

4

Wavelet Denoising

O capítulo 3 abordou a questão do ruído durante a extração dos atributos – as técnicas do SSCH e do PNCC, por exemplo, extraem com mais robustez a informação da voz a partir de um sinal corrompido. Mas existem outros métodos que também podem melhorar o desempenho do reconhecedor. Um deles é conhecido como *wavelet denoising*. Através da transformada wavelet, uma parte do ruído é removida do sinal de voz antes da extração de atributos.

4.1

Limpeza do Sinal

Conforme explicado em 2.4.3, o sinal de voz $x(n)$ às vezes é alterado com a adição de ruído $r(n)$, resultando num sinal corrompido $y(n)$.

$$y(n) = x(n) + r(n) \quad (29)$$

A técnica de wavelet denoising tenta transformar $y(n)$ novamente em $x(n)$, ou pelo menos num sinal similar $x'(n)$. O processo é ilustrado na Figura 40.

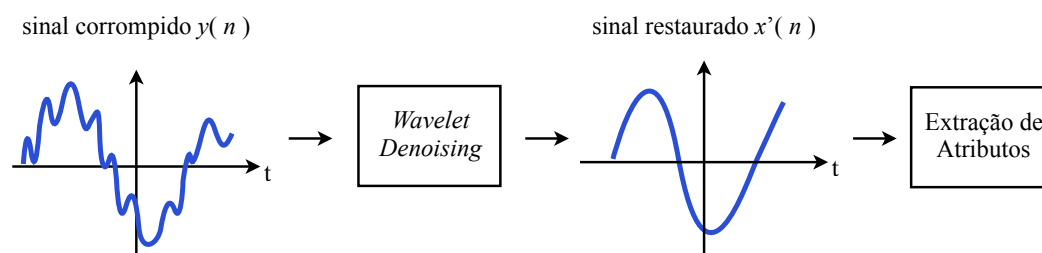


Figura 40: Restauração de um sinal corrompido, antes da extração de atributos.

Realizar a limpeza no domínio do tempo não é prático, pois é complicado diferenciar o que é sinal do que é ruído em $y(n)$. Uma alternativa inicial seria aplicar a transformada de Fourier discreta e fazer a análise no domínio da frequência. Por ser linear, a transformada permite reescrever (29) em

$$Y(k) = X(k) + R(k) \quad (30)$$

A vantagem da nova abordagem é que, em alguns casos, $X(k)$ e $R(k)$ ocupam faixas de frequência disjuntas. Isso permitiria diferenciá-los facilmente em $Y(k)$ após a adição, como mostra a Figura 41. Para transformar $Y(k)$ novamente em $X(k)$, bastaria eliminar o trecho do espectro à direita usando um filtro passa-baixa.

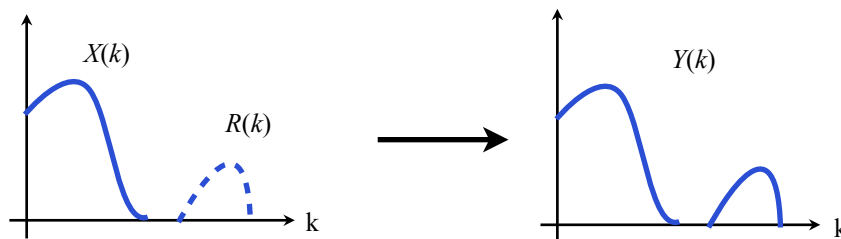


Figura 41: À esquerda, espectros do sinal e do ruído ocupando regiões diferentes. À direita, o resultado de sua adição.

Infelizmente, em muitos cenários de reconhecimento de voz, os espectros do sinal e do ruído se sobrepõem, tornando a separação por filtragem difícil. O exemplo da Figura 42 mostra que não é trivial transformar $Y(k)$ novamente em $X(k)$ nesse caso.

