Processamento Digital de Sinal

Trabalho Prático

Detecção de End-Point e Remoção de Silêncios

Pedro Castro
Nº55615

1 - Índice

2 - Introdução Teórica

O sinal de áudio capturado pode conter silêncio em diferentes posições, como no início de sinal, entre as palavras de uma frase, ou no final do sinal. Se os quadros silenciosos são incluídos, os recursos de modelação são gastos em partes do sinal que não contribuem para a identificação. O silêncio presente deve ser removido antes do processamento adicional.

Existem várias formas de fazer isso: a mais popular é *Short Time Energie and Zeros Crossing Rate*. Mas estes têm a sua própria limitação em relação a definição do *Threshold* como uma base ad hoc. O algoritmo utilizado, obedece a propriedades estatísticas de ruído de fundo, bem como aspectos fisiológicos da produção da fala e não assume qualquer *Threshold* ad hoc.

Normalmente, os primeiros 200ms ou mais de uma gravação de voz correspondem ao silêncio (ou ruído de fundo), porque o altifalante leva algum tempo para ler quando a gravação começa.

2.1 - Algoritmo de Detecção de End-point

1º passo:

Calculo a média (μ) e desvio padrão (σ) das amostras. O ruído de fundo é caracterizada por esta μ e σ.

2º passo:

Ir da primeira sample à última sample de gravação de voz. Em cada sample, verificar se as funções unidimensionais de distância de Mahalanobis , ou seja, $| x-\mu | / \sigma$ é maior que 3 ou não. Se a função Mahalanobis é maior do que 3, a amostra deve ser tratada como amostra voz , caso contrário, é uma amostra não-voz / silêncio.

O Threshold rejeita as amostras até 99,7%, conforme dado por $P [| x-\mu | \leq 3\sigma] = 0,997$ em uma distribuição Gaussiana aceitando, assim, apenas as amostras de voz.

3º passo:

Marcar as amostras de voz como 1 e não-voz como 0. Dividir todo o sinal de fala em janelas sem sobreposição. Representar o discurso completo por apenas zeros e uns.

4º passo:

Considere que há um número M de ‘0’ e um número N de ‘1’, numa janela. Se M ≥ N, então convertemos todos os 1 em 0 e vice-versa. Este método é adotado tendo em conta que um sistema de produção da fala que consiste em cordas vocais, língua, trato vocal, etc, não pode mudar abruptamente num curto período de janela de tempo.

5º passo:

Pegar nas amostras de voz, ou seja, as rotuladas como ‘1’, do array da janela e passá-las para um novo array. Recuperar todas as partes de voz do sinal original, ou seja, as samples rotuladas com ‘1’.

2.2 - Algoritmo de Detecção de End-point no coeficiente SNR

Neste algoritmo, atribuiu-se diferentes valores para o SNR (Signal-to-Noise Ratio), de modo a simular diferentes tipos de ambientes.

Pelas equações matemáticas do SNR temos:

$(S/N)dB = 10 \log (S/N)$

Calculei também a potência média do meu sinal de voz:

$SP = (1/Ns) \sum s^2[i]$

.Para gerar um sinal de ruído com diferentes SNR temos:

$N = sqt(SP)*n$

De modo a conhecer os melhores valores de Threshold para os diferentes valores da SNR foi testando, onde para um valor (S/R) teria de escolher o melhor valor de Threshold, que me separa-se completamente o sinal de voz do ruído.

3 - Resultados

3.1 – Gravação de voz

O objectivo inicial é gravar 5 segundos, alguém a falar, num ambiente com ou sem ruído, retirar-lhe todos os silêncios, tanto no início, como no meio das palavras, como também no fim e mostrar o resultado final, ou seja, as amostras com voz e sem silêncios.

Nas várias pesquisas que fiz, diziam para efectuar a gravação em 1 canal e a 8 bits. Depois de alguns testes, optei por fazer a gravação em 1 canais e a 16 bits, isto porque se torna mais fácil para a manipulação dos resultados.

Além deste problema de definição da melhor forma de gravar, foi necessário também encontrar uma função "paralela" à função *wavrecord*, pois esta não pode ser utilizada na versão em que fiz o trabalho. A função encontrada foi a *audiorecorder*, que funciona praticamente da mesma forma.

