se em duas fases, a fase da construção da função de deteção e remoção do silêncio da amostra e a fase da construção da interface gráfica.

Na primeira fase acima descrita fez-se a transcrição do algoritmo descrito anteriormente para tal foi necessário estudar as funções existentes no ambiente de desenvolvimento utilizado.

Na segunda fase necessitou-se de estudar o desenvolvimento de interface gráfica desde as ferramentas que este oferecia aos mecanismos por trás dessas ferramentas tal como *callbacks, handles* entre outras funções de manipulação de interface gráficas.

Nesta fase ainda se acrescentou mais funcionalidades ao sistema tal como adicionar ruido branco a uma amostra, guardar a amostra num ficheiro .wav e ouvir todas os componentes intervenientes.

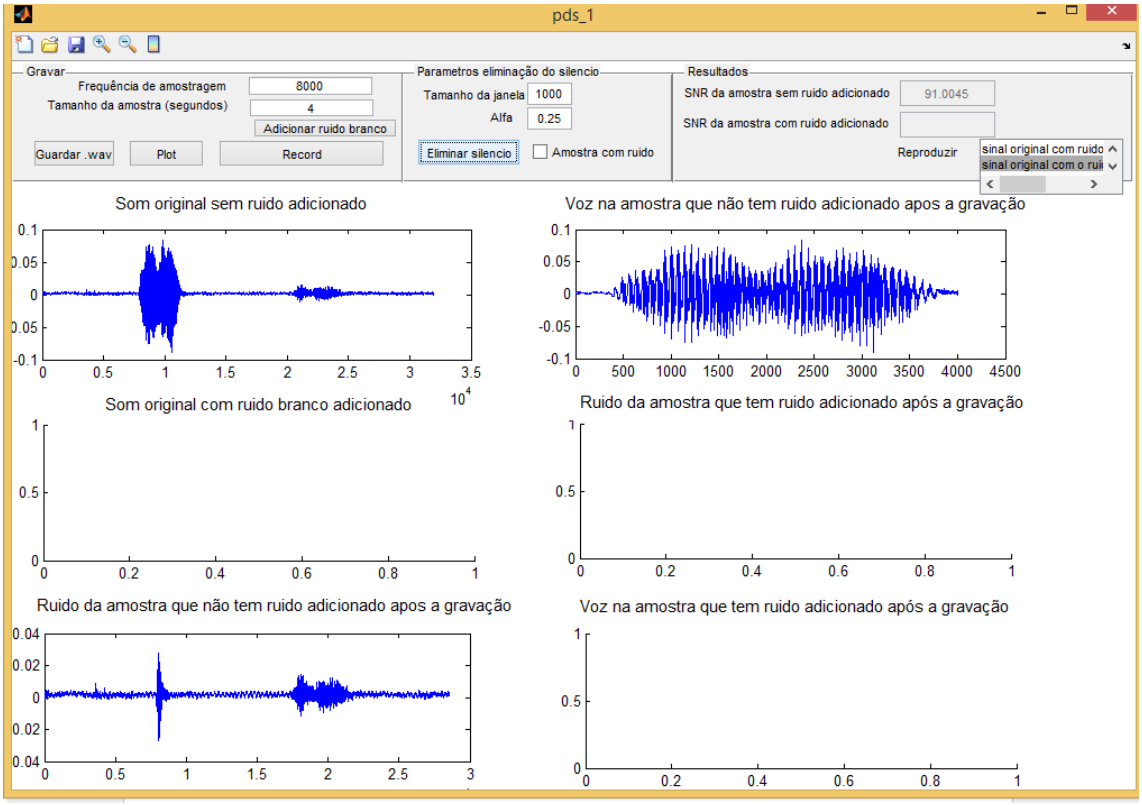


Figura 5 - layout da interface grafica

4. Analise dos resultados

Nesta parte do relatório irá se mostrar os resultados obtidos e a análise dos mesmos em várias situações e sob diferentes valores de *threshold*.

Parâmetros fixos para os testes em seguida:

- Frequência de amostragem = 8000 Hz;

- Duração da gravação de 4 s logo 32000 amostras.