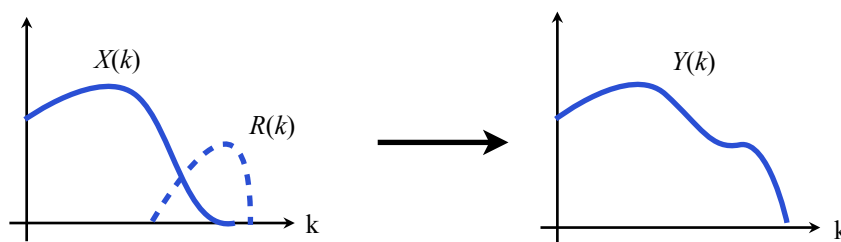


Figura 42: À esquerda, superposição dos espectros do sinal e do ruído. À direita, o resultado de sua adição.

Portanto, é necessário outro tipo de operação que consiga separar melhor a informação do ruído da informação do sinal. A transformada wavelet se mostrou uma boa alternativa, sendo explicada a seguir.

4.2 Transformada Wavelet Discreta

A transformada wavelet [29][30] pode ser considerada como uma evolução da transformada de Fourier, trazendo mais poder e flexibilidade para analisar o sinal de voz.

Relembrando Fourier, cada valor da função $G(k)$ está associado a uma oscilação numa dada frequência digital k , através das expressões

$$G(k) = \sum_{n=0}^{N-1} g(n) e^{-\frac{j2\pi kn}{N}} \quad (31)$$

$$g(n) = G^{-1}(k) = \frac{1}{N} \sum_{k=0}^{N-1} G(k) e^{\frac{j2\pi kn}{N}} \quad (32)$$

onde N é o total de amostras do sinal.

Quanto maior o valor do módulo de $G(k)$, mais intensa é a contribuição da oscilação $e^{-j2\pi kn}$ na formação de $g(n)$. Isso é muito útil na análise de sinais com muitas oscilações, que é o caso da voz humana. No entanto, a desvantagem é que $G(k)$ não informa os instantes em que essas oscilações ocorrem.

A transformada wavelet parte de um conceito similar, trazendo algumas novidades. As expressões da transformada diretas e inversa são dadas inicialmente por

$$W(a,b) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} g(n) \psi_{a,b}(n) \quad (33)$$

$$g(n) = W^{-1}(a,b) = \frac{1}{\sqrt{N}} \sum_a \sum_b W(a,b) \psi_{a,b}(n) \quad (34)$$

(33) e (34) são bem parecidas com (31) e (32), respectivamente. A diferença é a função $\psi_{a,b}(n)$ no lugar da exponencial e dois parâmetros a e b em vez de apenas de k . Mais precisamente, $\psi_{a,b}(n)$ é a modificação de uma função mãe $\psi(n)$, como indicado abaixo.

$$\psi_{a,b}(n) = \sqrt{2^a} \psi(2^a n - b) \quad (35)$$

A primeira vantagem da nova transformada é a flexibilidade de escolha de $\psi(n)$. A função mãe pode ser definida de acordo com o tipo de sinal analisado, desde que ela atenda às restrições de uma base ortonormal, que estabelecem

$$\sum_n \psi_{a,b}(n) \psi_{c,d}(n) = \begin{cases} 0, & \text{se } a \neq c, b \neq d \\ 1, & \text{caso contrário} \end{cases} \quad (36)$$

A Figura 43 a seguir mostra dois exemplos de $\psi(n)$ usados na literatura.

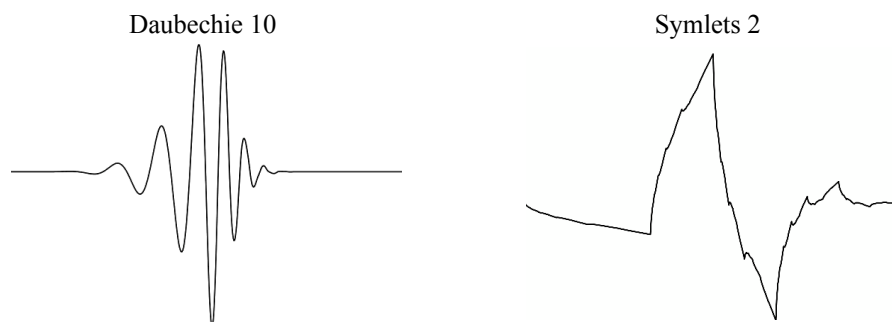


Figura 43: Exemplos de duas funções mãe $\Psi(n)$: Daubechies-10 e Symlets 2.

