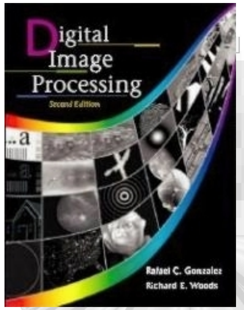


Aula 7

Domínio da Frequência
Outras Transformadas

Outras transformadas

A transformada discreta de Fourier unidimensional é uma das classes de transformadas importantes, que podem ser expressas em termos da relação geral

$$T(u) = \sum_{x=0}^{N-1} f(x)g(x,u) \qquad (3.5.1)$$

Como é o caso da Transformada de Laplace

$$\mathcal{L}\{f(t)\} = \int_{-\infty}^{\infty} f(t)\, e^{-st}\, \mathrm{d}t$$

Outras transformadas

3.5.1 Transformada de Walsh

Quando $N = 2^n$, a transformada discreta de Walsh de uma função $f(x)$, denotada por $W(u)$, é obtida pela substituição do núcleo

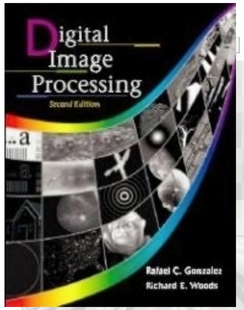
$$g(x,u) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)} \qquad \textbf{(3.5-14)}$$

na Equação (3.5-1). Em outras palavras,

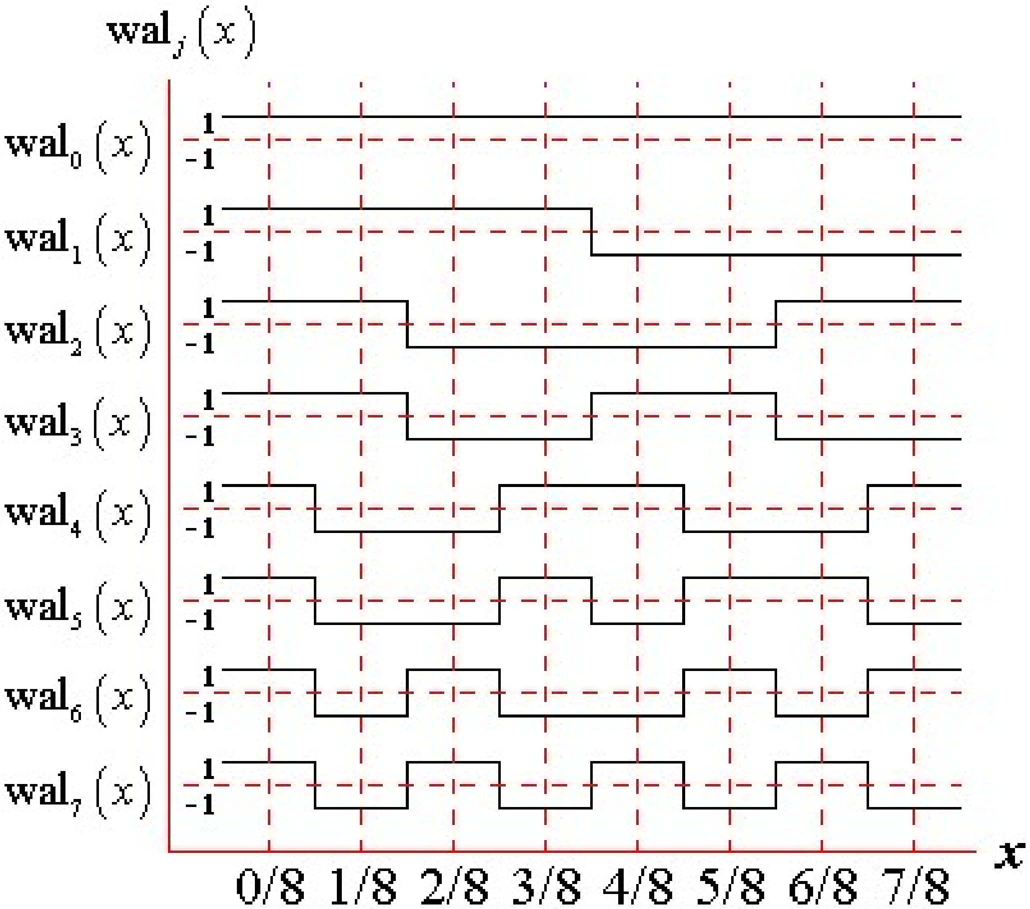
$$W(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)} \qquad \textbf{(3.5-15)}$$

em que $b_k(z)$ é o k-ésimo bit na representação binária de z. $b_0(z) = 0$, $b_1(z) = 1$ e $b_2(z) = 1$.