Depois de obter uma gravação perceptível, o gráfico obtido da gravação, que será utilizada posteriormente para aplicação do algoritmo atrás explicado, está descrito na figura 1:

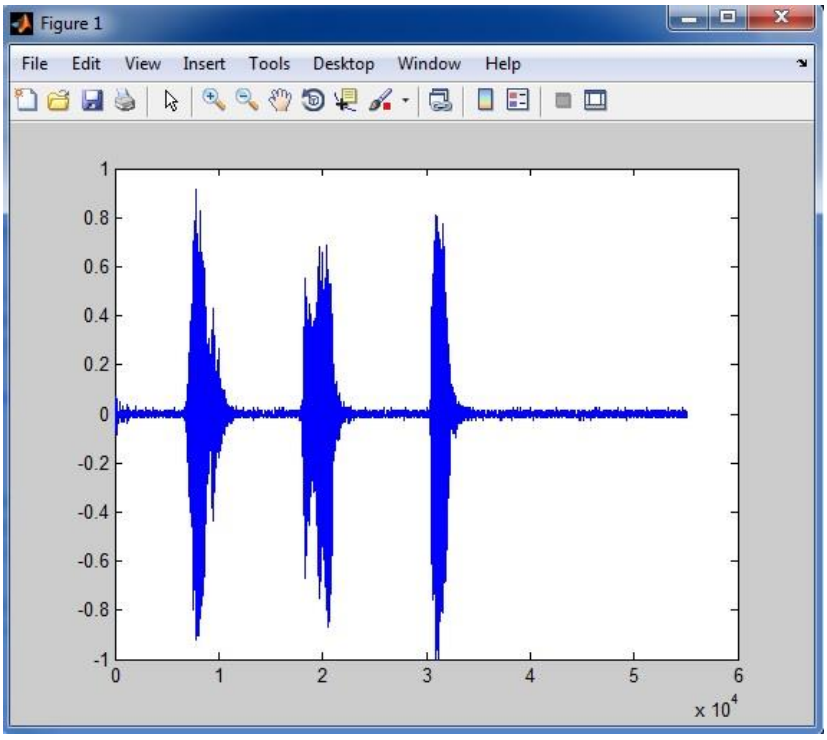


Figura 1: Sinal de entra (y)

3.2 – Aplicação do algoritmo na gravação

Depois de gravada em ficheiro, a gravação é aberta e trabalhada. Para isso foi lida a gravação do ficheiro onde foi gravada. De seguida, calculei a potência do sinal, gerei um ruído e somei o ruído ao meu sinal de entrada, a partir desse momento trabalho sobre o sinal já com um ruído inserido. De seguida analisei o sinal para cálculo da média e desvio padrão. Através destes, foi possível definir o que é voz e o que é não-voz no sinal. Através do obtido no cálculo anterior, analisei frame a frame para verificar se as amostras obtidas anteriormente continham voz ou não. A partir desta análise, foi possível classificar o sinal com '0' para frames não-voz e '1' para frames com voz. Finalmente, juntei todas essas frames '1' num novo array e, assim, obtive apenas as frames com voz.

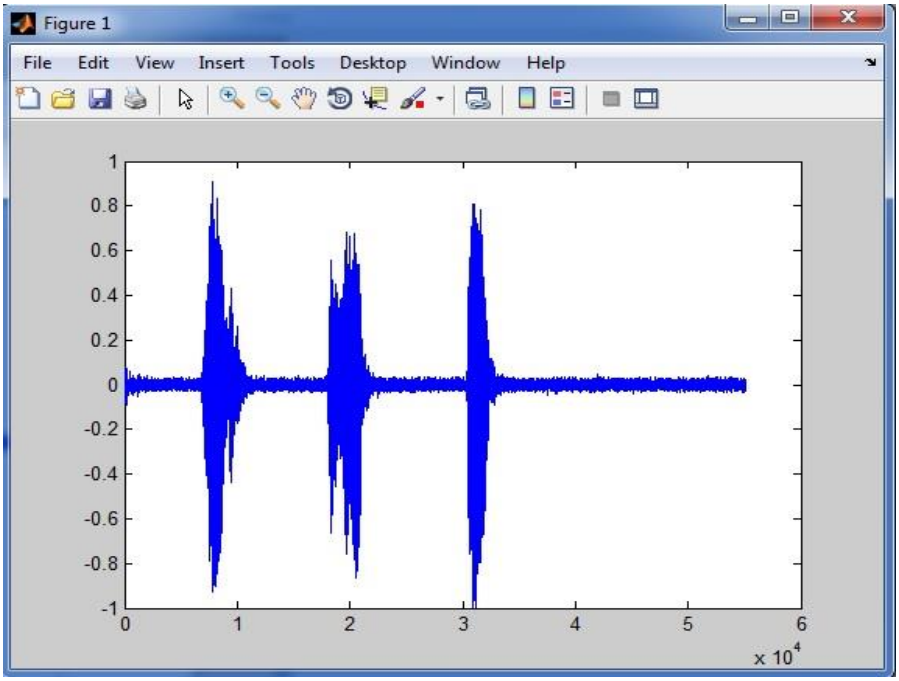
Aplicando o algoritmo de detecção de end-point e remoção de silêncios, todos os silêncios foram retirados e as palavras ditas pelo utilizador foram isoladas e juntas numa só amostra.

