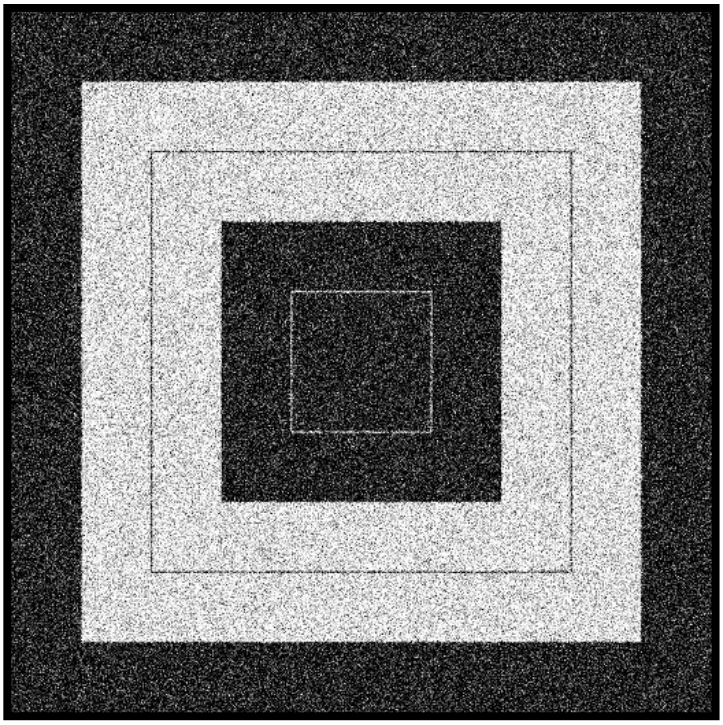
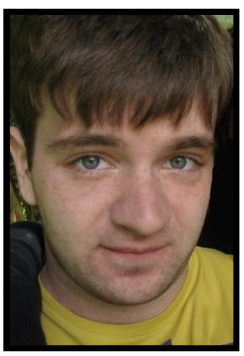
Universidade do Minho

Processamento Digital de Sinal (2012/2013)



Estudo do ruído no sinal e maneiras para o remover

Sergey Zubchevskyy (A58340, EEIC)



Introdução

Este trabalho consiste em estudo da probabilidade, ruído no sinal e maneiras de como este mesmo ruído pode ser removido.

Primeiro é necessário esclarecer o conceito da probabilidade:

"Deriva do Latim probare (provar ou testar). Informalmente, provável é uma das muitas palavras utilizadas para eventos incertos ou conhecidos, sendo também substituída por algumas palavras como "sorte", "risco", "azar", "incerteza", "duvidoso", dependendo do contexto"

http://pt.wikipedia.org/wiki/Probabilidade

O ruído que vai ser estudado, é um tipo de ruído especial, chamado "Ruído gaussiano", que tem como característica o facto de ter densidade de probabilidade corresponde a uma distribuição normal ou também conhecida como "Distribuição de Gauss".

Um exemplo de ruído gaussiano pode ser observado na seguinte figura:

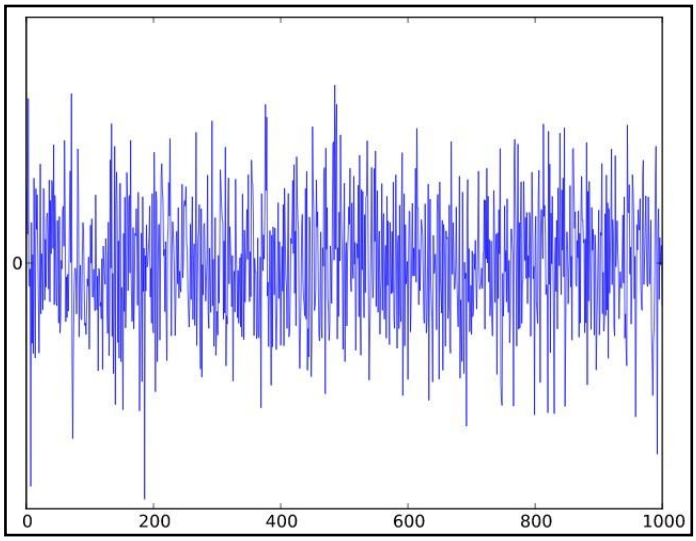


Fig. 1 - Ruído gaussiano

Distribuição de Gauss esta representada na seguinte figura:

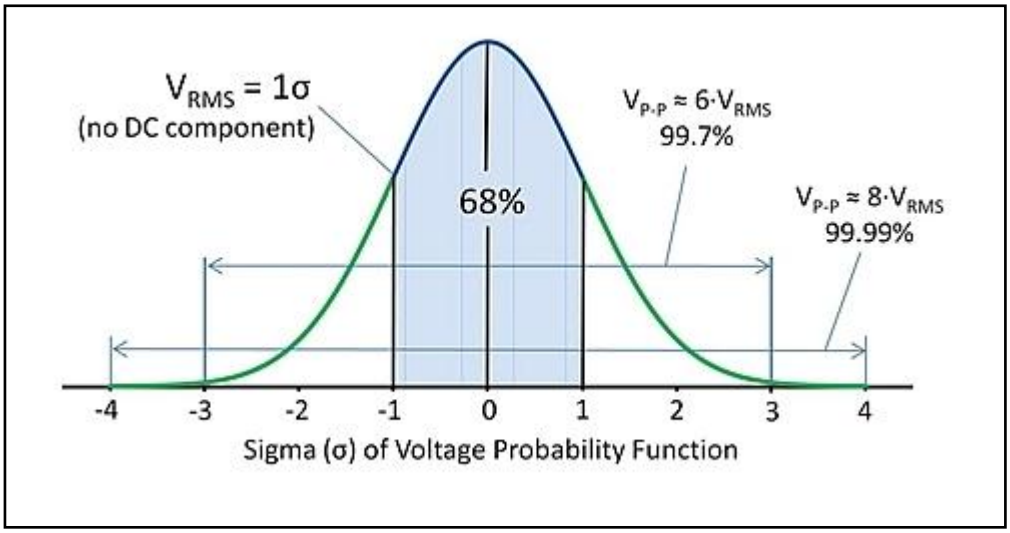


Fig. 2 – Distribuição de Gauss

É conhecido que tudo o que esta para fora de *(media – 2 * desvio-padrão)* e *(media+ 2 * desvio-padrão*) é fala, logo já temos uma forma de encontrar a fala num array de sons.

De seguida serão apresentados 3 algoritmos, feitos em MatLab, que resolvem parcialmente ou completamente o problema de identificação da fala, num array.

Solução nº 1

Primeira solução permite encontrar apenas o inicio da palavra, ou seja, algoritmo ignora todo o ruído ate o utilizador começar a falar.

```
function fala = end_nosso(texto, nseg)

        fala = 0;
        m = mean(texto(1:nseg));
        v = var(n(1:nseg));
        fim = 1;
        j = 0;

    while(fim)
        j = j+1;
        if(abs(texto(nseg + j)-m)>sqrt(v)
            fim = 0;
        end
    end

        fala = texto(nseg+j:length(texto));
end
```

Fig. 3 – Solução nº 1

Para encontrar o inicio da palavra é feita a verificação se o modulo do ponto do array, menos a media são maiores que o desvio padrão elevado ao quadrado, caso seja, então estamos no inicio de uma palavra.

Como já foi mencionado em cima, este algoritmo encontra apenas o inicio da fala no inteiro array, o problema desta solução é que parte do principio que tudo o que vem a seguir do inicio da palavra, é fala, ou seja, algoritmo não distingue pausas, ruídos altos como por exemplo queda da caneta, que tem amplitude do som alta mas duração pequena, e se no inteiro array, a pessoa disser apenas uma palavra, todo o ruído que vem a seguir dessa palavra, é considerado fala.

