Universidade do Minho-Escola de Engenharia

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores.

Deteção e Remoção de Intervalos de Silêncio na Fala

Projeto no âmbito da UC de Processamento Digital de Sinal

Docente: Carlos Lima

Sofia Paiva A78838

2017/2018

Índice

Introdução

No mundo digital é muito comum a gravação de áudio, nomeadamente da nossa fala. No entanto, num segmento de voz gravada existem segmentos de silêncio, que mesmo não contendo informação linguística relevante, contêm dados (de ruído branco) que será também enviado. Tal informação é desnecessária de ser processada e enviada, pelo que se concebeu um algoritmo que permite analisar todo um segmento de voz gravada e localizar os segmentos de silêncio para posterior eliminação desse conteúdo "ruidoso".

Para se conseguir tal algoritmo, baseou-se nos gráficos de controlo de Shewhart, aplicando-se limites (*threshold*) variáveis para que se possam adaptar às condições do ruído.

Inicialmente efetuou-se um estudo sobre o melhor *threshold* a aplicar para cada relação sinal-ruído (SNR) diferente. Testando com valores de SNR a escalar de 0 dB até 50 dB com intervalos de 10 dB e aplicando então valores de *threshold* entre os 0.2 e 16, para se obter uma tabela que permite decidir qual o melhor *threshold* a utilizar consoante diferentes valores de SNR.

A partir desta tabela é então possível que o algoritmo consiga, através do cálculo automático do SNR, escolher o valor de *threshold* mais adequado para proceder à deteção e remoção de segmentos de silêncio na fala.

SNR e Ruído Branco

O SNR (*Signal to Noise Ratio*) é uma medida que compara os níveis de um sinal com o do ruído de fundo que este possa conter. Este pode ser definido em decibéis, em que 0 dB indica que o nível do ruído se equipara (sobrepõe-se) ao do sinal e quanto maior for, menor influência o ruído tem no sinal analisado. O SNR é definido como sendo:

$$SNR = \frac{P_{sinal}}{P_{ruído}} = \left(\frac{A_{sinal}}{A_{ruído}}\right)^2 \quad (1)$$

Onde *P* é a potência média e *A* é a amplitude eficaz dos sinais.

Para o cálculo em dB, poder-se-á usar a mesma expressão aplicando-se a base dos decibéis, isto é:

$$SNR_{dB} = 10\log_{10}\left[\left(\frac{A_{sinal}}{A_{ruído}}\right)^2\right] \quad (2)$$

Foi com base nesta expressão que se chegou ao cálculo da potência desejada (neste caso da amplitude desejada) do sinal de ruído gerado para testes (“Signal-to-noise ratio,” n.d.).

As características deste ruído são as típicas de um ruído branco, que não é mais que um sinal aleatório com igual intensidade em todas as frequências do espectro, traduzindo-se numa densidade espectral de potência constante (Mancini, Carter, & Texas Instruments Incorporated., 2009). Em tempo amostrado, este ruído, que será gerado com uma amplitude que cumpra o SNR desejado, será constituído por um vetor de pontos aleatórios com distribuição normal e comprimento igual ao de amostras do sinal.

Controlo de Shewhart e Distância Normalizada

Walter A. Shewhart desenvolveu os denominados *control charts*, que permitem verificar se um processo se encontra sob controlo ou não, através da estatística que os seus dados apresentem. Para se chegar a esses *control charts* são necessárias amostras, a média dessas amostras (μ), que indica o centro do gráfico, e o desvio padrão dessas amostras (σ), que é depois usado para definir os limites de controlo, fixos por Shewhart em três vezes o desvio padrão calculado (3σ) a partir da linha central do gráfico(Tague, 2004). Caso uma amostra ultrapassasse qualquer um desses limites, o processo seria considerado como não controlado. ("Control Charts - CQE Academy," n.d.)

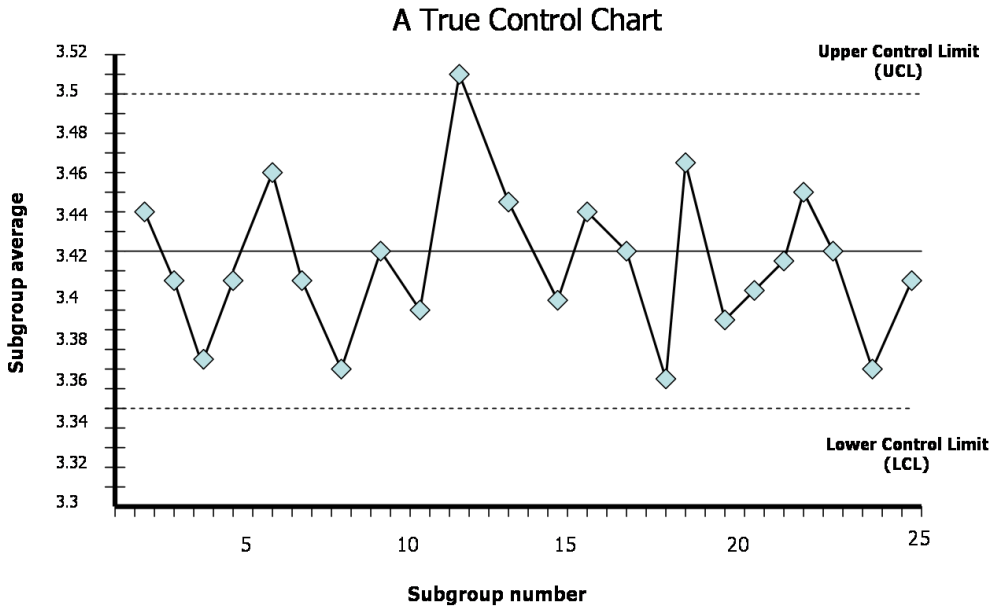

Figura 1 - Exemplo de um control chart

Uma maneira intuitiva de verificar se uma amostra deve pertencer ou não a um certo conjunto onde surja inserida é através da distância normalizada, isto é:

$$\frac{|x-\mu|}{\sigma} \tag{3}$$

Sendo *x* o valor dessa amostra. Desta maneira entende-se que quanto mais perto o ponto estiver da média das amostras tendo em conta a dimensão do conjunto (desvio padrão), maior a probabilidade de essa amostra pertencer ao conjunto em estudo (Mahalanobis, 1936). É necessário então definir o limite que justifica a pertença ou não do ponto (*threshold*).