Para o ajuste do melhor valor de *Threshold* para determinada relação S/R foram testados os valores entre 0.5 e 2, e os resultados obtidos encontram-se na tabela a seguir.

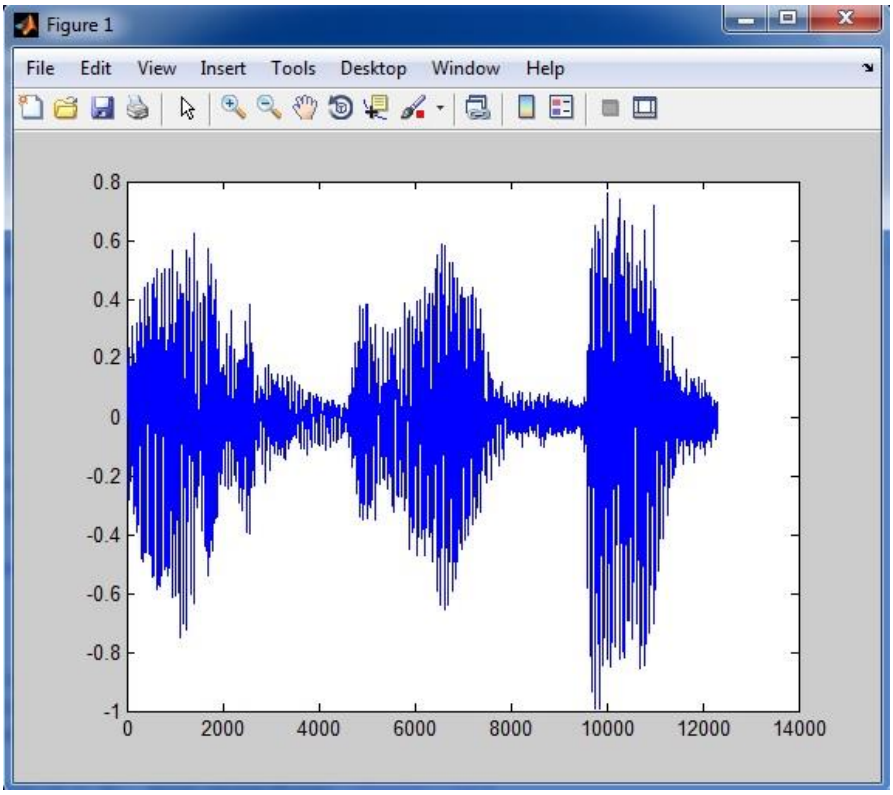
(S/N)dB	*Threshold*
70	1.6
60	1
50	0.8
40	0.7
30	0.65

Melhor valor encontrado para cada (S/R)

Comecei por testar para 70dB, pois quanto mais alto for a relação sinal ruído menor é o efeito do ruído de fundo sobre a deteção sinal, para este valor o melhor valor de *Threshold* que encontrei foi 1.6. Na primeira imagem que se segue podemos ver o sinal de voz de entrada já com o ruído (SNR de 70dB) adicionado e na segunda imagem podemos ver a saída de voz (saída final) já separada dos ruídos.

