Transformada de Cosseno

- Transformada Discreta de Cosseno (DCT) é uma transformada relacionada com a transformada discreta de Fourier
- É muito utilizada em processamento digital de imagens e compressão de dados
- Expressa uma sequência finita de dados em termos de uma soma de funções do cosseno oscilando em frequências diferentes

Eficiência na compressão

1. Proporção de compressão

$$P(c) = \frac{\#\{k : |\tilde{X}_k| > 0\}}{N} = \frac{\#\{k : |X_k| > cM\}}{N},$$

2. Distorção pós-compressão

$$D(c) = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{\|\mathbf{x}\|^2}$$

Desvantagem da DFT

- A presença de descontinuidades nas bordas no sinal criará um extravazamento de energia em grande parte do espectro
  - O mesmo acontece na versão n-D da transformada (e.g. 2-D, em imagens)
  - Seria necessário um limiar relativamente alto para manter a maior parte da informação original, diminuindo significativamente a taxa de compressão.
- Como podemos resolver esse problema ?

Divisão em blocos

- Uma forma é dividir o sinal em blocos, e comprimí-los individualmente
- Se os blocos forem relativamente pequenos, essas descontinuidades não irão aparecer
  - Aumentando a taxa de compressão
- No caso de imagens, o padrão JPEG especifica que a imagem seja dividida em blocos de 8x8 pixels
- Também ainda não nos utilizamos de nenhum processo de quantização das frequências

Idéia

- Podemos tentar resolver o problema da descontinuidade nas bordas, estendendo o sinal para o dobro do seu comprimento original
- Essa extensão é criada através da reflexão do sinal a partir do seu fim
- Calcularemos a DFT este sinal agora dobrado em comprimento, mas somente manteremos a sua metade para reconstrução.

Reflexão simétrica

Compression difficulties arise when x_0 differs substantially from x_{N-1}. Let us thus define an extension $\tilde{\mathbf{x}} \in \mathbb{C}^{2N}$ of $\mathbf{x}$ as

$$\tilde{x}_k = \begin{cases} x_k, & 0 \le k \le N-1, \\ x_{2N-k-1}, & N \le k \le 2N-1. \end{cases} \tag{3.2}$$

We then have $\tilde{\mathbf{x}} = (x_0, x_1, \ldots, x_{N-2}, x_{N-1}, x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$, so $\tilde{\mathbf{x}}$ is just $\mathbf{x}$ reflected about the right endpoint and $\tilde{x}_0 = \tilde{x}_{2N-1} = x_0$. When we take the $2N$-point DFT of $\tilde{\mathbf{x}}$, we will not encounter the kind of edge effects discussed above. Note that we duplicate x_{N-1} in our extension, called the *half-point symmetric extension*.

DFT do sinal estendido

The DFT $\tilde{\mathbf{X}}$ of the vector $\tilde{\mathbf{x}} \in \mathbb{C}^{2N}$ has components

$$\tilde{X}_k = \sum_{m=0}^{2N-1} \tilde{x}_m e^{-2\pi ikm/(2N)} = \sum_{m=0}^{2N-1} \tilde{x}_m e^{-\pi ikm/N}. \tag{3.3}$$

Note that we use a $2N$-point DFT. We can split the sum on the right in equation (3.3) and obtain

$$\tilde{X}_k = \sum_{m=0}^{N-1} \left(\tilde{x}_m e^{-\pi ikm/N} + \tilde{x}_{2N-m-1} e^{-\pi ik(2N-m-1)/N} \right), \tag{3.4}$$

since as the index of summation m in (3.4) assumes values $m = 0, \ldots, N-1$, the quantity $2N - m - 1$ in the second part of the summand assumes values $2N-1, \ldots, N$ in that order. Thus the sums in (3.3) and (3.4) are in fact identical.