Outra vantagem desta transformada é ser função de dois argumentos. O parâmetro a altera a escala horizontal da função mãe, de modo similar à frequência digital k da expressão de Fourier. Já o parâmetro b desloca a função horizontalmente, trazendo uma informação de posição no tempo. A Figura 44 mostra o efeito dessas modificações.

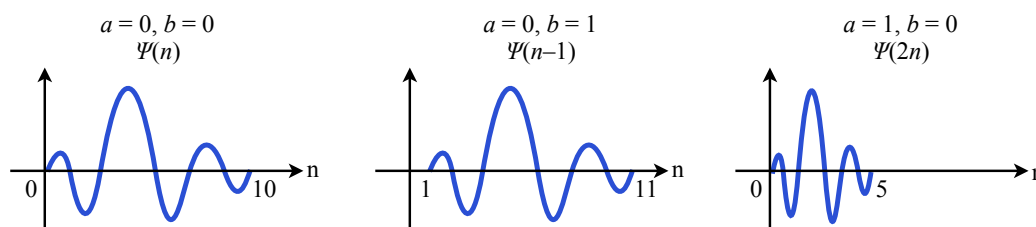


Figura 44: Efeito dos parâmetros a e b na função mãe $\Psi(n)$.

Ou seja, pode-se dizer que o domínio wavelet combina as informações do domínio do tempo com as do domínio da frequência, o que traz muito mais detalhes sobre o sinal. O resultado final é ilustrado na Figura 45: por causa do termo 2^a da expressão (35), a frequência dobra com o aumento de a , enquanto o descolamento do tempo cai pela metade.