- 1º teste – gravar um caso em que que acontece a seguinte situação:

Silêncio -> fala -> silêncio

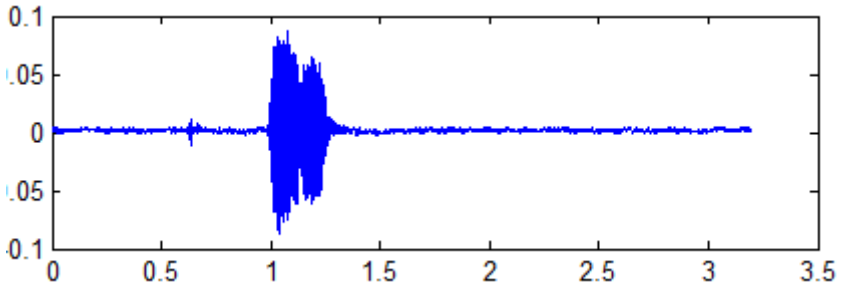


Figura 6 - som gravado nas condições acima referidas

Como se pode ver existe uma parte do sinal com uma amplitude claramente maior correspondendo esta à fala gravada.

- Para uma janela de 4000 amostras e uma proporção de “1” de 0.5 temos:

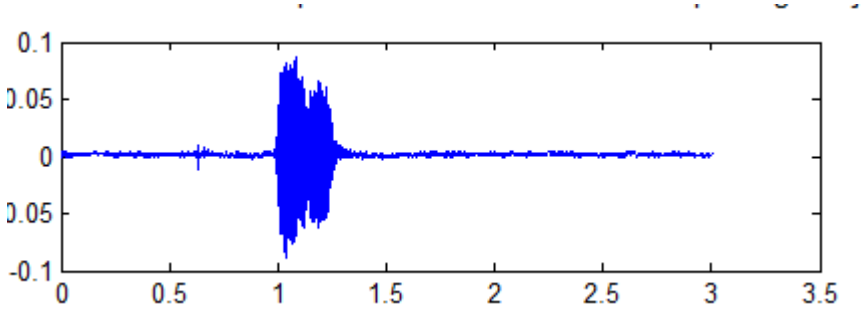


Figura 7 - ruido retirado da amostra para os parâmetros w= 4000 e fr = 0.5

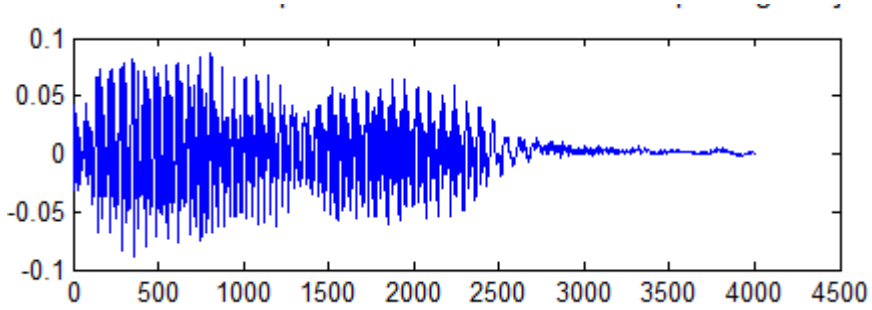


Figura 8 - sinal limpo

Neste caso grande parte da fala foi considerada ruido como se pode observar no gráfico da figura 6 e através da reprodução do sinal representado do mesmo que grande parte do sinal de informação útil foi considerado ruido, apesar de o sinal limpo também o conter informação útil, dai pode-se ver a vantagem de a janela conter sempre metade da janela anterior, ainda possui muito ruido principalmente na parte final do gráfico.

Obteve-se um SNR de 39.35 dB sendo considerado um valor baixo de qualidade de sinal.