DFT do sinal estendido

From equation (3.2), we have $\tilde{x}_{2N-m-1} = \tilde{x}_m = x_m$ for $0 \le m \le N-1$. Use this in (3.4), along with

$$e^{-\pi ik(2N-m-1)/N} = e^{\pi ik(m+1)/N} e^{-2\pi ik} = e^{\pi ik(m+1)/N}$$

to obtain

$$\tilde{X}_k = \sum_{m=0}^{N-1} \left(x_m e^{-\pi ikm/N} + x_m e^{\pi ik(m+1)/N} \right)$$

$$= e^{\pi ik/2N} \sum_{m=0}^{N-1} \left(x_m e^{-\pi ik(m+1/2)/N} + x_m e^{\pi ik(m+1/2)/N} \right)$$

$$= 2e^{\pi ik/2N} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi k(m+1/2)}{N} \right).$$

This defines the DFT coefficients $\tilde{X}_k$ on the range of $0 \le k \le 2N-1$.

Definição de DCT e IDCT

Equations (3.6) through (3.9) form a natural transform/inverse transform pair and are, up to a minor change of scale, the DCT. Let us modify the definition of the c_k slightly (we will of course compensate in equation (3.9)) and replace the c_k by C_k with

$$C_0 = \sqrt{\frac{1}{N}} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi 0(m+1/2)}{N} \right) = \sqrt{\frac{1}{N}} \sum_{m=0}^{N-1} x_m, \tag{3.10}$$

$$C_k = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi k(m+1/2)}{N} \right), \quad 1 \le k \le N-1. \tag{3.11}$$

Definition 3.3 Let $\mathbf{C} \in \mathbb{C}^N$. The IDCT of $\mathbf{C}$ is the vector $\mathbf{x} \in \mathbb{C}^N$ defined by

$$x_m = \frac{1}{\sqrt{N}} C_0 + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} C_k \cos\left(\frac{\pi k(m+1/2)}{N} \right) \tag{3.12}$$

for $0 \le m \le N-1$.

Formulação matricial

From equations (3.10) and (3.11), it is easy to see that we can compute the DCT as $\mathbf{C} = \mathcal{C}_N \mathbf{x}$, where

$$\mathcal{C}_N = \begin{bmatrix} \frac{1}{\sqrt{N}} & \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\frac{1}{2}\right) & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\frac{3}{2}\right) & \cdots & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\frac{2N-1}{2}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N}\frac{1}{2}\right) & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N}\frac{3}{2}\right) & \cdots & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N}\frac{2N-1}{2}\right) \end{bmatrix}.$$

The first row consists entirely of entries $1/\sqrt{N}$. For $k \ge 1$, the row k and column m entries are given by $\sqrt{2/N}\cos([\pi k(2m+1)]/(2N))$.

A careful examination of the IDCT in equation (3.12) shows that $\mathbf{x} = \mathcal{C}_N^T \mathbf{C}$ so that $\mathcal{C}_N^{-1} = \mathcal{C}_N^T$. The matrix $\mathcal{C}_N$ is thus orthogonal, that is,

$$\mathcal{C}_N^T \mathcal{C}_N = \mathcal{C}_N \mathcal{C}_N^T = \mathbf{I}_N. \tag{3.14}$$

Eficiência na compressão

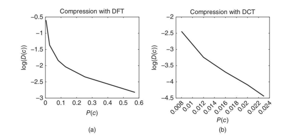
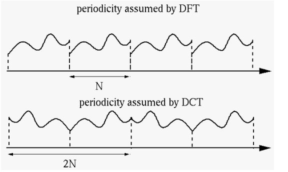


Figure 3.10 Data $\log(D(c))$ versus $P(c)$ for compression of $f(t) = t$ using DFT (a) and DCT (b).