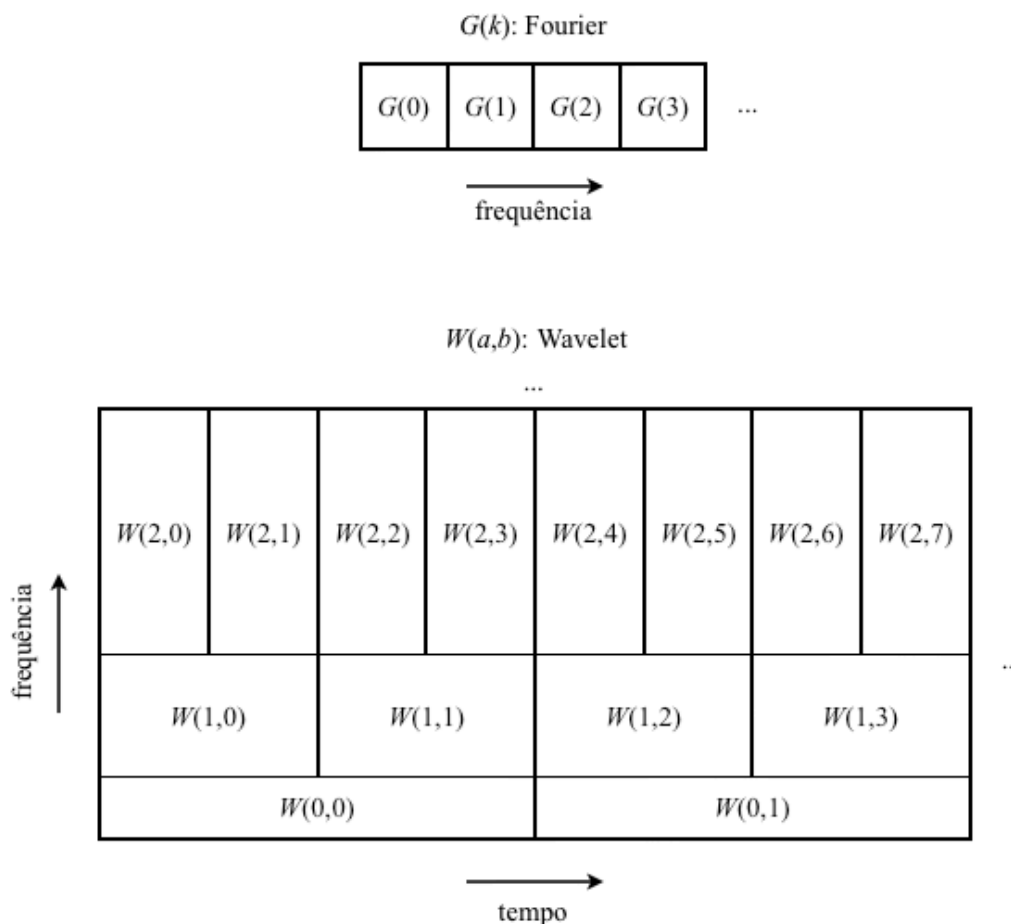


Figura 45: Representação dos valores da transformada de Fourier e da transformada Wavelet.

Um último detalhe é que, na prática, as expressões (33) e (34) não são usadas diretamente, pois seria preciso fazer um somatório para cada combinação de a e b . Existe um cálculo rápido da transformada que usa uma segunda função mãe $\phi_{a,b}(n)$, conforme explicado em [30].

4.3

Limiar (Thresholding)

A seção 4.2 mostrou que a transformada *wavelet* consegue capturar melhor as informações do sinal, combinando o domínio da frequência com o do tempo. Além disso, quanto maior o valor do módulo de $W(a,b)$, mais similar é a função $\Psi_{a,b}(n)$ com o sinal $g(n)$. Portanto, se escolhermos uma função mãe parecida com o sinal de voz, os valores maiores da transformada estarão associados a informações de voz, enquanto os menores corresponderão a tudo o que não é voz

– ou seja, ao ruído. Esse é o ponto-chave do *wavelet denoising*: eliminar valores pequenos de $W(a,b)$ para eliminar o ruído.

A Figura 46 ilustra o princípio comparando as transformadas do sinal de voz $x(n)$, do ruído $r(n)$ e do resultado de sua adição $y(n)$. Como $x(n)$ é parecido com função mãe, sua transformada tende a concentrar a informação em alguns poucos valores grandes (em módulo). Já no caso de $r(n)$, é o contrário: os números são menores e mais distribuídos. Após a soma, nota-se que a diferença continua visível.