- Para a mesma amostra uma janela de 500 e uma proporção de “1” de 0.5 temos:

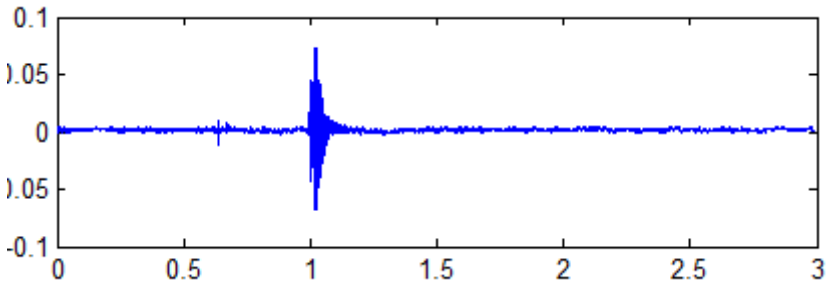


Figura 9 - ruido retirado para os parâmetros w = 500 e fr = 0.5

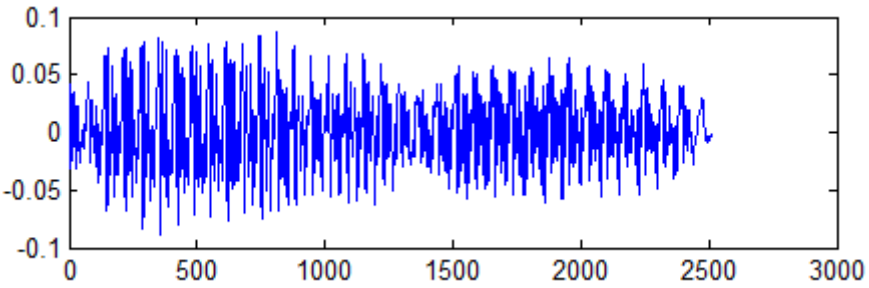


Figura 10 - sinal limpo para os parametros de w = 500 e fr = 0.5

Neste caso verifica-se que o ruido retirado contem muito menos informação útil que no caso anterior e que o sinal limpo resultando contem muito menos silêncio que o anterior resultando numa relação sinal limpo/ruido (SNR) de 81,87 dB apresentando uma qualidade de som bastante maior.

- Para a mesma amostra uma janela de 500 e uma proporção de "1" de 0.3 temos:

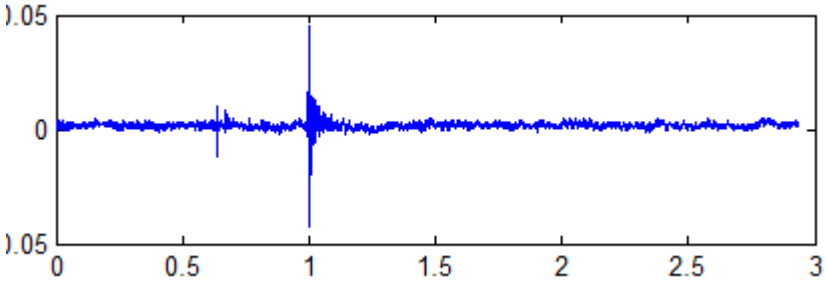


Figura 11 - ruido retirado da amostra com w = 500 e fr = 0.3

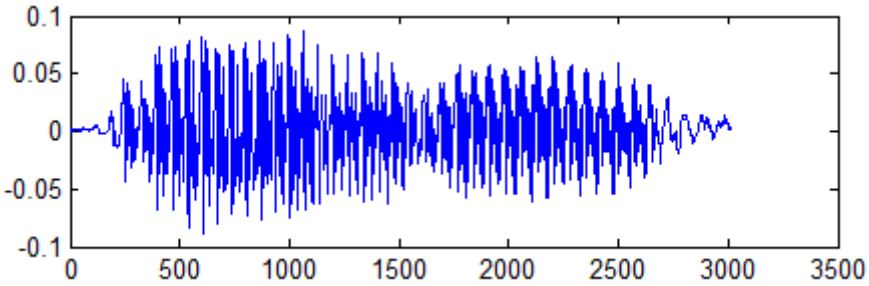


Figura 12 - sinal limpo com w = 500 e fr = 0.3

Neste caso ainda obtemos uma melhor qualidade de sinal, um SNR de 100.43 dB, caso se diminua mais a janela ou a proporção de "1" verifica-se que o SNR desce e o sinal ouvido perde informação dai conclui-se que estes valores serão o valor indicado para este caso.