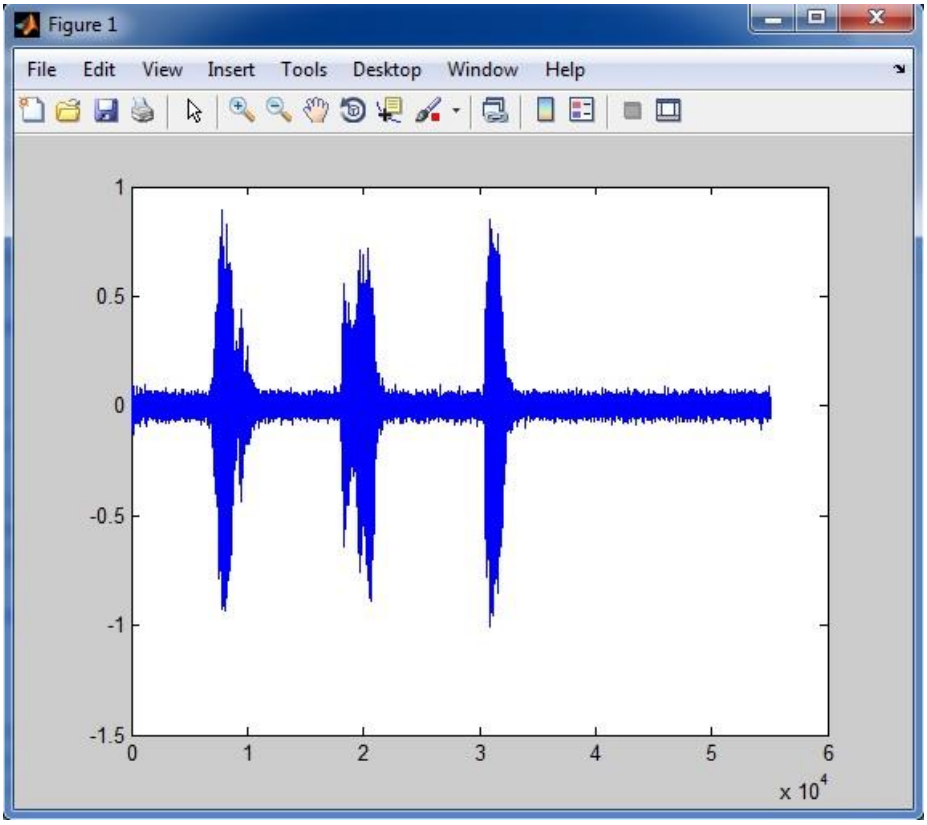


Sinal de entrada com ruído adicionado 70dB

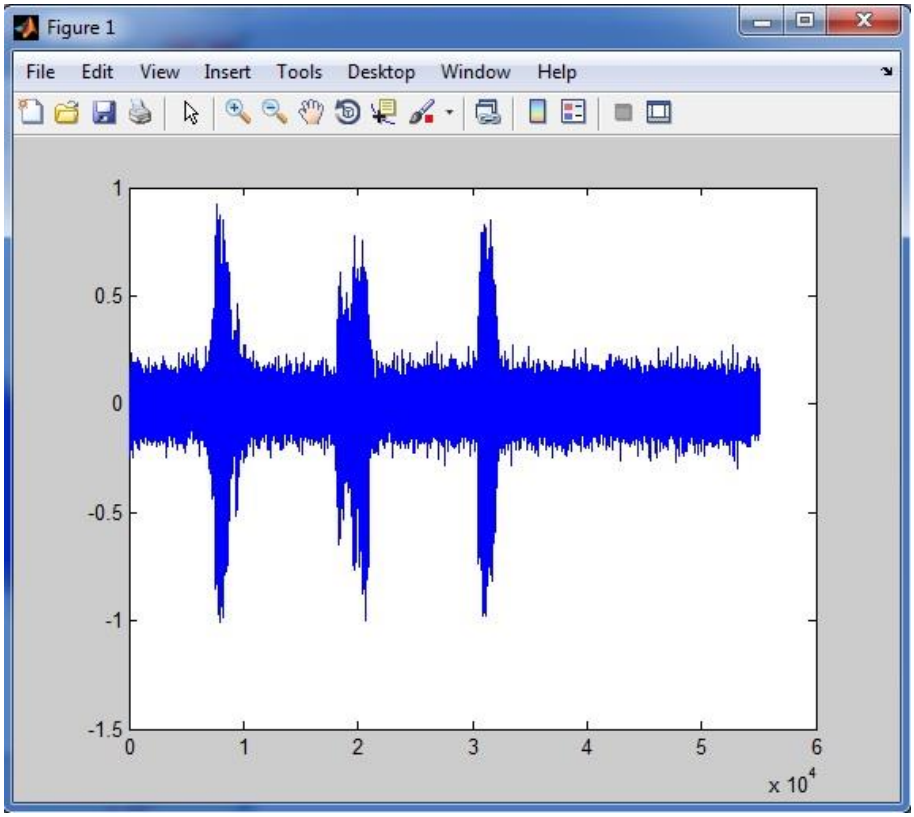


Sinal de fala já sem ruídos 70dB

Á medida que fomos baixando a S/N podemos reparar que os melhores valores de *Threshold* vão baixando também. Nas imagens seguintes já conseguimos ver o aumento do á medida que baixamos S/N.