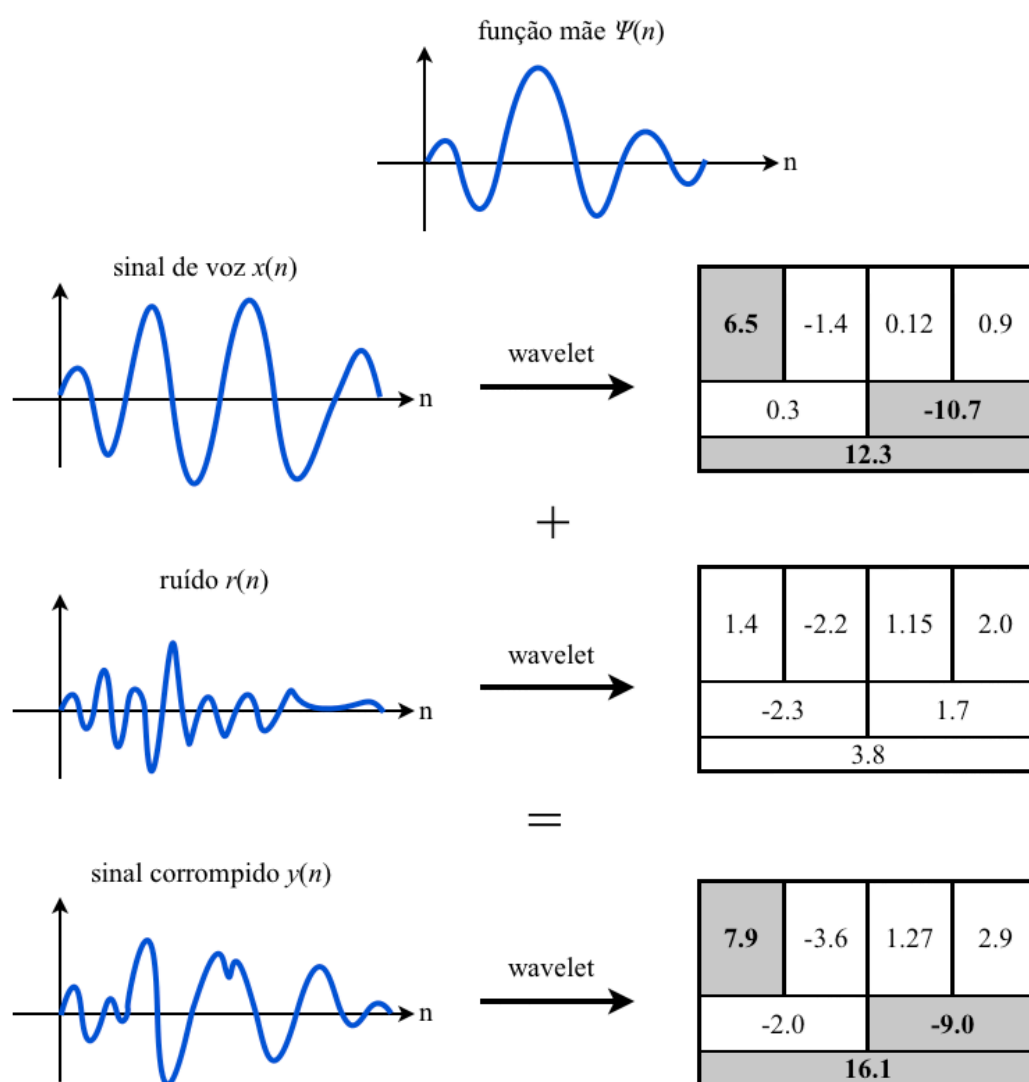


Figura 46: Valores wavelet do ruído e do sinal de voz.

Logo, ao contrário da Figura 42, a transformada wavelet de $y(n)$ consegue separar bem a informação de voz (valores com módulo grande, em negrito) da informação de ruído (valores com módulo pequeno). Portanto, para tornar $y(n)$ mais parecido com $x(n)$, basta anular os valores menores. Matematicamente,

números com módulo abaixo de um limiar t serão zerados. Esse processo, chamado de *hard thresholding*, modifica os valores W da transformada para novos valores W_H , através da expressão

$$W_H = \begin{cases} W, & |W| \geq t \\ 0, & \text{caso contrário} \end{cases} \quad (37)$$

Outro modo de fazer a eliminação é com a função *soft thresholding*, que gera novos valores W_S dados por

$$W_S = \begin{cases} \text{sgn}(W)(|W| - t), & |W| \geq t \\ 0, & \text{caso contrário} \end{cases} \quad (38)$$

onde $\text{sgn}(W)$ é o sinal de W (+1, 0 ou -1).

A primeira expressão gera uma descontinuidade: um valor ligeiramente anterior do limiar é eliminado, enquanto outro ligeiramente posterior não é modificado. Já a segunda faz uma transição suave. A diferença é vista nos gráficos da Figura 47.