DFT vs DCT



DCT em 2D

An explicit formula for the two-dimensional DCT easily falls out of equation (3.17) and is given by

$$\hat{a}_{k,l} = u_k v_l \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} \cos\left(\frac{\pi}{m} k \left(r + \frac{1}{2}\right) \right) \cos\left(\frac{\pi}{n} l \left(s + \frac{1}{2}\right) \right), \tag{3.18}$$

where

$$u_0 = \sqrt{\frac{1}{m}}, \quad u_k = \sqrt{\frac{2}{m}}, \quad k > 0,$$
$$v_0 = \sqrt{\frac{1}{n}}, \quad v_l = \sqrt{\frac{2}{n}}, \quad l > 0. \tag{3.19}$$

The basic waveforms are the $m \times n$ matrices $\mathcal{C}_{m,n,k,l}$ (or $\mathcal{C}_{k,l}$ when m, n are fixed) where $0 \le k \le m-1, 0 \le l \le n-1$. The row r and column s entries of $\mathcal{C}_{k,l}$ are given by

$$\mathcal{C}_{k,l}(r,s) = u_k v_l \cos\left(\frac{\pi}{m} k \left(r + \frac{1}{2}\right) \right) \cos\left(\frac{\pi}{n} l \left(s + \frac{1}{2}\right) \right).$$

Compressão JPEG

- É dividido em 5 etapas:
  - Transformação de RGB para YCbCr
  - Downsampling
  - Transformada discreta dos cossenos em 2D
  - Quantização
  - Codificação
- Toda a imagem a ser comprimida para JPEG é vista como um conjunto de blocos de 8x8 pixels.

Transformada discreta de cosseno

$$G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos\left[\frac{\pi}{8} \left(x + \frac{1}{2}\right) u \right] \cos\left[\frac{\pi}{8} \left(y + \frac{1}{2}\right) v \right]$$

where

- u is the horizontal spatial frequency, for the integers $0 \le u < 8$.
- v is the vertical spatial frequency, for the integers $0 \le v < 8$.
- $\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$ is a normalizing function
- $g_{x,y}$ is the pixel value at coordinates (x, y)
- $G_{u,v}$ is the DCT coefficient at coordinates (u, v)

Quantização

- É nesta fase que o tamanho do arquivo diminui drasticamente
- A partir de um coeficiente de compactação, os coeficientes DCT são "truncados"
- A grande quantidade de informação perdida nesta fase é irreversível e por isso, uma imagem JPEG possui menos detalhes que a original

$$B_{j,k} = \operatorname{round}\left(\frac{G_{j,k}}{Q_{j,k}} \right)$$

G são os coeficientes DCT
Q é a matriz de quantização

Exemplo de quantização

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Codificação por entropia

- A codificação de Huffman é a mais utilizada
  - Usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo
- Por isso, deve-se armazenar no cabeçalho do arquivo JPEG as tabelas de Huffman resultantes da codificação:
  - Y/DC
  - CbCr/DC
  - Y/AC
  - CbCr/AC

Descompressão

- Dado um arquivo JPEG (codificado), para descodificá-lo deve-se que executar os passos anteriores da forma reversa
  - Descodificar dados através das tabelas de Huffman
  - Descodificar zeros agrupados pelo RLE
  - "Desquantiza" blocos
  - Aplicar inversa da transformada DCT2
  - Transformar YCbCr para RGB

Introdução

- O termo *filtragem* refere-se à alteração sistemática de conteúdo de frequência(s) de um sinal ou imagem
  - Em particular, desejamos filtrar algumas frequências
- Esta operação é de natureza linear e normalmente realizada no domínio do tempo ou frequência
- A *convolução* é uma ferramenta importante para filtragem no domínio do tempo
  - A forma da convolução depende do espaço vetorial no qual os sinais residem

Remoção de ruído

- A sequência típica de operações para a remoção de ruído de um sinal no domínio de frquências:
  - Transformar o sinal para o domínio de frequências utilizando DFT
  - Zerar o(s) componente(s) correspondentes às frequências desejadas
  - Reconstruir o sinal utilizando a transformada DFT inversa