Solução nº 2

A segunda solução é bem mais completa do que a anterior. Tem o seguinte código:

```
function [noisebuff, buff] = end_nosso(x, nseg, wlen, step, fr)

    alfa = fr * wlen;
    m = mean(x(1:nseg));
    v = var (x(1:nseg));
    i = 0;
    buff = 0;
    noisebuff = 0;
    n = 1;
    w = 0;

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

    while(((n-1) * step + wlen) < length(x))
        w = a((n-1)*step+1:(n-1)*step+wlen);

        if sum(w) > alfa
            if ~buff
                buff = x((n-1) * step + 1:(n-1);
            else
                buff = cat(1, buff, x((n-1)*step+1:(n-1)));
            end
        else
            noisebuff=cat(1, noisebuff, x((n-1)*step+wlen));
        end
    end
    n = n+1;

end
```

Fig. 4 – Solução nº2

Esta solução em vez de encontrar o inicio da palavra, primeiro transforma todo o array de fala, em array de “0” e “1”. De seguida é feito um integrador, desta forma conseguimos distinguir palavra do ruído externo, como por exemplo queda de uma caneta. Por fim, juntamos todas as palavras num array, e esse array é devolvido ao utilizador, junto com array do ruído, caso seja necessário confirmar se não houve erros, para isso o array do ruído tem que conter apenas o ruído.

Solução nº 3

Última solução foi implementada pelo Sergey Zubchevskyy. Está solução em vez de usar media do ruído calculada no inicio do array com ruído e desvio padrão, usa o valor máximo do ruído, como base para distinguir fala do ruído.

```
function [speech, count] = our_speech(text, look_window,
            safety_window, safety_count, recovery_window, word_end)

    noise_max = max(text(1:look_window));
    count = 0; % indice do array speech
    speech = 0;
    c = 1;

while(c < length(text))

    if(text(c) > noise_max)
            for j=c:c+safety_window,
                    if(text(j) > noise_max)
                        count = count + 1;
                    end
            end

            if(count > safety_count)


                % recuperacao das posicoes anteriores
                speech = cat(1, speech, text(c-recovery_window:c));

                % copia das proximas posicoes
                speech = cat(1, speech, text(c:c+safety_window+word_end));
                c = c + safety_window + word_end;
                count = 0;

            end


    else
            c = c + 1;
    end
end
```

Fig. 5 – Solução nº 3

O script recebe os seguintes parâmetros:

- text: array com gravação de áudio;
- look_window: numero de pontos usados para calcular o valor máximo do ruído, no inicio do array “text”;
- safety_window: tamanho da janela, janela representa numero de pontos durante quais o algoritmo verifica se é uma palavra ou ruído;
- safety_count: numero de vezes que o valor do array “text” tem que ser superior ao noise_max dentro da “safety_window”, para ser uma palavra;
- recovery_window: ao encontrar uma palavra, o algoritmo usa um numero de pontos (recovery_window) para recuperar o inicio da palavra;
- word_end: parâmetro com a função semelhante ao anterior, mas desta vez, em vez de ser inicio da palavra, é o fim

As seguintes figuras vão explicar melhor o significado de cada paramento (escalas exageradas):