Por exemplo, se $k = 3$ e $z = 6$ (110 em binário) $b_0(z) = 0$, $b_1(z) = 1$ e $b_2(z) = 1$.



Funções de Walsh



Outras transformadas

a transformada inversa de Walsh é

i-ésimo bit

$$f(x) = \sum_{u=0}^{N-1} W(u) \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)}$$

2D

$$W(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \prod_{i=0}^{n-1} (-1)^{[b_i(x) b_{n-1-i}(u) + b_i(y) b_{n-1-i}(v)]}$$

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} W(u,v) \prod_{i=0}^{n-1} (-1)^{[b_i(x) b_{n-1-i}(u) + b_i(y) b_{n-1-i}(v)]}$$

Como mencionado anteriormente, um algoritmo usado para computar a FFT pelo método dos dobramentos sucessivos pode ser facilmente modificado para computar uma transformada rápida de Walsh

Exemplo: Se N = 4

N=4
n=2
pois $N=2^n$

$$W(0) = \frac{1}{4}\sum_{x=0}^{3}\left[f(x)\prod_{i=0}^{1}(-1)^{b_i(x)b_{1-i}(0)}\right]$$

$$= \frac{1}{4}[f(0) + f(1) + f(2) + f(3)]$$

$$W(1) = \frac{1}{4}\sum_{x=0}^{3}\left[f(x)\prod_{i=0}^{1}(-1)^{b_i(x)b_{1-i}(1)}\right]$$

$$= \frac{1}{4}[f(0) + f(1) - f(2) - f(3)]$$

$$W(2) = \frac{1}{4}\sum_{x=0}^{3}\left[f(x)\prod_{i=0}^{1}(-1)^{b_i(x)b_{1-i}(2)}\right]$$

$$= \frac{1}{4}[f(0) - f(1) + f(2) - f(3)]$$

$$W(3) = \frac{1}{4}\sum_{x=0}^{3}\left[f(x)\prod_{i=0}^{1}(-1)^{b_i(x)b_{1-i}(3)}\right]$$

$$= \frac{1}{4}[f(0) - f(1) - f(2) + f(3)].$$

Prática

Calcular a transformada de Walsh para

f(0), f(1), f(2), f(3)

N=4

n=2

pois $N=2^n$

obter W(0), W(1), W(2), W(3)

Outras transformadas

3.5.2 A Transformada de Hadamard

Uma das várias formulações conhecidas para o núcleo de Hadamard direto 1-D é a relação

$$g(x,u) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)} \tag{3.5-25}$$

em que o somatório no expoente é executado através de aritmética binária e, como na Equação (3.5-14), $b_k(z)$ é o k-ésimo bit na representação binária de z. A substituição da Equação (3.5-25) na Equação (3.5-1) produz a seguinte expressão para a transformada de Hadamard unidimensional:

$$H(u) = \frac{1}{N}\sum_{i=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)} \tag{3.5-26}$$

em que $N = 2^n$, e u assume valores em $0, 1, 2, \ldots, N-1$.

Outras transformadas

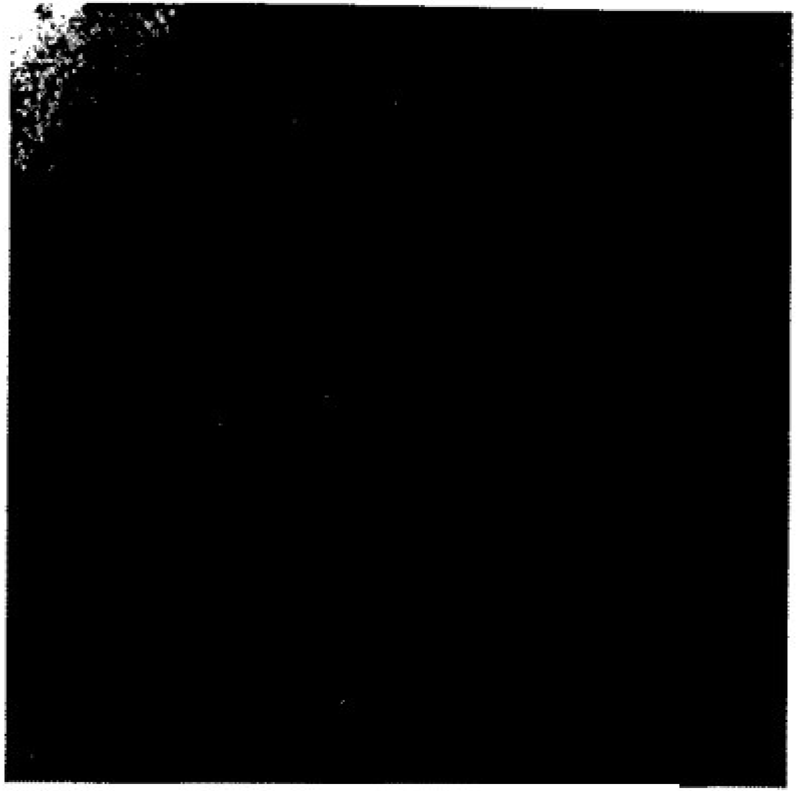
Hadamard - 2D

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]}$$

$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(u,v)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]}$$

(a)

(b)



Uma imagem simples e a magnitude logarítmica de sua transformada de Hadamard.