Comportamento da média

$x(t) = \sin(2\pi qt)$, then

$$w_k = \frac{1}{2}x_{k-1} + \frac{1}{2}x_k$$
$$= \frac{1}{2}\sin\left(\frac{2\pi q(k-1)}{N}\right) + \frac{1}{2}\sin\left(\frac{2\pi qk}{N}\right)$$
$$= \left(\frac{1+\cos(2\pi q/N)}{2}\right)\sin\left(\frac{2\pi qk}{N}\right) - \frac{1}{2}\sin\left(\frac{2\pi q}{N}\right)\cos\left(\frac{2\pi qk}{N}\right)$$
$$= A\sin\left(\frac{2\pi qk}{N}\right) - B\cos\left(\frac{2\pi qk}{N}\right),$$

where $A = (1 + \cos(2\pi q/N))/2$, $B = \frac{1}{2}\sin(2\pi q/N)$, and we make use of $\sin(a-b) = \sin(a)\cos(b) - \cos(a)\sin(b)$ with $a = 2\pi qk/N$, $b = 2\pi q/N$. If q is close to zero (more accurately, if q/N is close to zero), then $A \approx 1$ and $B \approx 0$. As a consequence, $w_k \approx \sin(2\pi qk/N) = x_k$. In short, a low-frequency waveform passes through the two-point averaging process largely unchanged. On the other hand, if $q \approx N/2$ (the Nyquist frequency), then $A \approx 0$ and $B \approx 0$, so $w_k \approx 0$. The highest frequencies that we can represent will be nearly zeroed out by this process.

Convolução

The low-pass filtering operation above is a special case of convolution, an operation that plays an important role in signal and image processing, and indeed many areas of mathematics. In what follows, we will assume that all vectors in $\mathbb{C}^N$ are indexed from 0 to $N-1$. Moreover, when convenient, we will assume that the vectors have been extended periodically with period N in the relevant index, via $x_k = x_{k \bmod N}$.

Remark 4.1 If we extend a vector $\mathbf{x}$ periodically to all index values k via $x_k = x_{k \bmod N}$, then for any value of m we have

$$\sum_{k=m}^{m+N-1} x_k = \sum_{k=0}^{N-1} x_{k+m} = \sum_{k=0}^{N-1} x_k.$$

Definição de convolução

Let us recast the filtering operation above in a more general format. We begin with a definition.

Definition 4.1 Let $\mathbf{x}$ and $\mathbf{y}$ be vectors in $\mathbb{C}^N$. The circular convolution of $\mathbf{x}$ and $\mathbf{y}$ is the vector $\mathbf{w} \in \mathbb{C}^N$ with components

$$w_r = \sum_{k=0}^{N-1} x_k y_{(r-k) \bmod N} \tag{4.2}$$

for $0 \le r \le N-1$. The circular convolution is denoted $\mathbf{w} = \mathbf{x} * \mathbf{y}$.

Convolução

We compute the quantity w_0 by taking the vector $\mathbf{x}$ and the vector $\mathbf{y}$ indexed in reverse order starting at $k = 0$, and lining them up:

$$\begin{matrix} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ y_0 & y_{N-1} & y_{N-2} & \cdots & y_1. \end{matrix}$$

Then w_0 is simply the dot product of these two rows, $w_0 = x_0y_0 + x_1y_{N-1} + \cdots + x_{N-1}y_1$. To compute w_1, take the y_1 at the end of the second row and move it to the front, pushing other components to the right, as

$$\begin{matrix} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ y_1 & y_0 & y_{N-1} & \cdots & y_2. \end{matrix}$$

Finalmente

The overall computation can be summarized as

	x_0	x_1	x_2	$\cdots$	x_{N-1}
w_0	y_0	y_{N-1}	y_{N-2}	$\cdots$	y_1
w_1	y_1	y_0	y_{N-1}	$\cdots$	y_2
w_2	y_2	y_1	y_0	$\cdots$	y_3
				$\vdots$	
w_{N-1}	y_{N-1}	y_{N-2}	y_{N-3}	$\cdots$	y_0.