Algoritmo de deteção de silêncios

Com base nos *control charts* de Shewhart e na distância normalizada, o algoritmo implementado (baseado em ("GT's Blog: Silence Removal and End Point Detection MATLAB Code," n.d.)) consiste na divisão do sinal em *frames*, para identificação de quais seriam vozeadas e quais seriam ruído com base numa primeira recolha de amostras consideradas ruído.

```
bgSampleCount = floor(Fs/5); %a primeira quinta parte do sinal e
ruído

%calculo da media e do desvio padrao do ruido
bgSample=[];
for i=1:1:bgSampleCount
    bgSample = [bgSample sinal(i)];
end
media=mean(bgSample);
desvio=std(bgSample);
```

Estes serão os parâmetros usados para definir o grupo onde as amostras devem pertencer, o limite ao qual será comparado será previamente definido para se obter resultados flexíveis consoante o SNR que o sinal tiver.

```
%identificar partes vozeadas para cada valor
for i=1:1:length(sinal)
    if(abs(sinal(i)-media)/desvio > threshold) %se o valor da
amostra menos a media do ruido, a dividir pelo desvio do ruido
for maior que o threshold (indica que não pertence ao conjunto
do ruido)
        voiced(i)=1; %guarda em voiced indicacao de quais
amostras no sinal sao vozeadas
    else
        voiced(i)=0;
    end
end
```

Após a verificação ponto a ponto, poder-se-á verificar segmento a segmento, se contiver mais amostras consideradas vozeadas que não vozeadas, este será mantido no sinal, caso contrário será considerado ruído (silêncio).

```
%identificar as partes vozeadas por frame
amostrasUteis = length(sinal)-
mod(length(sinal),amostrasPorFrame);
%quantos frames tem de percorrer
frameCount = amostrasUteis/amostrasPorFrame;
voicedFrameCount = 0;

for i=1:1:frameCount %percorrer todos os frames
    cVoiced=0;
    cUnvoiced=0;

    for j=i*amostrasPorFrame-amostrasPorFrame +1:1:
(i*amostrasPorFrame) %percorre todas as amostras da frame
        if(voiced(j)==1)
            %conta as amostras que sao vozeadas
            cVoiced = (cVoiced+1);
```

```
            else
                %conta as amostras que nao sao vozeadas
                cUnvoiced = cUnvoiced +1;
            end
        end

          %a frame contem mais amostras vozeadas que nao vozeadas
          if(cVoiced > cUnvoiced)
            %conta as frames vozeadas
            voicedFrameCount = voicedFrameCount+1;
            voicedUnvoiced(i)=1; %indica que e uma frame vozeada
        else
            voicedUnvoiced(i)=0; %indica que e uma frame nao vozeada
        end
    end

    sinal_sem_ruido=[];
    ruido=[];

    for i=1:1:frameCount
        for j=i*amostrasPorFrame-amostrasPorFrame+1:1:
    (i*amostrasPorFrame)
            if(voicedUnvoiced(i)==1) %se e uma frame vozeada
                sinal_sem_ruido = [sinal_sem_ruido sinal(j)];
            else
                ruido = [ruido sinal(j)];
            end
        end
    end
```

Método de obtenção do *threshold*

Para se obter os melhores valores para o *threshold* mencionado acima, foram efetuados vários testes com base num mesmo sinal gravado ao qual se somou ruído de amplitude tal que provocaria um SNR escolhido.

```
S=mean(recorded.^2); %potencia do sinal

noise = randn(1,length(recorded)); %ruído branco com
distribuicao normal
noise = noise - mean(noise); %ruído branco com media 0
noise = noise * sqrt(S/10^(SNR/10)); %ruido branco com potencia
pretendida - vem da formula de SNR
sinal=recorded+noise'; %contaminacao do sinal recebido
```

Pode-se observar que o ruído é obtido a partir da manipulação da equação (2) de modo a retirar-se a amplitude eficaz que os valores de ruído devem tomar.

$$A_{ruído}^2 = \frac{A_{sinal}^2}{10^{\left(\frac{SNR_{dB}}{10}\right)}} \tag{4}$$

$$<=> A_{ruído} = \sqrt{\frac{P_{sinal}}{10^{\left(\frac{SNR_{dB}}{10}\right)}}} \tag{5}$$

- **Resultados**

Definindo a frequência de amostragem como 11025 Hz por ser a mais utilizada para análise de áudio e gravando a 16 bits para se obter melhor qualidade de gravação, gravou-se um sinal de conteúdo "…1…2…3…4…" e com as seguintes características:

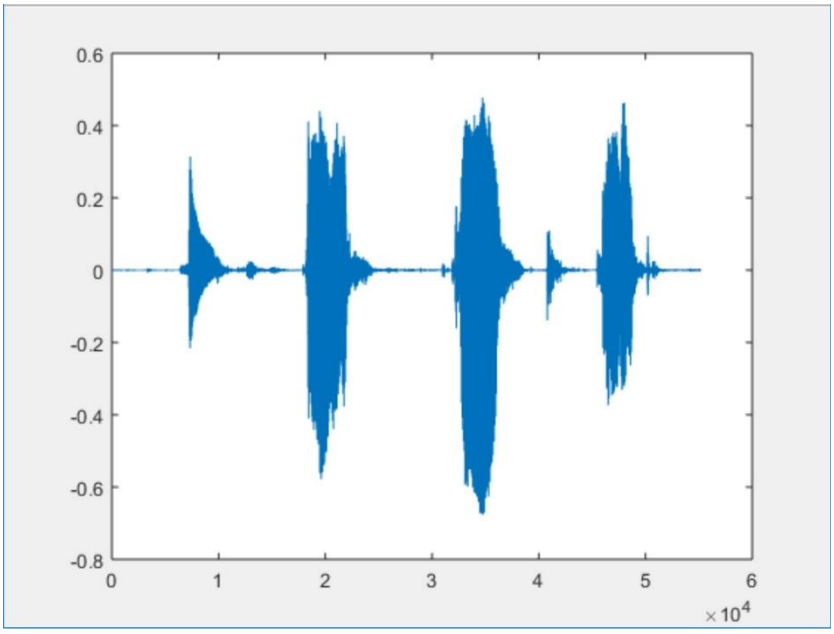

Figura 2- Áudio original

Sobre o qual se vai somar ruído para se encontrar o melhor *threshold* por tentativa e erro, certificado pela audição dos resultados e análise dos gráficos.