Outras transformadas

3.5.3 A Transformada cosseno discreta

A *transformada cosseno discreta* unidimensional (DCT - "Discrete Cosine Transform") é definida como

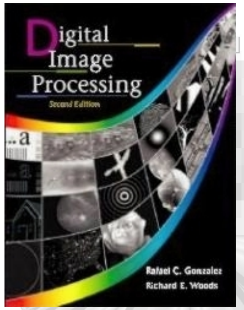
$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad (3.5\text{-}45)$$

para $u = 0, 1, 2, \ldots, N-1$. Do mesmo modo, a DCT inversa é definida como

$$f(x) = \sum_{u=0}^{N-1} \alpha(u)C(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad (3.5\text{-}46)$$

para $x = 0, 1, 2, \ldots, N-1$. Em ambas as Equações (3.5-45) e (3.5-46), a é

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{N}} & \text{para } u = 0 \\ \sqrt{\dfrac{2}{N}} & \text{para } u = 1, 2, \ldots, N-1. \end{cases} \qquad (3.5\text{-}47)$$



Outras transformadas

2D

O par DCT correspondente é

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \qquad (3.5\text{-}48)$$

para u, $v = 0, 1, 2, \ldots, N-1$, e

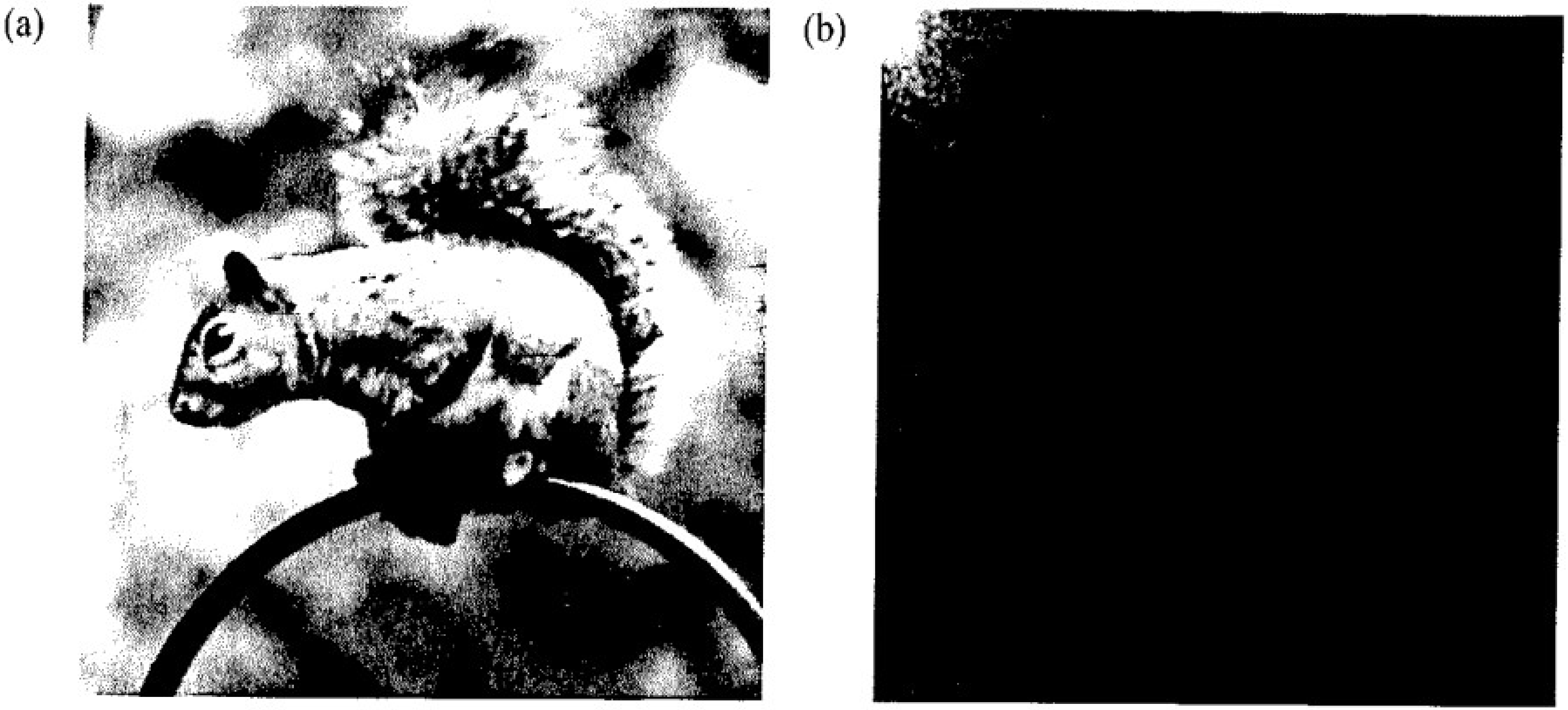
$$f(x,y) = \sum_{u=0}^{N1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \qquad (3.5\text{-}49)$$