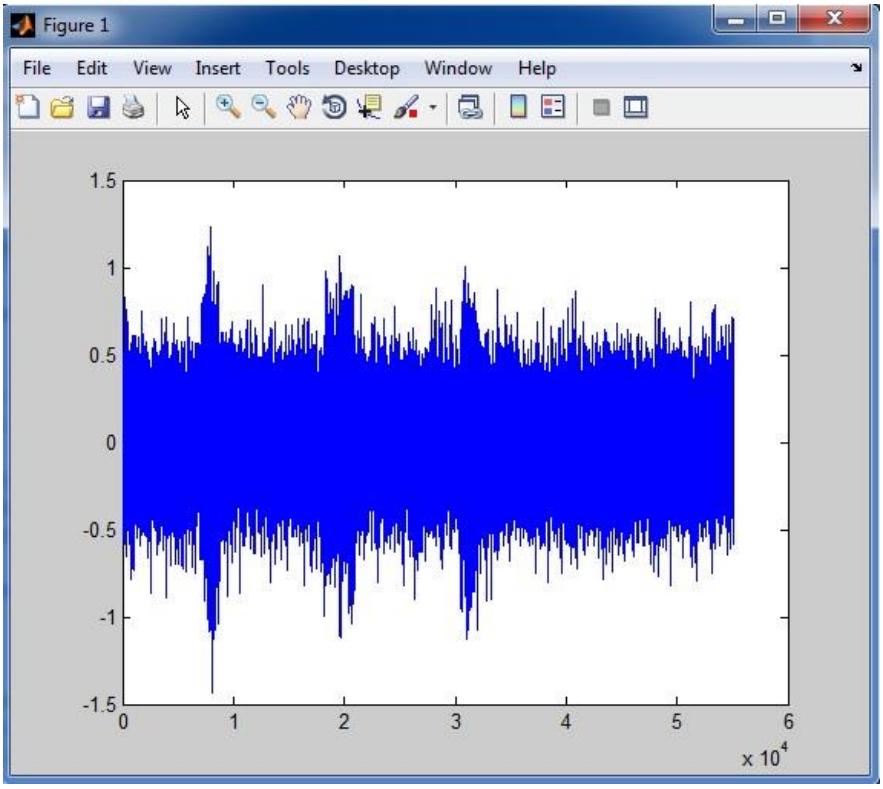


Sinal com ruído adicionado 60dB

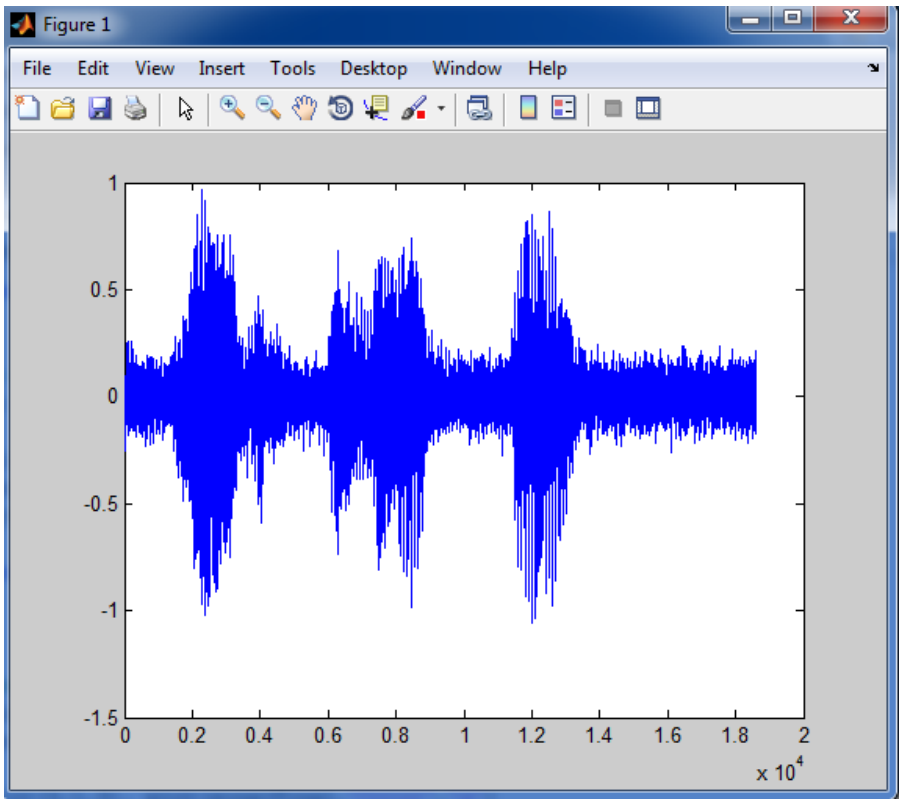


Sinal com ruído adicionado 50dB

Para valores abaixo de 40dB o ruído já é muito grande e o sinal de voz não é tão perceptível. Nas imagens a seguir podemos ver o sinal de entrada com