- 2º teste - gravar um caso em que que acontece a seguinte situação:

  Fala -> silêncio -> fala

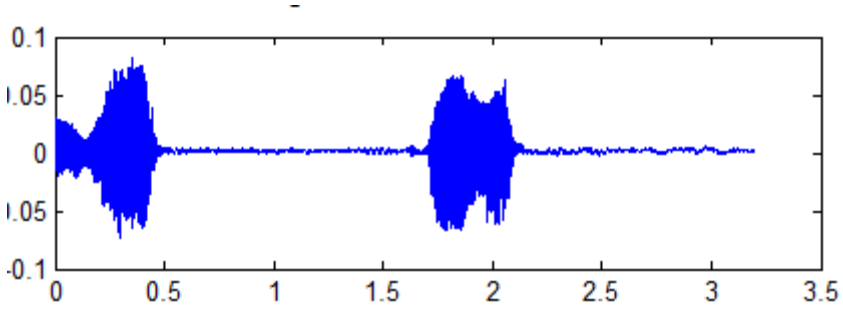


Figura 13 - sinal gravado para este teste

- Para uma janela de 4000 amostras e uma proporção de "1" de 0.5 temos:

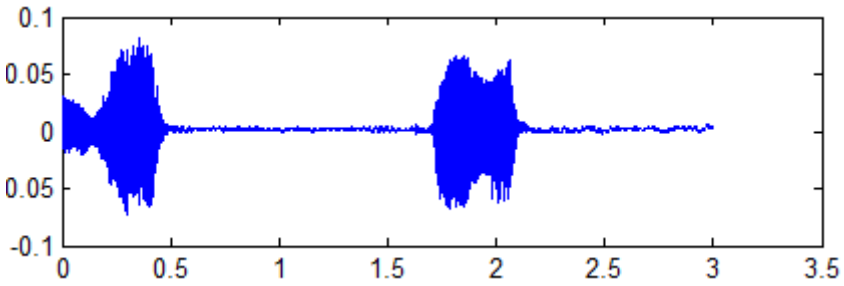


Figura 14 - ruido retirado da amostra para w =4000 e fr = 0.5

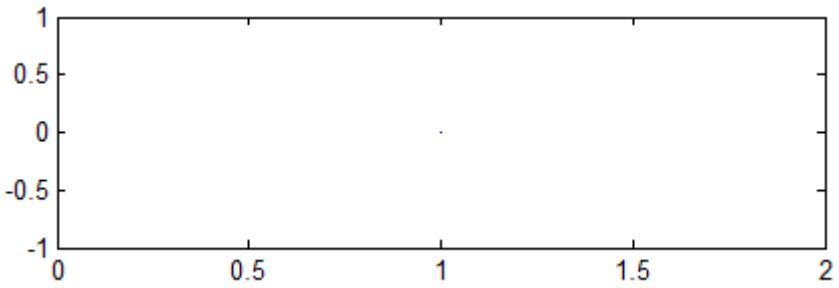


Figura 15 -sinal limpo para w =4000 e fr = 0.5

Verifica-se que para este caso o sistema considerou o sinal todo como ruido obtendo-se um SNR de menos infinito pois a media do sinal limpo dá zero.

- Para o mesmo sinal uma janela de 4000 amostras e uma proporção de "1" de 0.1 temos:

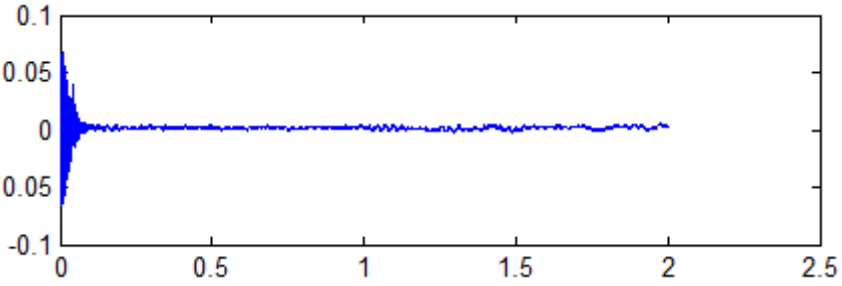


Figura 16 - ruido retirado nas condições acima referidas

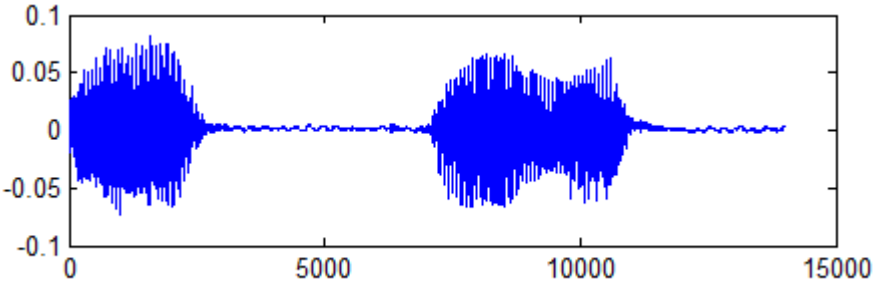


Figura 17 - sinal limpo nas condições acima referidas

Conseguiu-se obter um melhor resultado que no caso anterior, um SNR de 58.75 dB, mas parte da vocalização inicial foi considerada ruido e ainda aparece bastante silêncio entre os dois momentos de informação útil.

- Para o mesmo sinal uma janela de 1000 amostras e uma proporção de “1” de 0.25 temos:

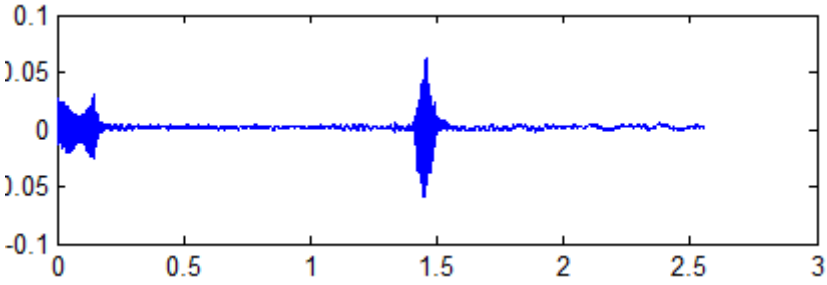


Figura 18 - ruido retirado nas condições acima referidas

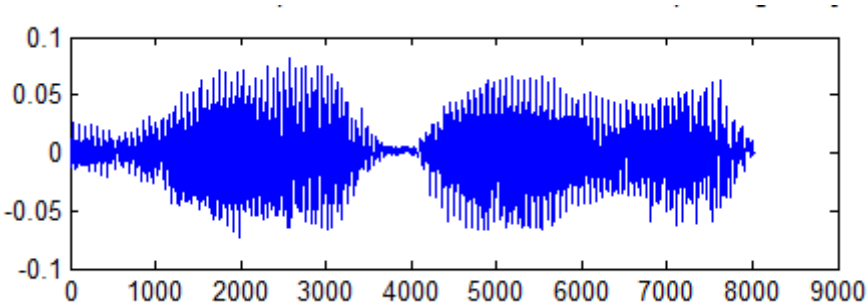


Figura 19 - sinal limpo nas condições acima referidas

Estes parâmetros proporcionaram um melhor resultado apesar de ainda se perder um pouco de informação no início o silêncio entre os momentos foi significatemente reduzido. Conseguiu-se um SNR de 65.38 dB.

Para esta amostra obteve-se resultados aceitáveis com uma janela de 300 amostras e uma proporção de "1" de 0.1 conseguindo-se um SNR de 88.152 dB.

- 2º teste - gravar um caso em que que acontece a seguinte situação:

  Fala -> silêncio -> murmúrio

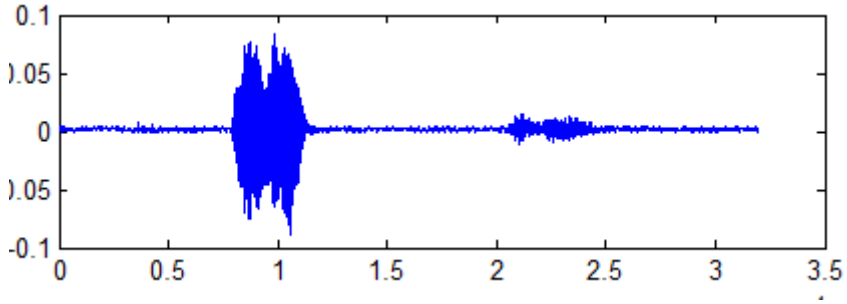


Figura 20 - sinal gravado para este teste

- Para uma janela de 1000 amostras e uma proporção de "1" de 0.25 temos:

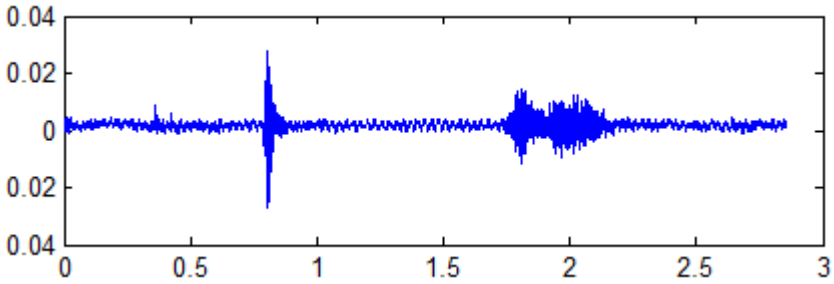


Figura 21 - ruido retirado nas condições acima referidas

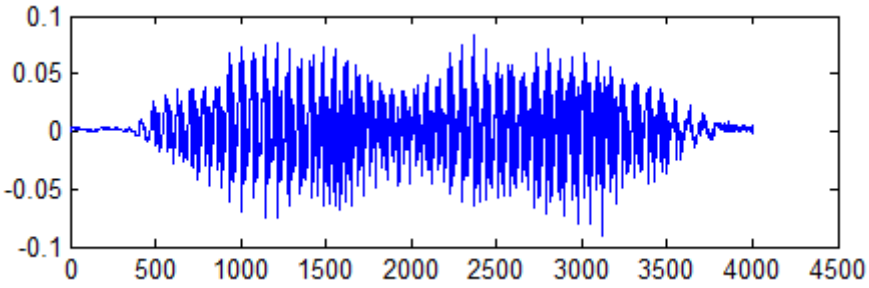


Figura 22 - sinal limpo nas condições acima referidas

Verificou-se que apesar de um SNR alto (cerca de 90 dB) a parte do murmúrio foi considerada integralmente ruido sendo apenas possível ouvir a parte da fala inicial no sinal limpo.

- Para uma janela de 500 amostras e uma proporção de “1” de 0.001 temos:

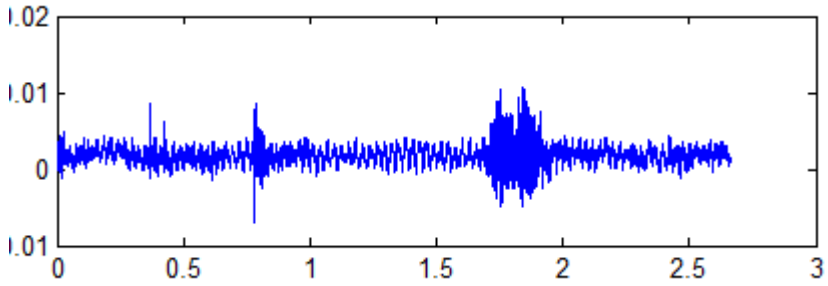
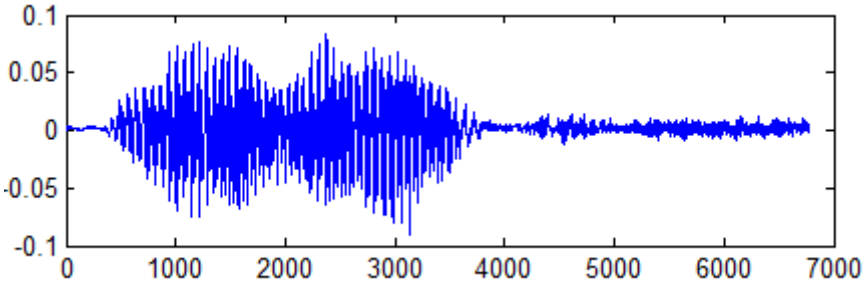


Figura 23 - ruido do sinal nas condições acima referidas



Neste caso, apesar de um SNR 2dB menor que no caso anterior consegue-se ouvir o murmúrio apesar que com uma intensidade menor que o sinal original.