- SNR de 0 dB
  O sinal gravado mais ruído pode observar-se pela figura 3

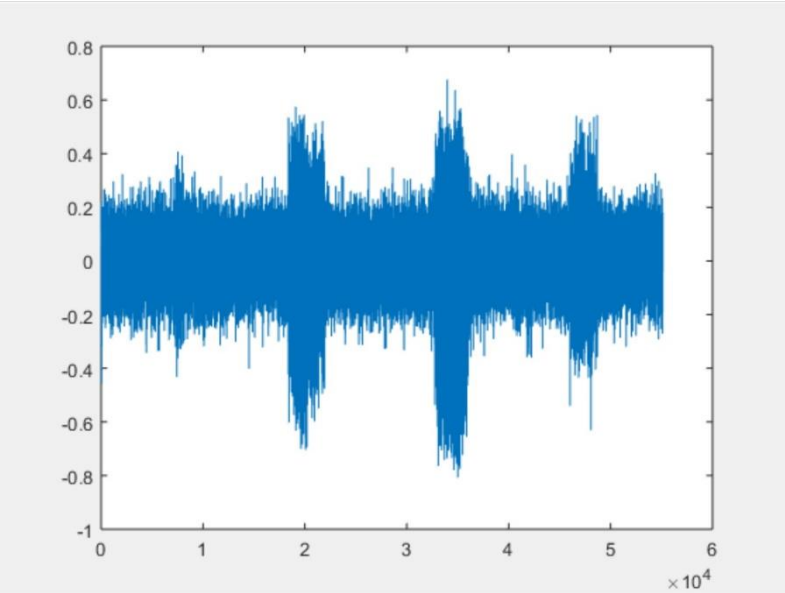

Figura 3- Áudio com SNR de 0 dB

Ao fim de algumas tentativas, começando com um *threshold* de 0.2, concluiu-se que o melhor valor seria de 0.75 resultando no seguinte sinal sem os silêncios:

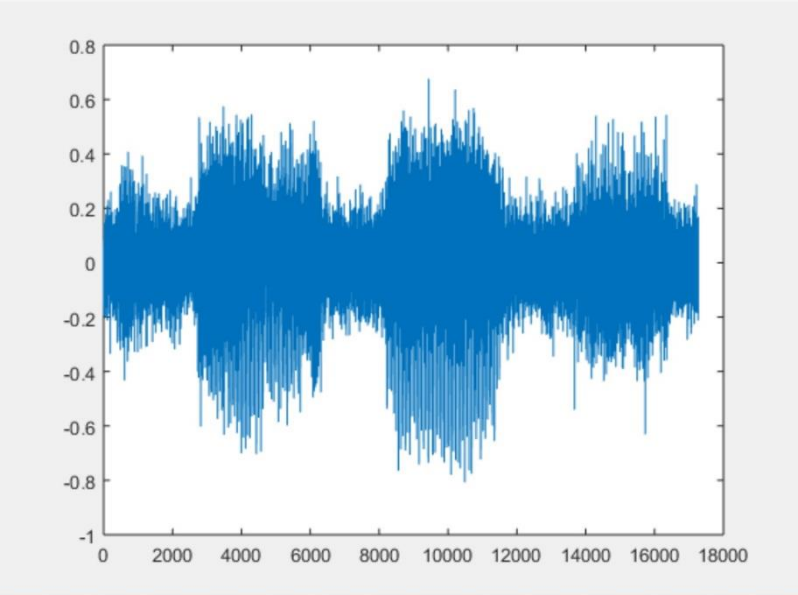

Figura 4- Áudio sem silêncios e com SNR de 0 dB

- SNR de 10 dB
  O sinal gravado mais ruído pode observar-se pela figura 5

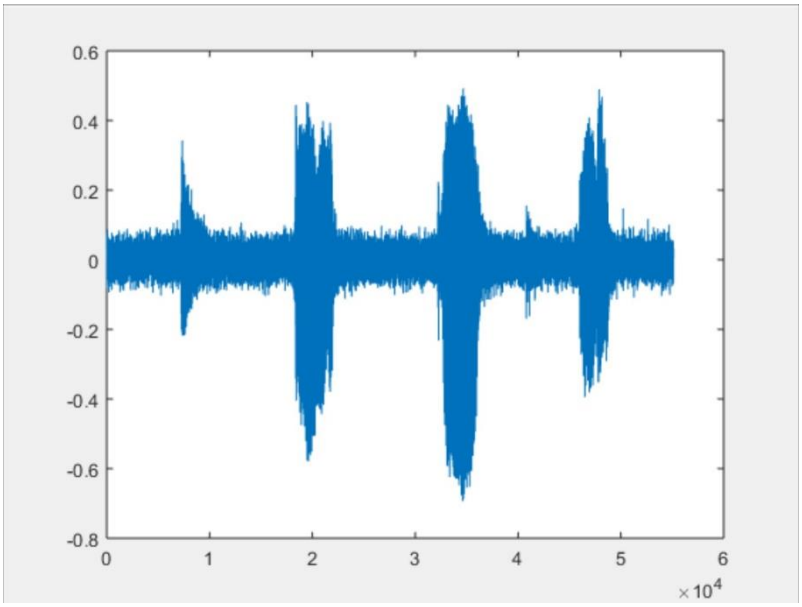

Figura 5- Áudio com SNR de 10 dB

Partindo do valor de *threshold* anterior e mais uma vez por tentativa e erro, conclui-se que o melhor valor para um SNR de 10 dB seria de 0.85, resultando no seguinte sinal sem os silêncios:

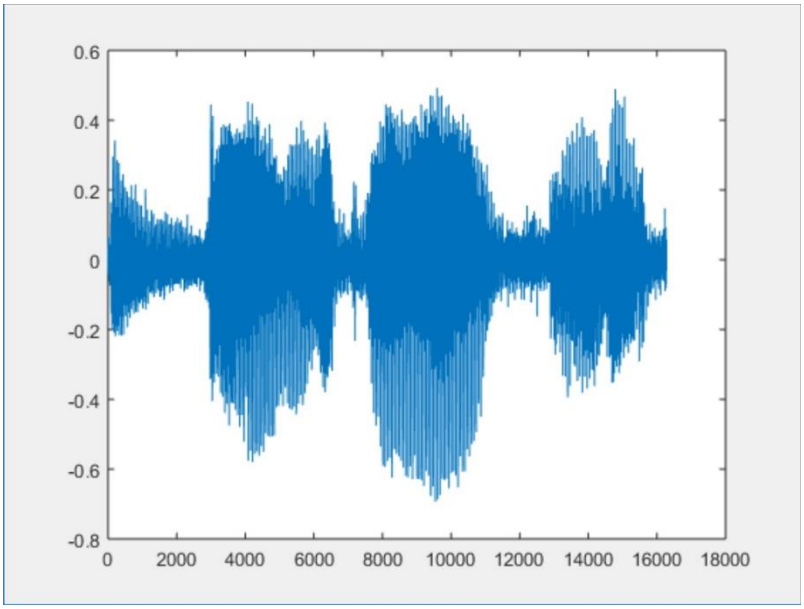

Figura 6- Áudio sem silêncios e com SNR de 10 dB

- SNR de 20 dB
  O sinal gravado mais ruído pode observar-se pela figura 7

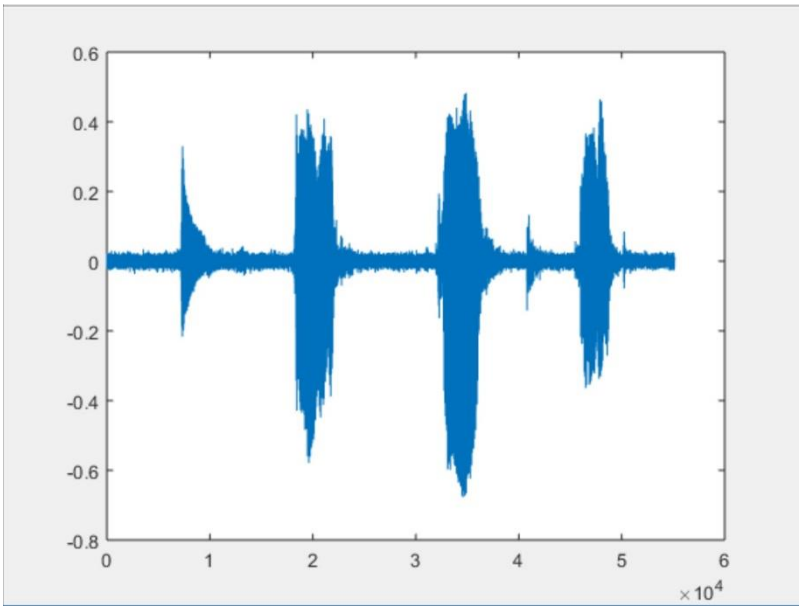

Figura 7- Áudio com SNR de 20 dB

De notar que quanto maior o SNR, menos impacto o ruído tem no sinal. Mais uma vez partindo do valor de *threshold* anterior e procurando sempre o melhor valor, conclui-se que para um SNR de 20 dB este seria de 1.5, resultando no seguinte sinal sem os silêncios:

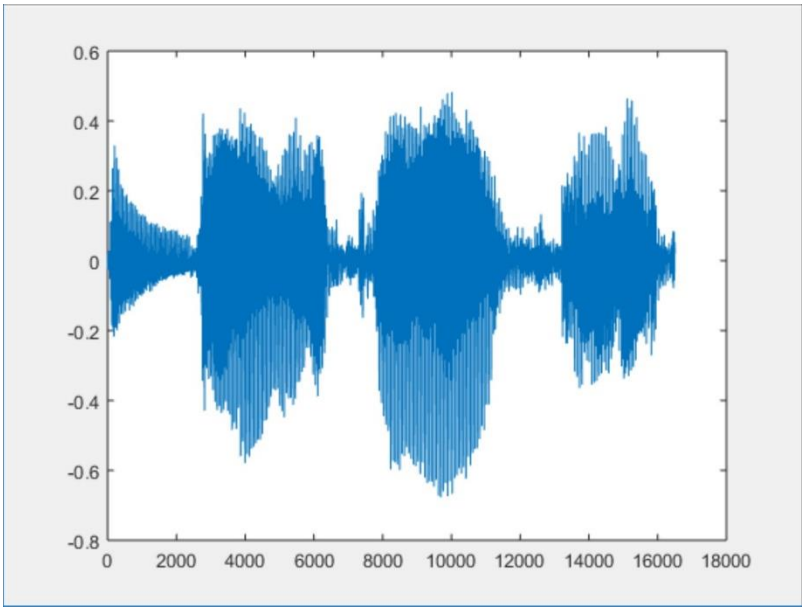

Figura 8- Áudio sem silêncios e com SNR de 20 dB

- SNR de 30 dB
  O sinal gravado mais ruído pode observar-se pela figura 9

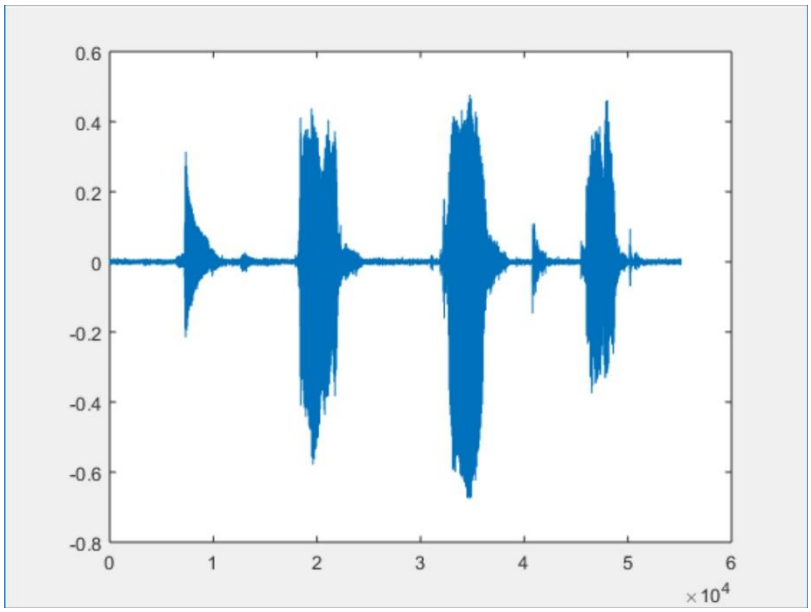

Figura 9- Áudio com SNR de 30 dB

Partindo do valor de *threshold* anterior e mais uma vez por tentativa e erro, conclui-se que o melhor valor para um SNR de 30 dB seria de 2.8, resultando no seguinte sinal sem os silêncios:

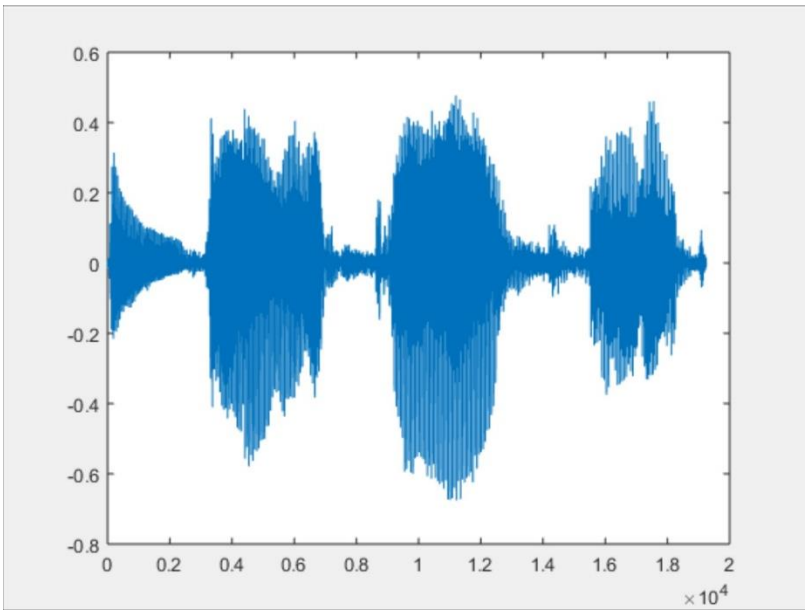

Figura 10- Áudio sem silêncios e com SNR de 30 dB

- SNR de 40 dB
  O sinal gravado mais ruído pode observar-se pela figura 11

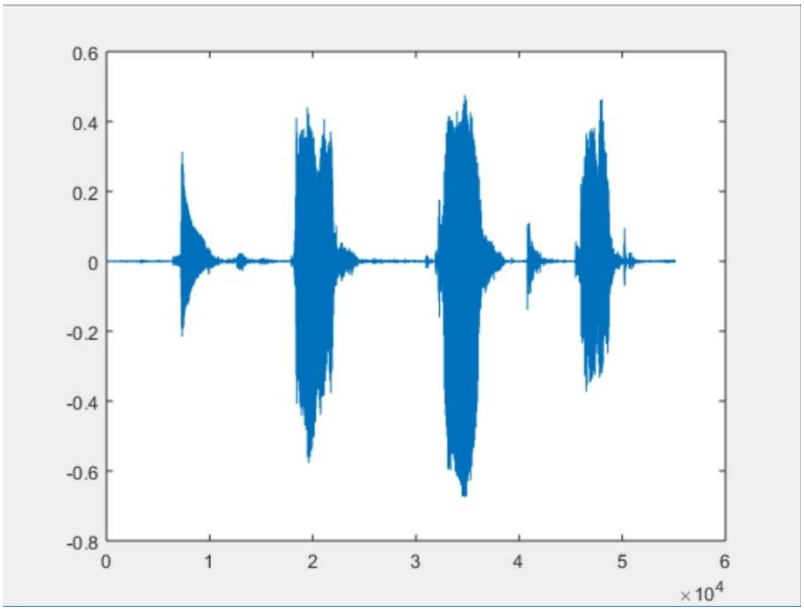

Figura 11- Áudio com SNR de 40 dB

Repare-se que o sinal de ruído adicionado já é quase impercetível no sinal, parecendo-se cada vez mais ao sinal original. Este facto implicou uma subida acentuada do melhor valor de *threshold*, concluindo-se que para um SNR de 40 dB seria de 8, ao fim de algumas tentativas, resultando no seguinte sinal sem os silêncios:

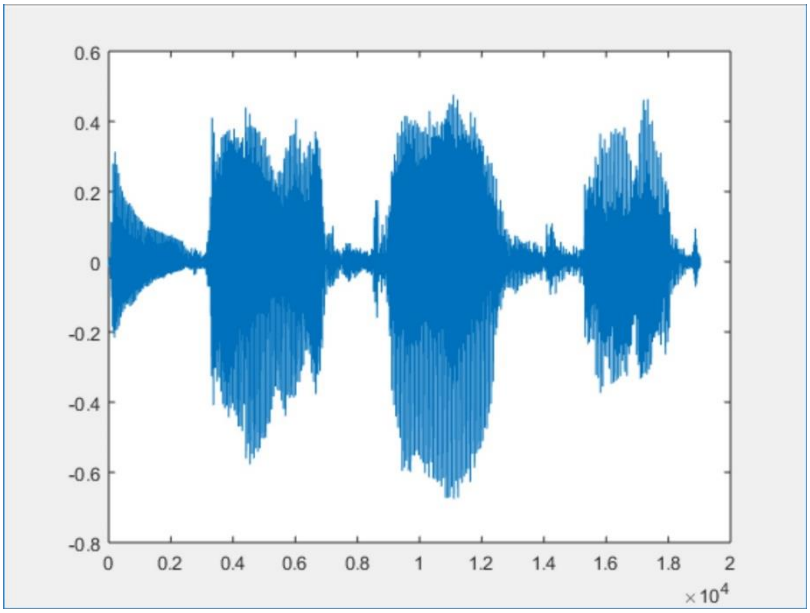

Figura 12- Áudio sem silêncios e com SNR de 40 dB

- SNR de 50 dB
  O sinal gravado mais ruído pode observar-se pela figura 13

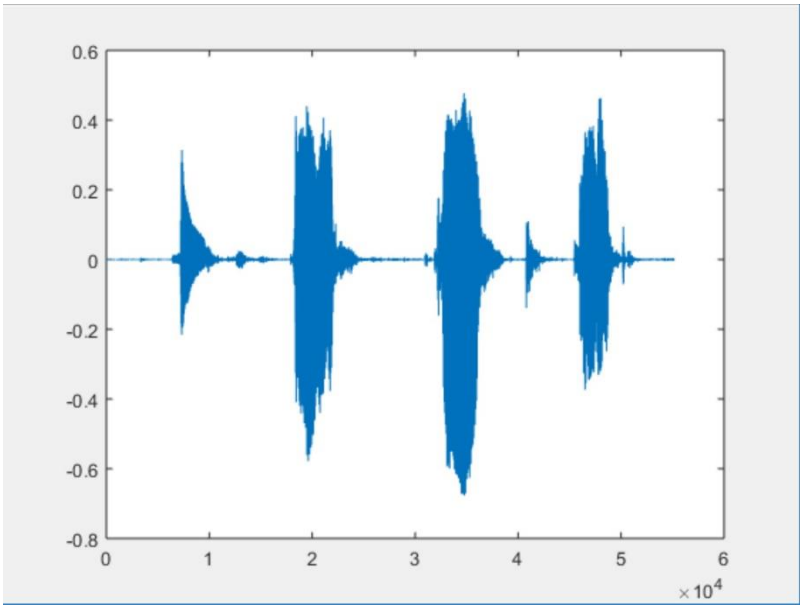

Figura 13- Áudio com SNR de 50 dB

A partir dos 50 dB, a influência do ruído é tão mínima, que já nem se nota diferenças com o sinal original, mais uma vez isso fez escalar o valor de *threshold* que coloca o gráfico do sinal sem os silêncios o mais parecido possível aos restantes já obtidos, concluindo-se que para tal o melhor valor seria de 16 e sendo o referido gráfico o seguinte:

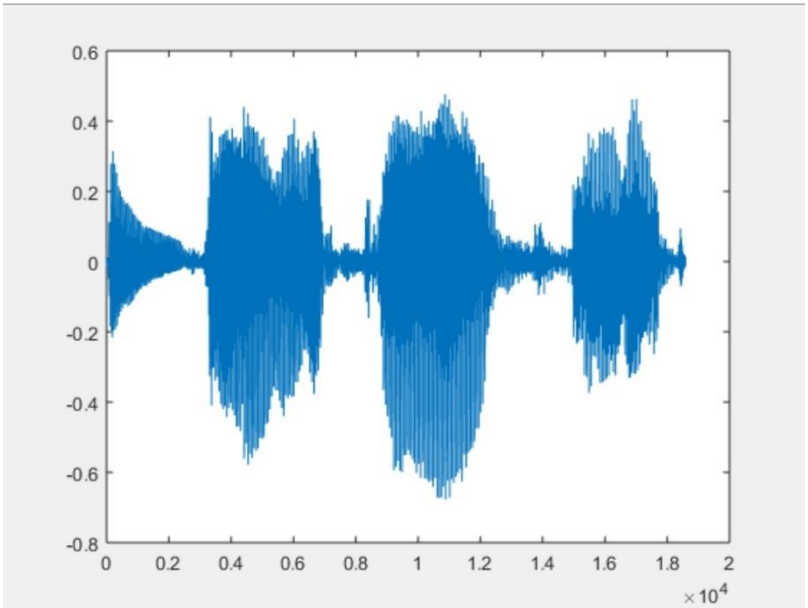

Figura 14- Áudio sem silêncios e com SNR de 50 dB

Deteção automática de SNR e ajuste de *threshold*

Depois de se obter a seguinte tabela de resultados:

Tabela 1 - Relação SNR - Threshold

SNR (dB)	*Threshold*
0	0.75
10	0.85
20	1.5
30	2.8
40	8
50	16

Pode-se então executar o algoritmo de forma automática, que, para cada sinal que recebe, calcula o SNR com base no ruído das primeiras amostras e vai buscar o valor de *threshold* mais adequado.

```
potencia_sinal = mean(sinal.^2); %potencia do sinal
potencia_ruido = mean(sinal(1:(Fs/5)).^2); %potencia do ruido,
considerando que a primeira quinta parte do sinal é ruído

SNR = 10*log10(potencia_sinal/potencia_ruido); %calculo do SNR
pela formula

if (SNR >= 0 && SNR<10)
      %Calculo da reta do intervalo de 0 a 10
      threshold = SNR*((0.85-0.75)/10)+0.75;
elseif (SNR >= 10 && SNR <20)
      %Calculo da reta do intervalo de 10 a 20
      threshold = SNR*((1.5-0.85)/10)+(0.85-((1.5-0.85)/10)*10);
elseif (SNR >=20 && SNR <30)
      %Calculo da reta do intervalo de 20 a 30
      threshold = SNR*((2.8-1.5)/10)+(1.5-((2.8-1.5)/10)*20);
elseif (SNR >=30 && SNR <40)
      %Calculo da reta do intervalo de 30 a 40
      threshold = SNR*((8-2.8)/10)+(2.8-((8-2.8)/10)*30);
elseif (SNR >=40 && SNR <50)
      %Calculo da reta do intervalo de 40 a 50
      threshold = SNR*((16-8)/10)+(8-((16-8)/10)*40);
elseif (SNR >=50)
      threshold = 16;
end
```