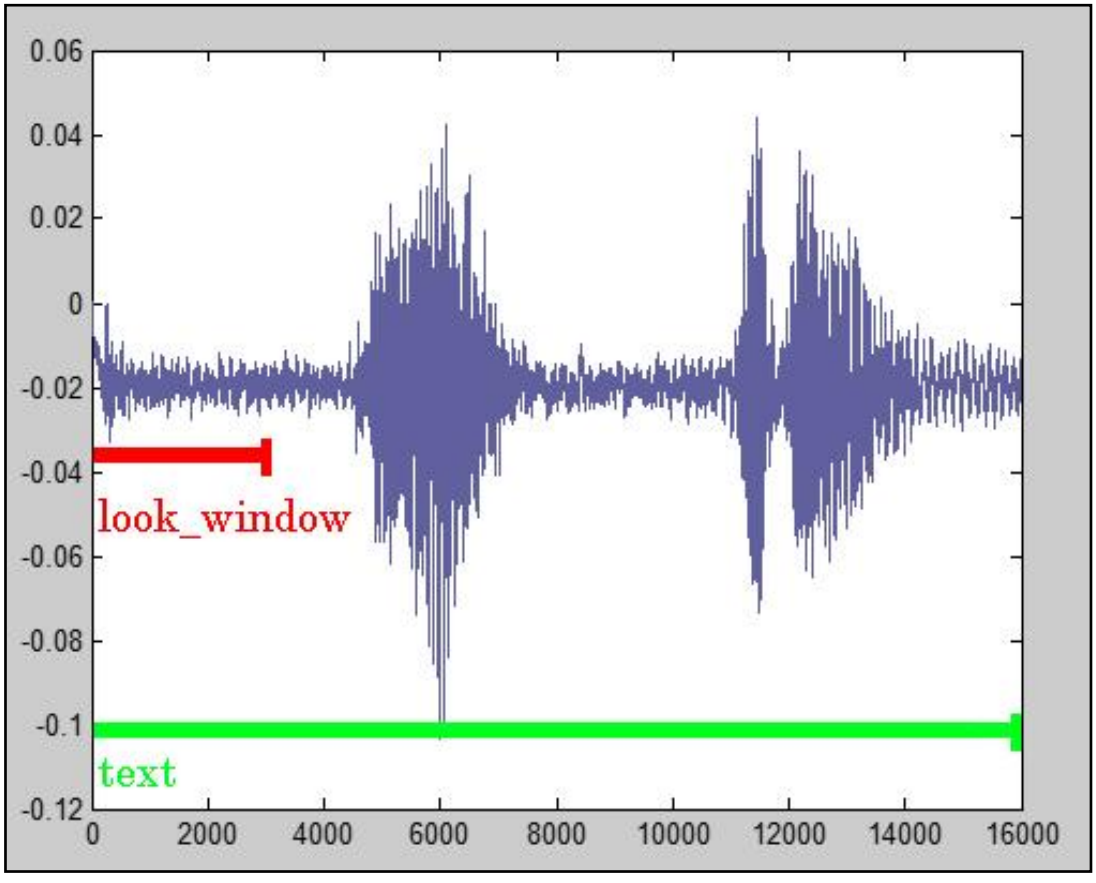


Fig. 6 – Significado de “look window” e “text”

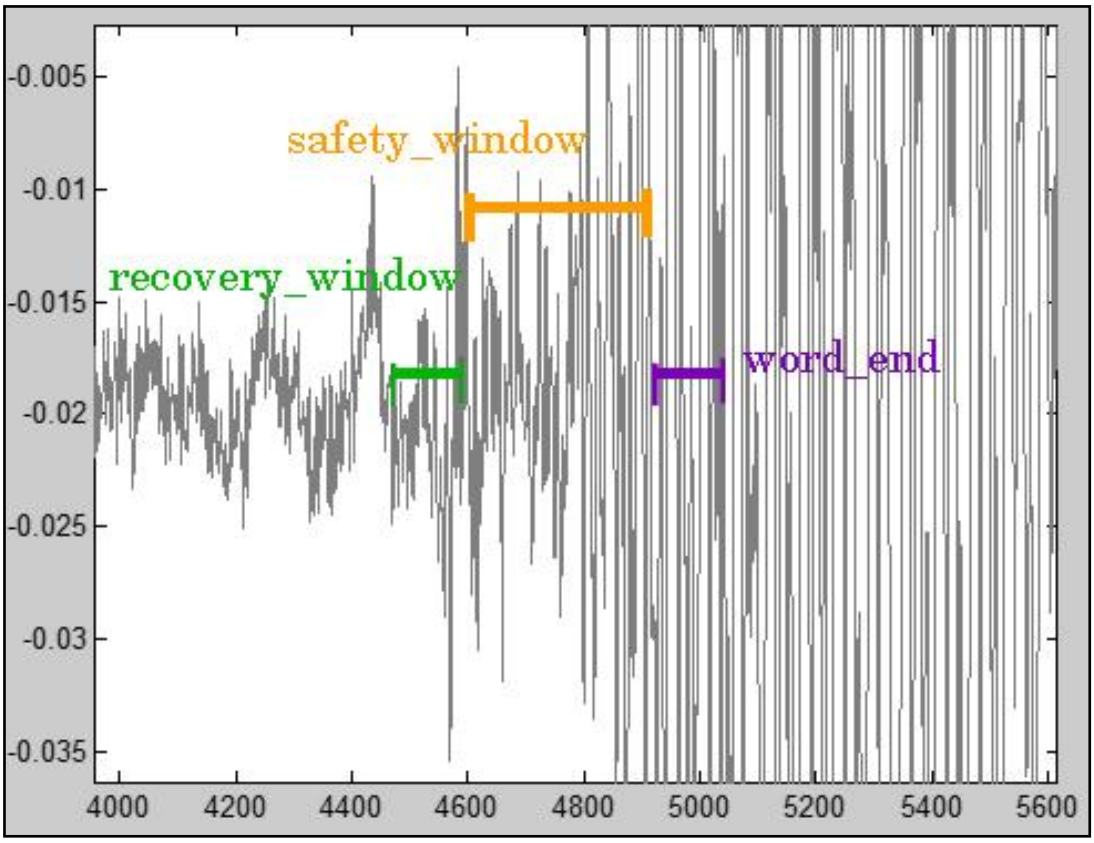


Fig. 7 – Significado de "recovery_window", "safety_window" e "word_end"