5. Conclusão

Com a realização dos testes demonstrou-se que o sistema funciona otimamente conseguindo-se através da calibração dos valores referentes ao *threshold*, o tamanho da janela e a proporção de amostras com informação útil obter a eliminação dos momentos de silêncio numa gravação de um sinal de voz.

Conseguiu-se também obter fortes conhecimentos em processos estocásticos aplicados ao processamento digital de sinal tal como a utilizar a ferramenta de desenvolvimento e simulação *MATLAB*.

6. Bibliografia

Aulas e apontamentos de Processamento digital de sinal ano letivo 2012/2013

Aulas e apontamentos de Processamento digital de sinal ano letivo 2013/2014

http://en.wikipedia.org/wiki/Signal-to-noise_ratio

http://pt.wikipedia.org/wiki/Ru%C3%ADdo_branco

Documentação do MATLAB

ANEXO

Função de remoção do silêncio

```
        function [noise, clean, SNR] = noise_elimination(x, wlen,fr)

    m=mean(x(1:length(x)));      %calcular a media
    v=var(x(1:length(x)));       %calcular a variÃ¢ncia

    alfa=fr*wlen;                %calcular o threshold
    clean=0;                     %buffer para a fala
    noise=0;                     %buffer para o ruido
    a = 1:1:length(x);            %buffer para as validaÃ§Ãµes

    if(rem(wlen,2)==1)           %ver se a janela e par senÃ£o Ã©
adicionar 1 ao tamanho da janela
         wlen=wlen+1;
    end

    % o ciclo seguinte Ã© para ver se cada posiÃ§ao do sinal amostrado
e fala
    % ou ruido se o modulo da soma dessa posiÃ§ao com media for maior
que o desvio padrao
    % e fala logo o  buffer de validdacao ficara com valor 1 nessa
posicao senao e ruido e ficara com o valor 0

    for i=1: length(x)
        if(abs(x(i)-m)>sqrt(v))
            a(i)=1;
        else
            a(i)=0;
        end
    end

     %iniciar a janela na posicao 1 atÃ© ao tamanho dado pelo
argumento de entrada wlen

    initw=1;
    endw=initw+wlen;
    prev_noise=0;  %inicializar a variavel correspondente ao estado
anterior(se ruido ou se fala) a 0

    while(endw<length(x))
        w=a(initw:endw);         %copiar os dados do buffer de
validacao correspondente a janela atual para um buffer auxiliar

        if(sum(w)>alfa)           %caso o somatorio dos valores do
buffer auxiliar seja maior que o threshold colocar os dados
correspondentes ao buffer do sinal no buffer de fala
            if(prev_noise==1)
                clean=cat(1, clean, x(initw:endw)); %caso o somatorio
dos valores do buffer auxiliar seja maior que o threshold colocar os
dados correspondentes ao buffer
```

```
                                                    % do sinal no
buffer de fala
            else
                clean=cat(1, clean, x(initw+wlen/2:endw)); % caso a
janela anterior tenha sido no buffer de fala colocar neste buffer
                                                             % coloca
os dados correspondentes so Ã  ultima metade da janela
            end                                              %atualiza
a variavel estado anterior para o estado atual
            prev_noise = 0;
        else
            if(prev_noise==1)
                noise=cat(1, noise, x(initw+wlen/2:endw));

            else
                noise=cat(1, noise, x(initw:endw));

            end
            prev_noise = 1;
        end

        initw=initw+wlen/2;  %define a proxima janela
        endw=endw+wlen/2;

        if(endw>length(x))      % condicÃ£o caso o numero de janelas
nÂ°ao de um numero inteiro
            endw=length(x);
            initw=endw-initw;
        end
    end

    %subplot(3,1,1), plot(x);
    %subplot(3,1,2), plot(clean);
    %subplot(3,1,3), plot(noise);

    clean_mean=mean(clean.^2);     % calcular a media do buffer de fala
    noise_mean=mean(noise.^2);     % calcular a media do buffer de
ruido
    SNR=20*log(clean_mean/noise_mean)     %calcular o SNR
```