o ruído adicionado e o sinal de voz já com a remoção de ruídos.

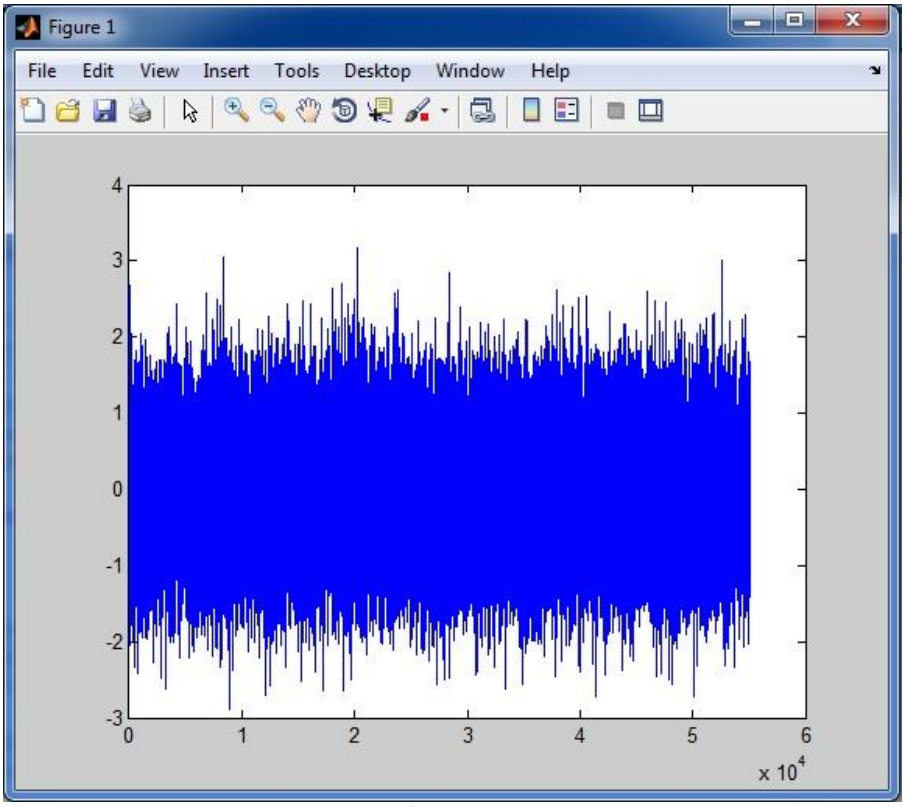


Sinal com ruído adicionado 40dB



Sinal de saída 40dB

Para uma relação sinal ruído abaixo dos 30dB já é imperceptível o sinal de entrada, uma vez que o ruído se sobrepõe, ultrapassando-o.