para x, $y = 0, 1, 2, \ldots, N-1$, em que a é dado na Equação (3.5-47).

assunto a ser estudado na aula 9

Em anos recentes a transformada cosseno discreta tem se tornado um método freqüentemente escolhido para compressão de imagens

o padrão JPEG usa esta transformada para obter uma redução significativa das imagens, sem perder muito a qualidade.



Uma imagem simples e a magnitude logarítmica de sua transformada de cosseno discreta.

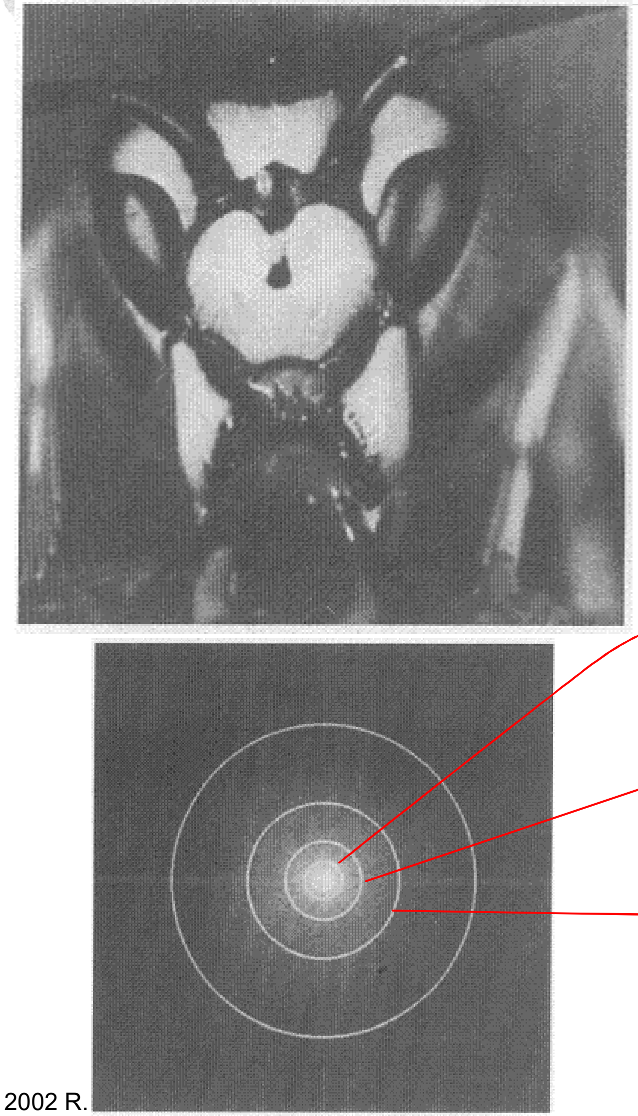
Prática 1

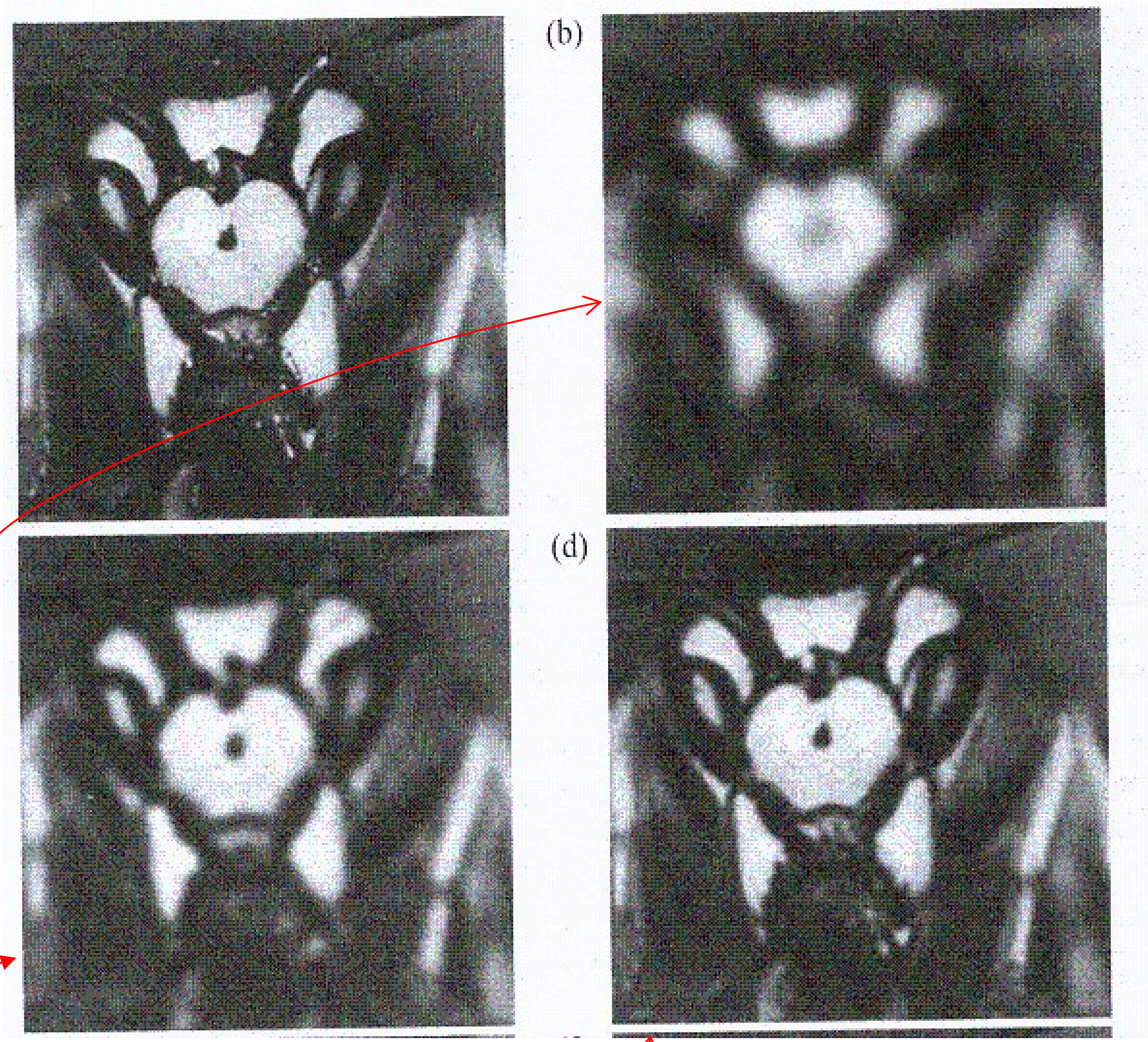