Each w_r is obtained as the dot product of the corresponding row with the vector $(x_0, x_1, \ldots, x_{N-1})$.

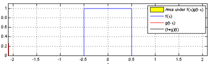
Convolução

The expression

$$s_o(x) = \int_{-\infty}^{+\infty} s_i(\xi)h(x-\xi)d\xi = s_i * h$$

is called *convolution*, defined as:

$$s_1(x) * s_2(x) = \int_{-\infty}^{+\infty} s_1(\xi)s_2(x-\xi)d\xi = \int_{-\infty}^{+\infty} s_1(x-\xi)s_2(\xi)d\xi$$



Propriedades

Theorem 4.1 *Let* $\mathbf{x}$, $\mathbf{y}$, *and* $\mathbf{w}$ *be vectors in* $\mathbb{C}^N$. *The following hold:*

1. *Linearity:* $\mathbf{x} * (a\mathbf{y} + b\mathbf{w}) = a(\mathbf{x} * \mathbf{y}) + b(\mathbf{x} * \mathbf{w})$ *for any scalars* a, b.
2. *Commutativity:* $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$.
3. *Matrix formulation: If* $\mathbf{w} = \mathbf{x} * \mathbf{y}$, *then* $\mathbf{w} = \mathbf{M}_\mathbf{y}\mathbf{x}$, *where* $\mathbf{M}_\mathbf{y}$ *is the* $N \times N$ *matrix*

$$\mathbf{M}_\mathbf{y} = \begin{bmatrix} y_0 & y_{N-1} & y_{N-2} & \cdots & y_1 \\ y_1 & y_0 & y_{N-1} & \cdots & y_2 \\ y_2 & y_1 & y_0 & \cdots & y_3 \\ & & & \vdots & \\ y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_0 \end{bmatrix}.$$

   In particular, the row k *and column* m *entries of* $\mathbf{M}_\mathbf{y}$ *are* $y_{(k-m) \bmod N}$, *rows and columns indexed from 0. Moreover,* $\mathbf{M}_\mathbf{x}\mathbf{M}_\mathbf{y} = \mathbf{M}_{\mathbf{x}*\mathbf{y}}$.
   The matrix $\mathbf{M}_\mathbf{y}$ *is called the* circulant matrix *for* $\mathbf{y}$. *Note that the rows of* $\mathbf{M}_\mathbf{y}$, *or the columns, can be obtained by the circular shifting procedure described after equation (4.2).*
4. *Associativity:* $\mathbf{x} * (\mathbf{y} * \mathbf{w}) = (\mathbf{x} * \mathbf{y}) * \mathbf{w}$.
5. *Periodicity: If* x_k *and* y_k *are extended to be defined for all* k *with period* N, *then the quantity* w_r *defined by equation (4.2) is defined for all* r *and satisfies* $w_r = w_{r \bmod N}$.

Teorema da convolução

Computing the convolution of two vectors in the time domain may look a bit complicated, but the frequency domain manifestation of convolution is very simple.

Theorem 4.2 ***The Convolution Theorem*** *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be vectors in* $\mathbb{C}^N$ *with DFT's* $\mathbf{X}$ *and* $\mathbf{Y}$, *respectively. Let* $\mathbf{w} = \mathbf{x} * \mathbf{y}$ *have DFT* $\mathbf{W}$. *Then*

$$W_k = X_k Y_k \tag{4.4}$$

for $0 \le k \le N-1$.