Depois de fazer varios testes, os parâmetros que deram melhores resultados foram os seguintes:

- look_window = 4000;
- safety_window = 500;
- safety_count = 100;
- recovery_window = 50;
- word_end = 500.

Exemplo da invocação da função:

[speech2] = our_speech(texto, 4000, 500, 100, 50, 500);

De seguida serão apresentados os resultados do uso do algoritmo.

Nos primeiros testes, algoritmo foi testado com apenas 1 palavra (“olá”).

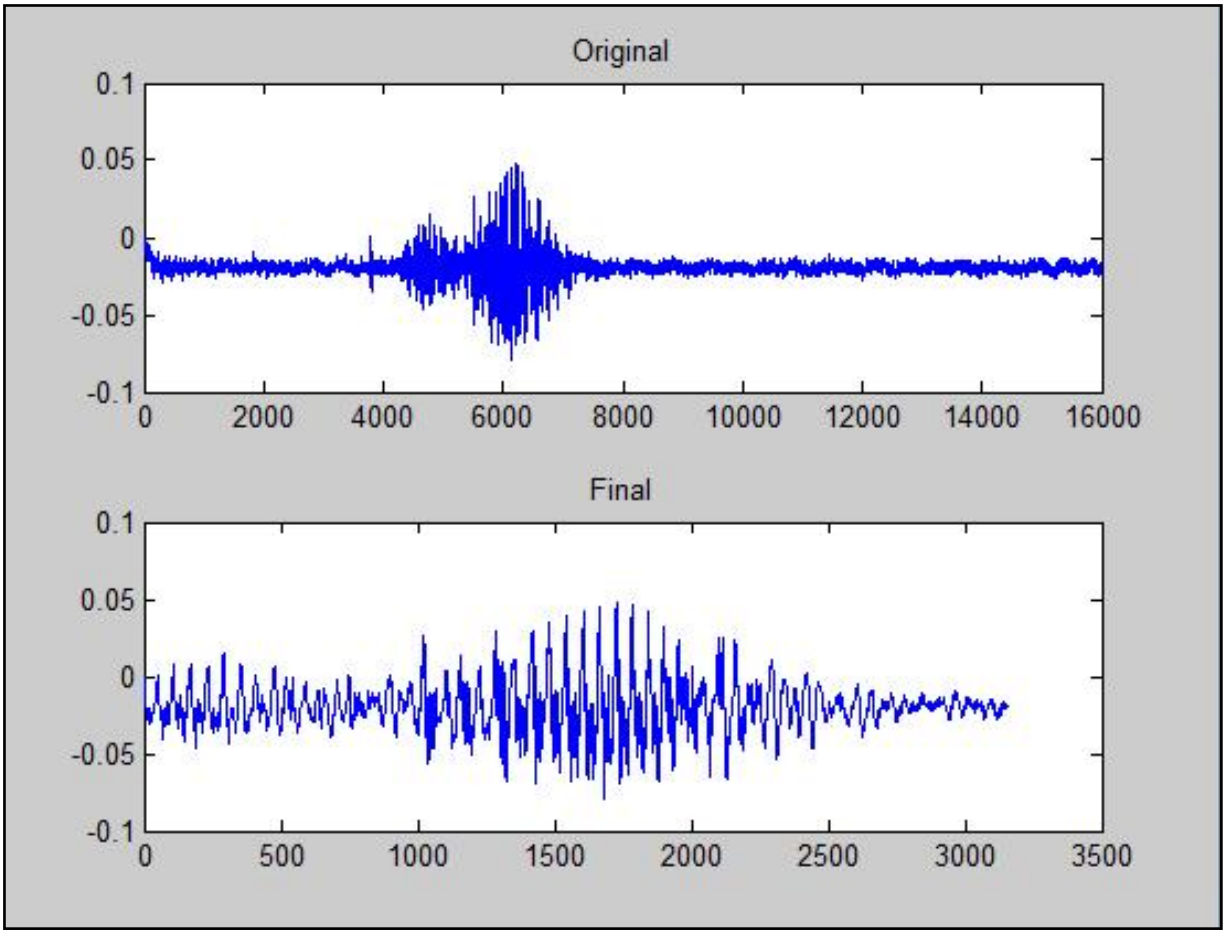


Fig. 8 – Teste da frase com apenas uma palavra (“olá”).