1. Implemente a transformada discreta do cosseno
2. Use uma imagem 128 x 128
3. Grave o resultado em uma matriz C[128][128]
4. Exiba na image2
5. Implemente a inversa da DCT
6. Grave o resultado em uma matriz f[128][128]
7. Exiba na image2

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right), \text{ para}$$

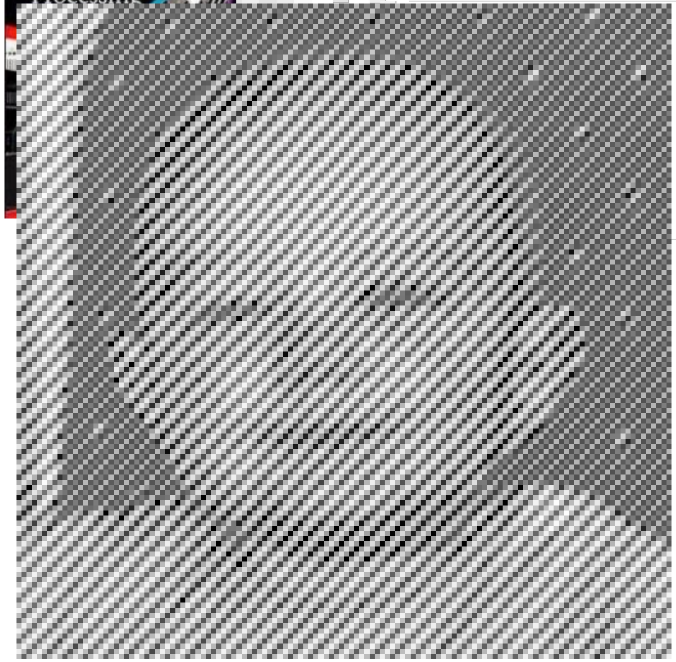
$$0 \leq i, j \leq n-1$$

$$\text{onde } C_{i,j} = \begin{cases} \frac{1}{\sqrt{n}}, & i, j = 0 \\ \sqrt{2/n}, & i, j > 0 \end{cases}$$