Teorema da convolução

$$W_k = \sum_{m=0}^{N-1} e^{-2\pi ikm/N} w_m.$$

Since $\mathbf{w} = \mathbf{x} * \mathbf{y}$, we have $w_m = \sum_{r=0}^{N-1} x_r y_{m-r}$. Substitute this into the formula for W_k above and interchange the summation order to find

$$W_k = \sum_{r=0}^{N-1} \sum_{m=0}^{N-1} e^{-2\pi ikm/N} x_r y_{m-r}.$$

Make a change of index in the m sum by substituting $n = m - r$ (so $m = n + r$). With the appropriate change in the summation limits and a bit of algebra, we obtain

$$W_k = \sum_{r=0}^{N-1} \sum_{n=-r}^{N-1-r} e^{-2\pi ik(n+r)/N} x_r y_n$$
$$= \left(\sum_{r=0}^{N-1} e^{-2\pi ikr/N} x_r \right) \left(\sum_{n=-r}^{N-1-r} e^{-2\pi ikn/N} y_n \right),$$
$$W_k = \left(\sum_{r=0}^{N-1} e^{-2\pi ikr/N} x_r \right) \left(\sum_{n=0}^{N-1} e^{-2\pi ikn/N} y_n \right)$$
$$= X_k Y_k.$$

Observações

H scales, and maybe phase-shifts, the input sinusoid S_i

In essence, we have now two alternative representations:

- determine the effect of L on s_i by convolution with h: $s_i * h$
- determine the effect of L on s_i by multiplication with H: $S_i \cdot H$

$$s_i * h \leftrightarrow S_i \cdot H$$

Since convolution is expensive for wide h, the multiplication may be cheaper

- but we need to perform the Fourier transforms of s_i and h

O processo de filtragem então baseia-se na confecção do vetor de amostras h, com DFT H

Design de filtro passa-baixa

- Design issue
  - G(u,v)=F(u,v) H(u,v)
  - Remove high freq. component (details, noise, …)
- Ideal low-pass filter
- Butterworth filter
- Gaussian filter

More smooth in the edge of cut-off frequency

Filtro passa-baixa ideal

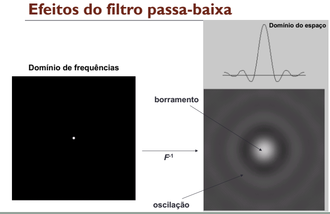
- Sharp cut-off frequency

$$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)\ge D_0\end{cases}$$

where D(u,v) is the distance to the center freq.

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2}$$

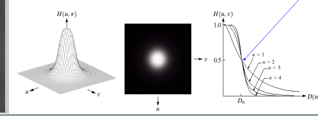
Efeitos do filtro passa-baixa



Filtros passa-baixa Butterworth

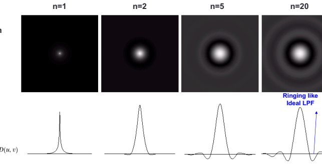
$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}}$$

H=0.5 when D(u,v)=D_0



Ordem do filtro Butterworth

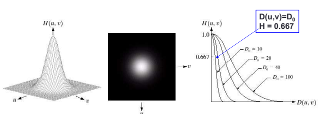
Spatial domain filter of butterworth filters



Filtro passa-baixa Gaussiano

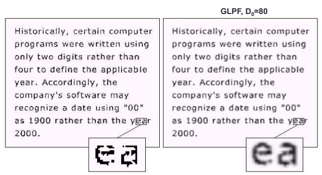
$$H(u,v)=e^{-D^2(u,v)/2D_0^2}$$

Variance or cut-off freq.



Aplicações práticas (i)

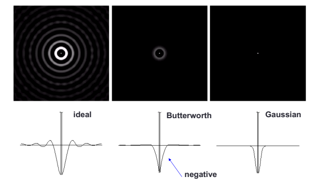
GLPF, D_0=80



Filtros passa-alta

- Image details corresponds to high-frequency
  - Sharpening: high-pass filters
  - $H_{hp}(u,v)=1-H_{lp}(u,v)$
- Ideal high-pass filters
- Butterworth high-pass filters
- Gaussian high-pass filters
- Difference filters
  - Laplacian filters

Filtros passa-alta (domínio do espaço)