Na fase seguinte, a frase passou a ter 2 palavras (“olá” e “adeus”), com pausa entre elas.

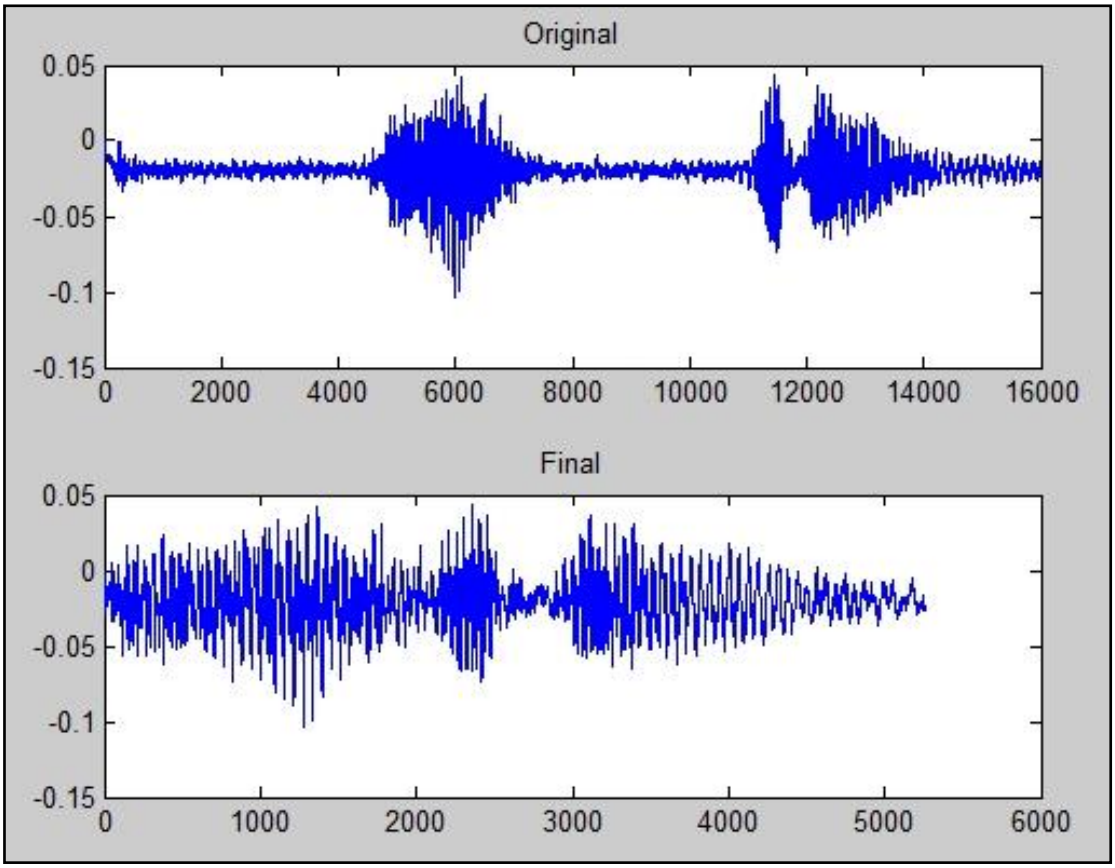


Fig. 9 – Teste da frase com 2 palavras (“olá” e “adeus” com pausa entre 2 palavras).

Esse relatório contem um anexo (“vars_ambiente2”) que contem variáveis de ambiente do MatLab, com os testes das frases, para a primeira experiencia foram usadas variáveis “texto” (array inicial), e “speech2” (array final), para a segunda experiencia foram usadas variáveis “texto2” e “speech3”.