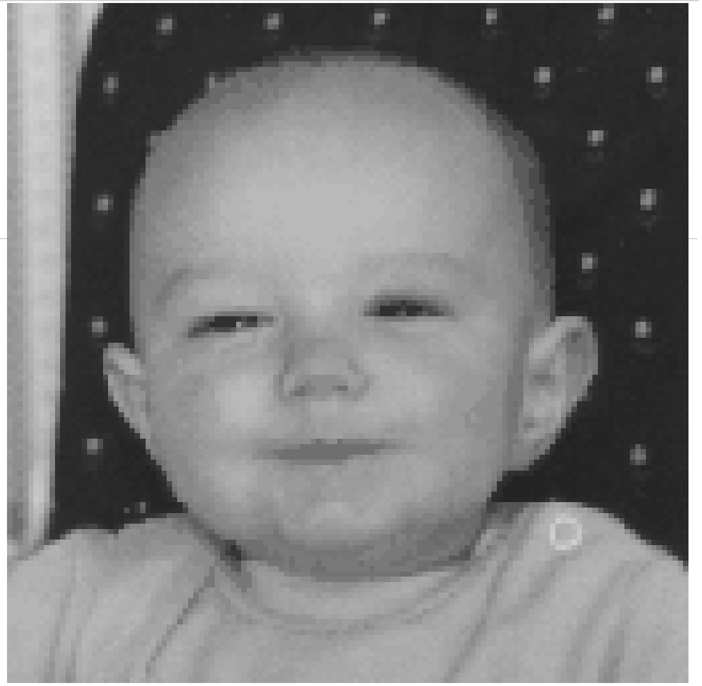


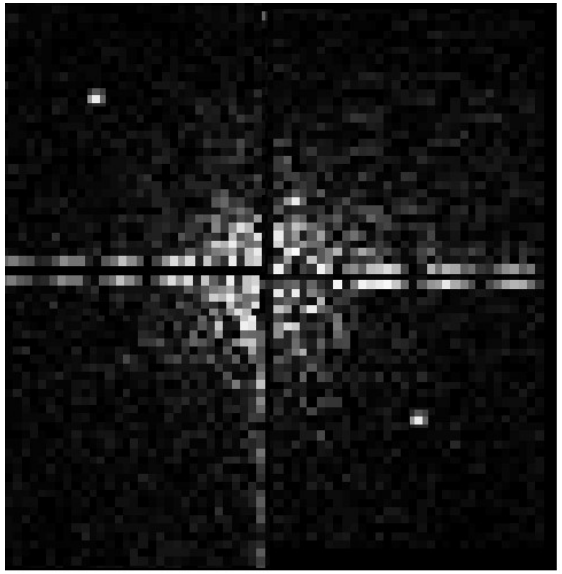
Filtragem no domínio da freqüência
Utiliza freqüências de corte