Filtro Laplaciano

- Spatial-domain Laplacian (2nd derivative)

$$\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}$$

- Fourier transform

$$\Im\left[\frac{\partial^n f(x)}{\partial x^n}\right]=(ju)^n F(u)$$

$$\Im\left[\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2}\right]=(ju)^2F(u,v)+(jv)^2F(u,v)$$

$$=\boxed{-(u^2+v^2)}F(u,v)$$

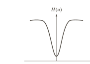
H(u,v) = -(u²+v²)

Diferença de Gaussianas (DoG)

Let $H(u)$ denote the difference of Gaussian filter

$$H(u)=Ae^{-u^2/2\sigma_1^2}-Be^{-u^2/2\sigma_2^2}$$

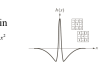
with $A\ge B$ and $\sigma_1\ge\sigma_2$



The corresponding filter in the spatial domain

$$h(x)=\sqrt{2\pi}\sigma_1 Ae^{-2\pi^2\sigma_1^2x^2}-\sqrt{2\pi}\sigma_2 Ae^{-2\pi^2\sigma_2^2x^2}$$

Approximates the Laplacian



Filtragem homomórfica

- Can be used to remove shading effects in an image (i.e., due to uneven illumination)
  - Enhance high frequencies
  - Attenuate low frequencies but preserve fine detail.

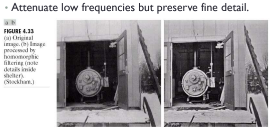


FIGURE 4.33 (a) Original image. (b) Image processed by homomorphic filtering (note details inside shelter). (Stockham.)

Filtragem homomórfica

- Consider the following model of image formation:

$$f(x,y)=i(x,y)\,r(x,y)$$

- Illumination $i(x,y)$: varies slowly and affects low frequencies mostly
- Reflection $r(x,y)$: varies faster and affects high frequencies mostly

Filtragem homomórfica

Mas supomos que:

$$z(x,y)=\ln(f(x,y))$$
$$=\ln(i(x,y))+\ln(r(x,y))$$

Então:

$$\Im(z(x,y))=\Im(\ln(f(x,y)))$$
$$=\Im(\ln(i(x,y)))+\Im(\ln(r(x,y)))$$

$$Z(u,v)=I(u,v)+R(u,v)$$

Filtragem homomórfica

Se processarmos $Z(u,v)$ com um filtro $H(u,v)$:

$$Z(u,v)=I(u,v)+R(u,v)$$

$$S(u,v)=H(u,v)Z(u,v)$$

$$S(u,v)=H(u,v)I(u,v)+H(u,v)R(u,v)$$

onde $S(u,v)$ é a transformada de Fourier do resultado

Filtragem homomórfica

No domínio espacial:

$$\Im^{-1}\{S(u,v)\}=s(x,y)$$

$$s(x,y)=\Im^{-1}\{H(u,v)I(u,v)\}+\Im^{-1}\{H(u,v)R(u,v)\}$$

supondo

$$i'(x,y)=\Im^{-1}\{H(u,v)I(u,v)\}$$

e

$$r'(x,y)=\Im^{-1}\{H(u,v)R(u,v)\}$$

$$s(x,y)=i'(x,y)+r'(x,y)$$

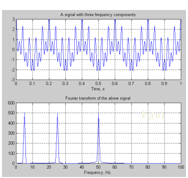
Filtragem homomórfica

Finalmente, uma vez que $z(x,y)$ foi construía como o logaritmo de $f(x,y)$, a inversa de $s(x,y)$ leva ao resultado desejado:

$$g(x,y)=e^{[s(x,y)]}$$
$$=e^{[i'(x,y)+r'(x,y)]}$$
$$=e^{i'(x,y)}e^{r'(x,y)}$$

Limitações

$$f_4(t)=\cos(2\pi\cdot 5\cdot t)+\cos(2\pi\cdot 25\cdot t)+\cos(2\pi\cdot 50\cdot t)$$



DFT provides excellent localization in the frequency domain but poor localization in the time domain.