Interface gráfica

```
function varargout = pds_1(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pds_1_OpeningFcn, ...
                   'gui_OutputFcn',  @pds_1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pds_1 is made visible.
function pds_1_OpeningFcn(hObject, ~, handles, varargin)
global soundx;
global sound_noise;
global noise;
global clean;
global aux;
global clean_r;
% Choose default command line output for pds_1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

varargout{1} = handles.output;


% --- Executes on button press in guardar.
function guardar_Callback(hObject, ~, handles)
fs = str2num(get(handles.fs,'string'));
disp(fs)
wavwrite(soundx,fs,8,'C:\Users\NUNO\Documents\pds-
trabalho\original_sound.wav');


function fs_Callback(hObject, ~, handles)
fs=str2double(get(hObject,'String'));
handles.metricdata.fs = fs;
guidata(hObject,handles)


function segundos_Callback(hObject, eventdata, handles)
seconds=str2double(get(hObject,'String'));
handles.metricdata.seconds = seconds;
guidata(hObject,handles)
```

```
% --- Executes on button press in record.
function record_Callback(hObject, eventdata, handles)
fs = str2num(get(handles.fs,'string'));
disp(fs)
sec =str2num(get(handles.segundos,'string'));
disp(sec)
global soundx;
soundx = wavrecord(sec*fs,fs);
sound(soundx);
guidata(hObject,handles)

% --- Executes on button press in Grafico_record.
function Grafico_record_Callback(hObject, ~, handles)
global soundx;
axes(handles.Original_record);
plot(soundx);


% --- Executes on button press in add_noise.
function add_noise_Callback(hObject, ~, handles)
global soundx;
global sound_noise;
global aux;
sound_noise = awgn(soundx,60);
aux = 1;
sound(sound_noise);
axes(handles.added_noise);
plot(sound_noise);
% Hint: get(hObject,'Value') returns toggle state of add_noise



function janela_Callback(hObject, eventdata, handles)
global soundx;
wlen=str2double(get(hObject,'String'));
if(wlen > (length(soundx)/2))
    errordlg('A janela tem de ser menor que metade do tamanho da
amostra','Error');
else
handles.metricdata.janela = wlen;
guidata(hObject,handles)
end

function alfa_Callback(hObject, eventdata, handles)
threshold=str2double(get(hObject,'String'));
if(threshold > 1)
    errordlg('O alfa tem de ser menor que 1','error');
else
handles.metricdata.threshold = threshold;
guidata(hObject,handles)
end

% --- Executes on button press in select_amostra_ruido.
function select_amostra_ruido_Callback(hObject, eventdata, handles)
select=get(hObject,'value');
handles.metricadata.select = select;
guidata(hObject,handles)

% --- Executes on button press in silence_elimination.
function silence_elimination_Callback(hObject, eventdata, handles)
```

```
global noise;
global clean;
global soundx;
global sound_noise;
global clean_r;
SNR = 0;
SNR_r = 0;

wlen = str2num(get(handles.janela,'string'));
threshold =str2num(get(handles.alfa,'string'));
select = get(handles.select_amostra_ruido,'value');

if(select == 0)
    [noise, clean, SNR] = noise_elimination(soundx, wlen,threshold);
    axes(handles.ruido);
    plot(noise);
    axes(handles.voz);
    plot(clean);
    sound(clean);
    set(handles.snr,'string',num2str(SNR));
end

if(select == 1)
    [noise_r, clean_r, SNR_r] = noise_elimination(sound_noise,
wlen,threshold);
    axes(handles.ruido_added_noise);
    plot(noise_r);
    axes(handles.voz_added_noise);
    plot(clean_r);
    sound(clean_r);
    set(handles.snr_R,'string',num2str(SNR_r));
end


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)

global clean;
global soundx;
global sound_noise;
global clean_r;
s =get(hObject,'value');

switch s
    case(1)
        sound(soundx);
    case(2)
        sound(sound_noise);
    case(3)
        sound(clean);
    case(4)
        sound(clean_r);
end
```

Layout interface gráfico