- # Resultados

Gravando agora um áudio de conteúdo "…2…3…4…5" e com as seguintes características:

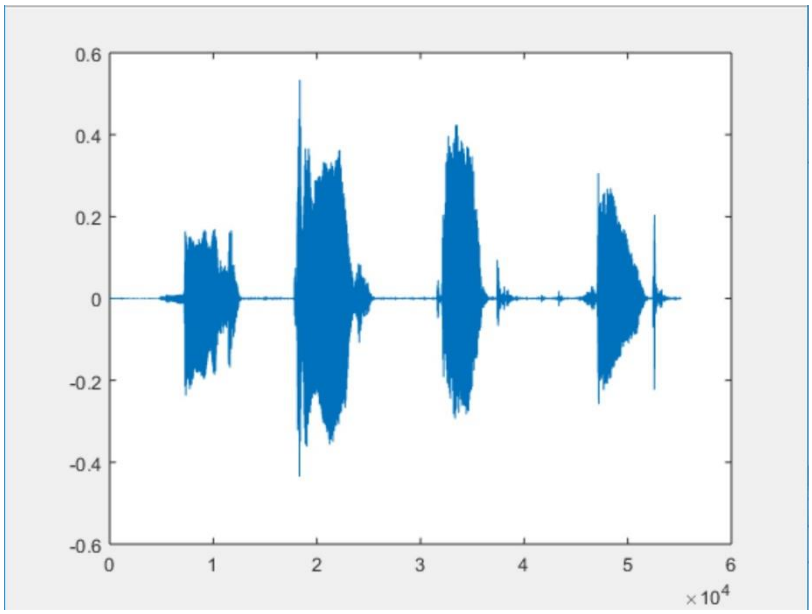

Figura 15- Áudio para o teste automático

Calculou-se o *threshold* que resultou num valor de 11.05, indicando que o SNR deste sinal estaria entre 40 e 50 dB. Após passar pelo algoritmo de remoção de silêncio, obteve-se o seguinte sinal:

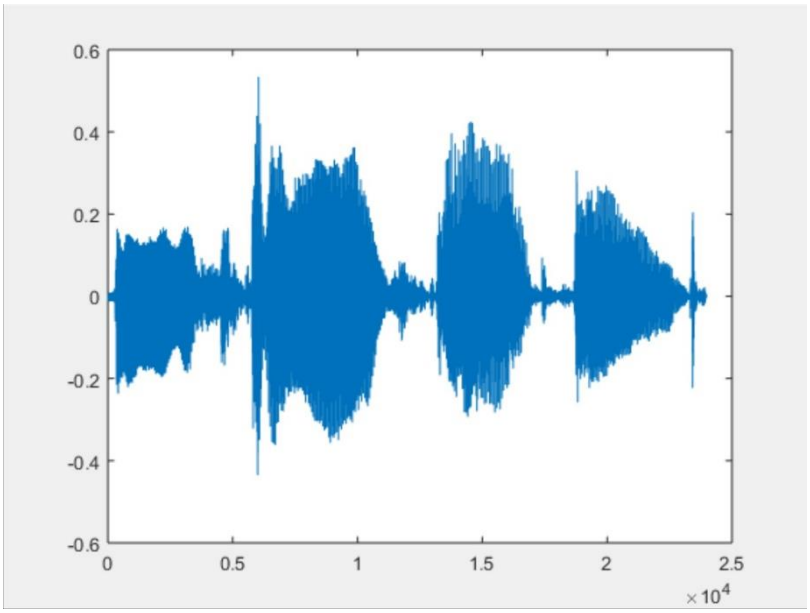

Figura 16- Áudio sem silêncios do teste automático

Provando que realmente se obtém resultados consistentes com os testes efetuados.

Conclusão

Com este trabalho foi possível verificar como um algoritmo de deteção de silêncios pode envolver um pouco de tudo, desde o cálculo do SNR até à fabricação de ruído, para que se consiga ajustar da melhor maneira o algoritmo, envolvendo componentes estatística e cálculo de probabilidades, de uma maneira intuitiva e simplificada, baseando-se inclusive em métodos de controlo como o de Shewhart e adaptando-se às implicações do problema.

Conclui-se que quanto maior o SNR, maior o *threshold* que se aplica no algoritmo, pois a influência do ruído no sinal passa a ser menor e não é necessário aplicar limites tão restritos na diferenciação de ruído e sinal.

De referir que o algoritmo usado não consegue detetar ruídos isolados que surjam descontextualizados, apenas serve para remoção do chamado *background noise* que esteja presente desde o início da gravação (uma vez que baseia a sua amostra de ruído nos primeiros pontos da gravação).

Índice de Figuras

Índice de Tabelas

Referências

Control Charts - CQE Academy. (n.d.). Retrieved June 13, 2018, from http://www.cqeacademy.com/cqe-body-of-knowledge/continuous-improvement/quality-control-tools/control-charts/

GT's Blog: Silence Removal and End Point Detection MATLAB Code. (n.d.). Retrieved June 11, 2018, from https://ganeshtiwaridotcomdotnp.blogspot.com/2011/08/silence-removal-and-end-point-detection.html?m=1

Mahalanobis, P. C. (1936). On the generalised distância in statistics. *Proceedings of the National Institute of Sciences of India*, *2*(1), 49–55.