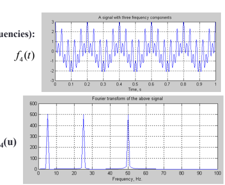
Sinais estacionários

Stationary signal (non-varying frequencies):

$f_4(t)$

Three frequency components, present at all times!

$F_4(u)$



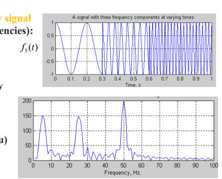
Sinais não-estacionários

Non-stationary signal (varying frequencies):

$f_5(t)$

Three frequency components, NOT present at all times!

$F_5(u)$



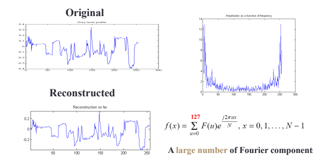
Limitações adicionais

- Not very useful for analyzing time-variant, non-stationary signals.
- Not efficient for representing discontinuities or sharp corners.

Reconstrução

Original

Reconstructed

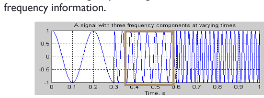


$$f(x)=\sum_{u=0}^{127}F(u)e^{\frac{j2\pi ux}{N}},\ x=0,1,\dots,N-1$$

A large number of Fourier components is needed to represent discontinuities.

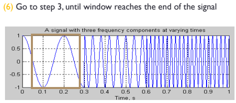
Janelamento

- Segment signal into narrow time intervals (i.e., narrow enough to be considered stationary) and take the FT of each segment.
- Each FT provides the spectral information of a separate time-slice of the signal, providing simultaneous time and frequency information.



Passos na versão janelada

(1) Choose a window of finite length
(2) Place the window on top of the signal at t=0
(3) Truncate the signal using this window
(4) Compute the FT of the truncated signal, save results.
(5) Incrementally slide the window to the right
(6) Go to step 3, until window reaches the end of the signal



Short Time Fourier Transform (STFT)

Time parameter, Frequency parameter, Signal to be analyzed

$$STFT_f^u(t',u)=\int_t\left[f(t)\cdot W(t-t')\right]\cdot e^{-j2\pi ut}dt$$

2D function

STFT of f(t): computed for each window centered at t=t'

Windowing function

Centered at t=t'

Tamanho da janela

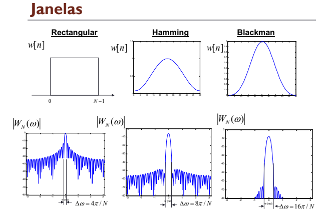
$$STFT_f^u(t',u)=\int_t\left[f(t)\cdot W(t-t')\right]\cdot e^{-j2\pi ut}dt$$

W(t) infinitely long: STFT turns into FT, providing excellent frequency localization, but no time localization

W(t) infinitely short: results in the time signal (with a phase factor), providing excellent time localization but no frequency localization

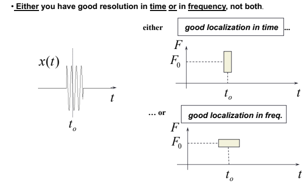
$$STFT_f^u(t',u)=\int_t\left[f(t)\cdot\delta(t-t')\right]\cdot e^{-j2\pi ut}dt=f(t')\cdot e^{-jut'}$$

Janelas



Princípio da incerteza

- Either you have good resolution in time or in frequency, not both.

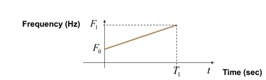


Chirp

A "Chirp" is a sinusoid with time varying frequency, with expression

$$x(t)=A\cos(2\pi F(t)t+\alpha)$$

If the frequency changes linearly with time, it has the form shown below:



Separação de frequências

Since this signal contains music we expect to distinguish between musical notes. These are the frequencies associated to it (rounded to closest integer):

Notes	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
Freq. (Hz)	262	277	294	311	330	349	370	392	415	440	466	494

Desired Frequency Resolution*:** $\Delta F \approx 2\frac{F_S}{N} \leq 15Hz$

This yields a window length of at least *N*=1024