Sinal com ruído adicionado 30dB

4 - Conclusão

A realização deste trabalho contribuiu muito para o meu desenvolvimento no que diz respeito a processamento digital de sinal, mas também na programação em matlab.
Pelos gráficos apresentados, posso então concluir que os objectivos a que me propus na resolução deste trabalho foram conseguidos e em parte, isso deve-se também a ajuda que tive do professor.

6 - Anexos

6.1 - Código Matlab

```
%--------------------Capturar o som-------------------------

%Definir a frequência
Fs = 11025;
% Gravar a voz por 5 segundos
gravarObj = audiorecorder(Fs, 16, 1);
disp('Diz o nome')
recordblocking(gravarObj, 5);

% Guardar dados
disp('A guardar gravação');
myRecording = getaudiodata(gravarObj);

% Plot da gravação.
figure; subplot(3,1,1); plot(myRecording,'color','b')
title('Sinal original','Color','b');
%Gravar, em ficheiro

wavwrite(myRecording,Fs,8,'C:\Users\windows\Desktop\UM\PDS\PDS
final\pedro.wav');
disp('Gravação guardada em ficheiro');


%---------------ler o som a partir do ficheiro-----------------

y = wavread('C:\Users\windows\Desktop\UM\PDS\PDS final\pedro.wav');

samplePerFrame=floor(Fs/100);
bgSampleCount=floor(Fs/5);


%Definição do THRESHOLD

THRESHOLD = 0.7;

%Criação de um ruído-----cálculo da potência-----definição do S/N
soma=0;
for i=1:1:length (y);
    soma=soma + y(i)^2;
end

z=mean(soma);

n=randn (1,length(y));

x=sqrt(z/10000);

N=x*n;

Y=y+N';
```

```
%------------Calculo da media e do desvio padrão-----------------
bgSample=[];
for i=1:1:bgSampleCount
    bgSample=[bgSample Y(i)];
end
meanVal=mean(bgSample);
sDev=std(bgSample);



%Identificar, para cada valor, se há voz ou não
for i=1:1:length(Y)
   if(abs(Y(i)-meanVal)/sDev > THRESHOLD)
       voiced(i)=1;
   else
       voiced(i)=0;
   end
end


%identificar se há voz ou não para cada frame

%rejeitar as samples insuficientes da ultima frame

usefulSamples=length(Y)-mod(length(Y),samplePerFrame);
frameCount=usefulSamples/samplePerFrame;
voicedFrameCount=0;
ruidoCont=0;
for i=1:1:frameCount
   cVoiced=0;
   cUnVoiced=0;
   for j=i*samplePerFrame-samplePerFrame+1:1:(i*samplePerFrame)
       if(voiced(j)==1)
           cVoiced=(cVoiced+1);
       else
           cUnVoiced=cUnVoiced+1;
       end
   end

   %marcar as frames voz ou não-voz
   if(cVoiced>cUnVoiced)
       voicedFrameCount=voicedFrameCount+1;
       voicedUnvoiced(i)=1;
   else
       voicedUnvoiced(i)=0;
       ruidoCont=ruidoCont+1;
   end
end



sinalFinal=[];

%-----------------------------------------------------
for i=1:1:frameCount
    if(voicedUnvoiced(i)==1)
    for j=i*samplePerFrame-samplePerFrame+1:1:(i*samplePerFrame)
            sinalFinal= [sinalFinal Y(j)];
        end
```

```
    end
end
%-------------------------------------------------
ruido=[];

for i=1:1:frameCount
    if(voicedUnvoiced(i)==0)
    for j=i*samplePerFrame-samplePerFrame+1:1:(i*samplePerFrame)
            ruido= [ruido Y(j)];
        end
    end
end

%------ mostrar os gráficos do sinal original e do sinal sem
silêncios
    subplot(3,1,2), plot(Y,'Color','b')
    title('Antes de aplicada a Remocao de Silencios','Color','b');

    subplot(3,1,3), plot(sinalFinal,'Color','g')
    title('Depois de aplicada a Remocao de Silencios','Color','g');


%----------------play dos sinais-------------------------------
sound(y,Fs);
sound(ruido,Fs);
sound(sinalFinal,Fs);
```