Mancini, R., Carter, B., & Texas Instruments Incorporated. (2009). *Op amps for everyone*. Newnes/Elsevier.

Signal-to-noise ratio. (n.d.). *Scholarpedia.Org*. Retrieved from http://www.scholarpedia.org/article/Signal-to-noise_ratio

Tague, N. R. (2004). Seven Basic Quality Tools. *The Quality Toolbox*, 15. Retrieved from http://www.asq.org/learn-about-quality/seven-basic-quality-tools/overview/overview.html

ANEXOS

```
function sinal = Contamina(recorded, SNR)

%contamina o sinal recorder com um ruido de média 0 e potencia que
%corresponde ao SNR (relacao sinal-ruido) desejado

S=mean(recorded.^2); %potencia do sinal

noise = randn(1,length(recorded)); %ruído branco com distribuicao
normal
noise = noise - mean(noise); %ruído branco com media 0
noise = noise * sqrt(S/10^(SNR/10)); %ruido branco com potencia
pretendida - vem da formula do SNR
sinal=recorded+noise'; %contaminacao do sinal recebido


function [sinal_sem_ruido, ruido]=Retira_ruido(sinal,threshold,Fs)

amostrasPorFrame = floor(Fs/100);
bgSampleCount = floor(Fs/5); %a primeira quinta parte de amostras são
consideradas ruido

%calculo da media e do desvio padrao do ruido
bgSample=[];
for i=1:1:bgSampleCount
    bgSample = [bgSample sinal(i)];
end
media=mean(bgSample);
desvio=std(bgSample);

%identificar partes vozeadas para cada valor
for i=1:1:length(sinal)
    if(abs(sinal(i)-media)/desvio > threshold) %se o valor da amostra
menos a media do ruido, a dividir pelo desvio do ruido for maior que o
threshold
        voiced(i)=1; %guarda em voiced indicacao de quais amostras no
sinal sao vozeadas
    else
        voiced(i)=0;
    end
end

%identificar as partes vozeadas por frame
amostrasUteis = length(sinal)-mod(length(sinal),amostrasPorFrame);
frameCount = amostrasUteis/amostrasPorFrame; %quantos frames tem de
percorrer
voicedFrameCount = 0;
for i=1:1:frameCount %percorrer todos os frames
    cVoiced=0;
    cUnvoiced=0;
    for j=i*amostrasPorFrame-amostrasPorFrame+1:1:(i*amostrasPorFrame)
%percorre todas as amostras da frame
        if(voiced(j)==1)
            cVoiced = (cVoiced+1); %conta as amostras que sao vozeadas
        else
            cUnvoiced = cUnvoiced +1; %conta as amostras que nao sao
vozeadas
        end
    end
```

```
    if(cVoiced > cUnvoiced) %a frame contem mais amostras vozeadas que
nao vozeadas
        voicedFrameCount = voicedFrameCount+1; %conta as frames
vozeadas
        voicedUnvoiced(i)=1; %indica que e uma frame vozeada
    else
        voicedUnvoiced(i)=0; %indica que e uma frame nao vozeada
    end
end

sinal_sem_ruido=[];
ruido=[];
for i=1:1:frameCount
    for j=i*amostrasPorFrame-amostrasPorFrame+1:1:(i*amostrasPorFrame)
        if(voicedUnvoiced(i)==1) %se e uma frame vozeada
            sinal_sem_ruido = [sinal_sem_ruido sinal(j)];
        else
            ruido = [ruido sinal(j)];
        end
    end
end


function threshold = Threshold_automatico(sinal, Fs)

%calculo do SNR para posterior obtencao do valor de threshold mais
indicado

potencia_sinal = mean(sinal.^2); %potencia do sinal
potencia_ruido = mean(sinal(1:(Fs/5)).^2); %potencia do ruido,
considerando que a primeira quinta parte do sinal é ruído

SNR = 10*log10(potencia_sinal/potencia_ruido); %calculo do SNR pela
formula

if (SNR >= 0 && SNR<10)
    threshold = SNR*((0.85-0.75)/10)+0.75; %Calculo da reta do
intervalo de 0 a 10
elseif (SNR >= 10 && SNR <20)
    threshold = SNR*((1.5-0.85)/10)+(0.85-((1.5-0.85)/10)*10);
%Calculo da reta do intervalo de 10 a 20
elseif (SNR >=20 && SNR <30)
    threshold = SNR*((2.8-1.5)/10)+(1.5-((2.8-1.5)/10)*20); %Calculo
da reta do intervalo de 20 a 30
elseif (SNR >=30 && SNR <40)
    threshold = SNR*((8-2.8)/10)+(2.8-((8-2.8)/10)*30); %Calculo da
reta do intervalo de 30 a 40
elseif (SNR >=40 && SNR <50)
    threshold = SNR*((16-8)/10)+(8-((16-8)/10)*40); %Calculo da reta
do intervalo de 40 a 50
elseif (SNR >=50)
    threshold = 16;
end
```

```
function Trabalho()

Fs=11025;
recorder = audiorecorder(Fs,16,1);
disp('Tem agora 5s, fale pausadamente')
record(recorder,5);
pause('on');
pause(5);
record2 = getaudiodata(recorder);


%display do som gravado
disp('Este é o áudio a ser truncado')
sound(record2, Fs);
pause(6);

%calculo do threshold do audio
threshold = Threshold_automatico(record2,Fs)

%remoção do silêncio
[sinal_sem_ruido,ruido] = Retira_ruido(record2,threshold,Fs);
f2 =figure('Name','Sinal original vs Sinal sem
ruido','NumberTitle','off');
subplot(2,1,1);
plot(record2);
title('Sinal Original');
subplot(2,1,2);
plot(sinal_sem_ruido);
title('Sinal Sem Ruido');

disp('Este é o áudio truncado')
sound(sinal_sem_ruido,Fs);
pause(5);
```