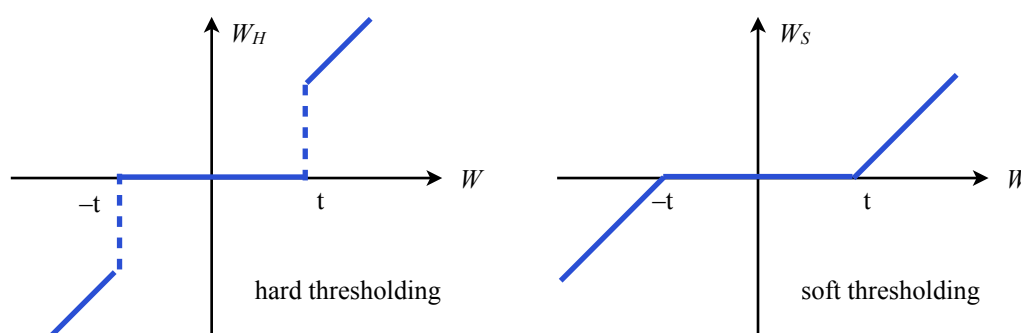


Figura 47: Comparação entre o gráfico da função *hard thresholding* com a da função *soft thresholding*.

[5] mostrou que a função *soft thresholding* gera melhores resultados e, portanto, ela foi escolhida para este trabalho.

O resumo do denoising é visto então na Figura 48: obtém-se a transformada, aplica-se o *thresholding* e desfaz-se a transformada.

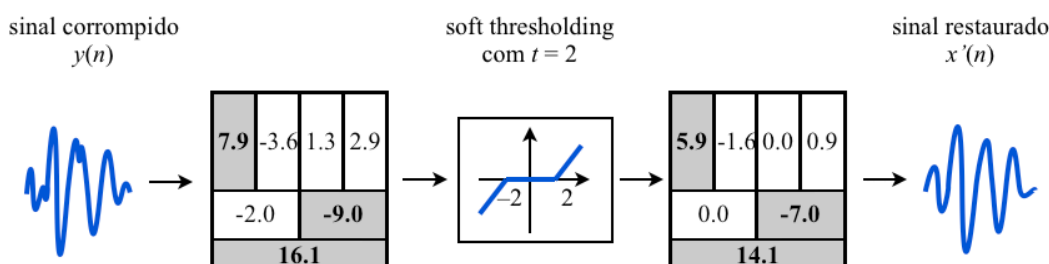


Figura 48: Exemplo da aplicação do wavelet denoising.

Vale notar que o limiar $t = 2$ foi arbitrado apenas para exemplificar os cálculos. Em 4.4, será analisado o único ponto que resta: qual o valor de t que deve ser usado? Se for muito pequeno, o ruído não será eliminado. Se for muito grande, a informação da voz será prejudicada.

4.4 Estimação do Limiar

Existem várias abordagens para estimar do limiar t do *wavelet denoising*. Este trabalho escolheu um método relativamente simples de implementar e que mostrou bons resultados em [5].

Em vez aplicar um mesmo limiar para todos os valores da transformada, cada nível a da transformada (linhas da transformada da Figura 45) tem o seu limiar, dado por

$$t_a = \gamma s_a \sqrt{2 \log(N_a)} \quad (39)$$

$$s_a = \frac{\text{median}(|W_{a,0}|, |W_{a,1}|, \dots)}{0.6745} \quad (40)$$

onde N_a é o total de valores no nível a e γ é uma constante determinada experimentalmente (vide seção 5.4.2). Essa constante não constava no artigo original, mas foi acrescentada para melhorar os resultados.

A lógica por trás das equações é bem simples. Começando por (40), o grande desafio na busca de t é determinar se um valor W da transformada é significativo ou não perante os demais. Num primeiro pensamento, a média dos módulos seria um bom indicador: se $|W|$ estiver muito abaixo da média, ele seria considerado pequeno. Entretanto, o cálculo da média é prejudicado se um dos valores for muito grande comparado com o resto. Por isso, a mediana foi escolhida; ela consegue calcular um valor intermediário sem ser tão afetada por valores extremos.

Logo, a expressão (39) traz t_a diretamente proporcional a s_a : quanto maior a mediana, maior o limiar. Ao lado de s_a , está um segundo termo em função de N_a . Isso é necessário porque um nível com poucos valores pode concentrar muita informação da voz. Nesse caso, o limiar deve ser menor, para não eliminar dados importantes para o reconhecimento.