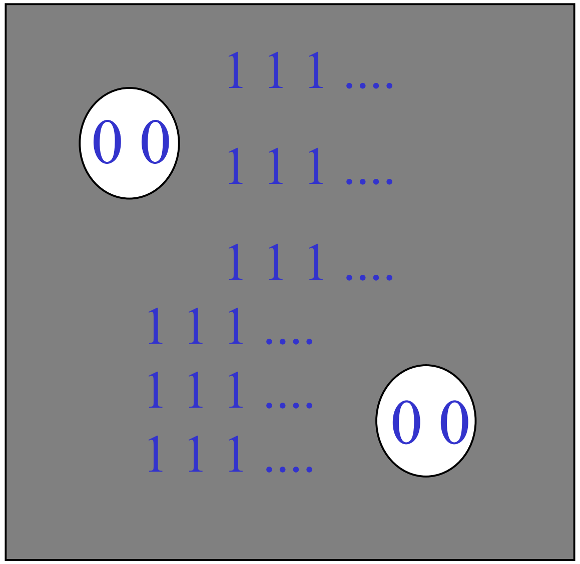


Redução de ruído periódico

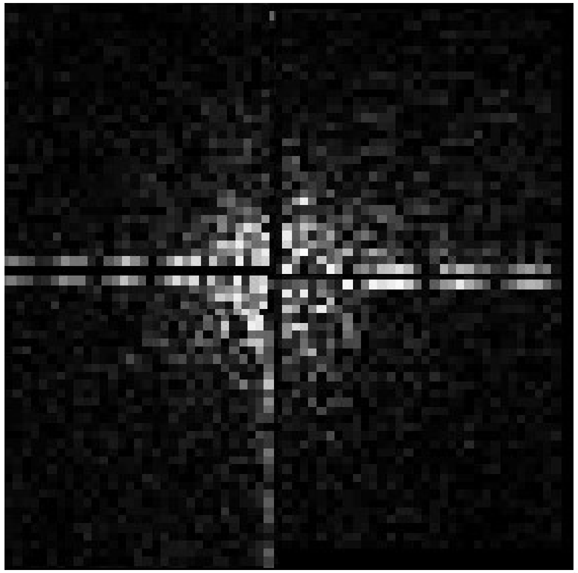
Fourier



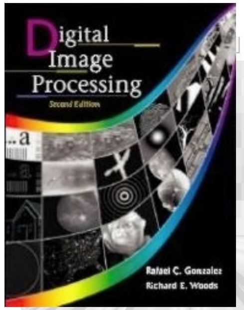




=

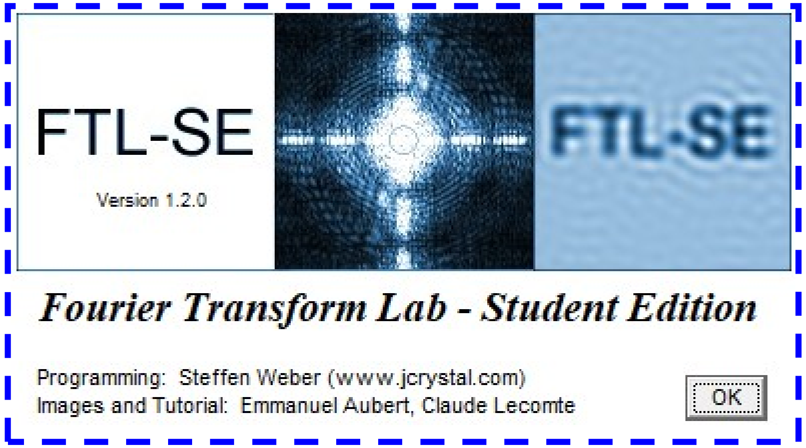


Produto



Demonstração

Usar o programa FTL-SE

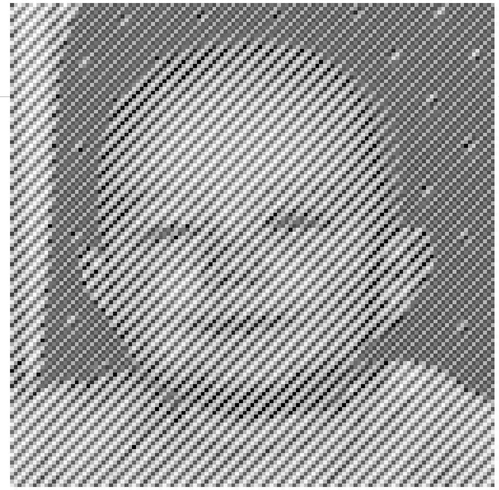


Operação → FFT

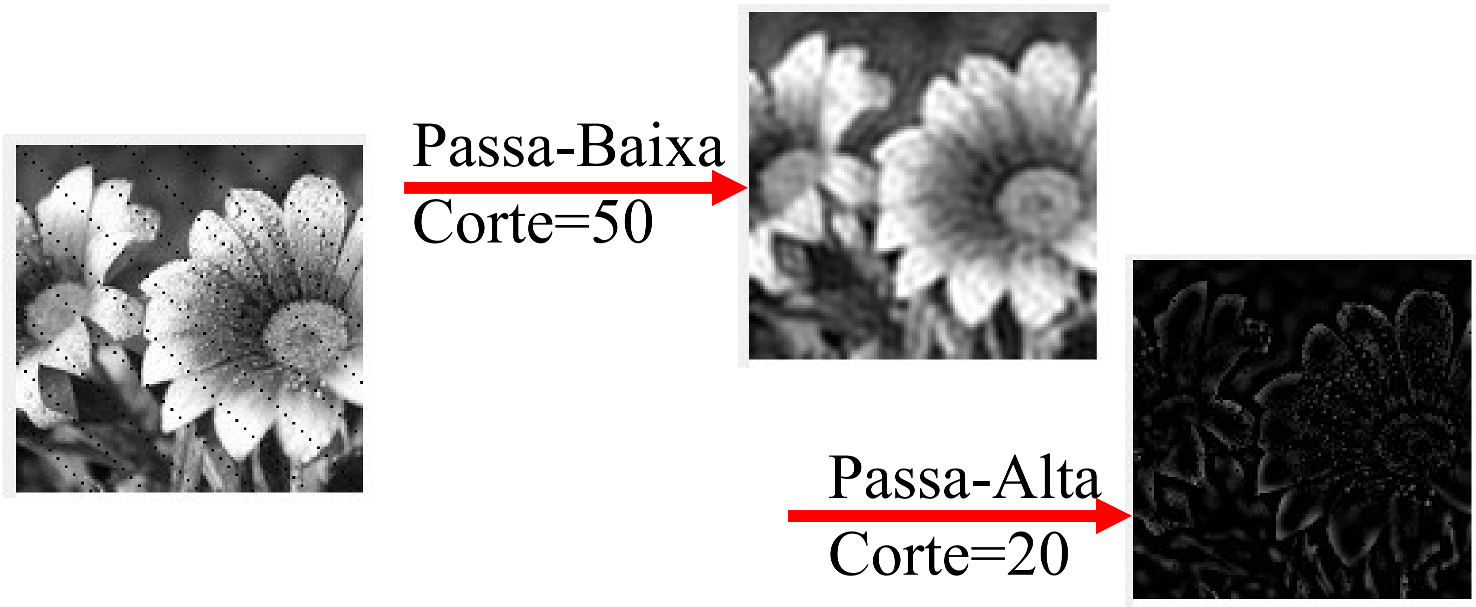
→ Custom Masks

Eliminar as componentes de grande amplitude de alta frequência

Prática 2



1. Aplique os filtros passa-baixa e passa-alta usando a transformada do cosseno na Imagem
2. O usuário define a frequência de corte
3. Veja o resultado no domínio do espaço

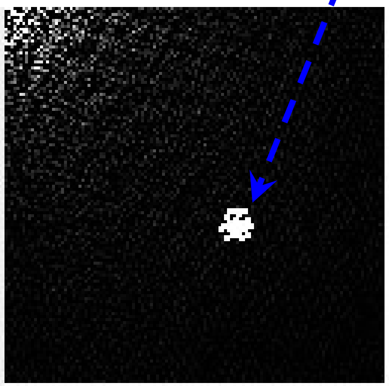


4. O usuário define um ruído no domínio da frequência e depois visualiza o resultado no domínio do espaço

(Clicando na imagem, insira um valor 255 no local)

DCT

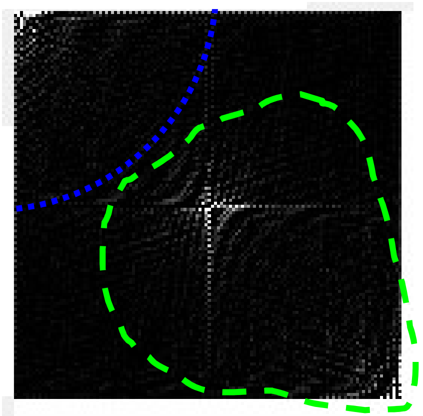
Insere Ruído



IDCT

5. Aplique o passa-baixa na imagem do bebê
Com corte = 77

DCT



Ruído

Passa-Baixa
Corte=77

IDCT

```cpp
// CPP program to perform discrete cosine transform
#include <bits/stdc++.h>
using namespace std;
#define pi 3.142857
const int m = 8, n = 8;

// Function to find discrete cosine transform and print it
int dctTransform(int matrix[][n])
{
    int i, j, k, l;

    // dct will store the discrete cosine transform
    float dct[m][n];

    float ci, cj, dct1, sum;

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {

            // ci and cj depends on frequency as well as
            // number of row and columns of specified matrix
            if (i == 0)
                ci = 1 / sqrt(m);
            else
                ci = sqrt(2) / sqrt(m);
            if (j == 0)
                cj = 1 / sqrt(n);
            else
                cj = sqrt(2) / sqrt(n);

            // sum will temporarily store the sum of
            // cosine signals
            sum = 0;
            for (k = 0; k < m; k++) {
                for (l = 0; l < n; l++) {
                    dct1 = matrix[k][l] *
                           cos((2 * k + 1) * i * pi / (2 * m)) *
                           cos((2 * l + 1) * j * pi / (2 * n));
                    sum = sum + dct1;
                }
            }
            dct[i][j] = ci * cj * sum;
        }
    }

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("%f\t", dct[i][j]);
        }
        printf("\n");
    }
}

// Driver code
int main()
{
    int matrix[m][n] = { { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
                         { 255, 255, 255, 255, 255, 255, 255, 255 }
    dctTransform(matrix);
    return 0;
}
```

Entrada

255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

Saída

2039.999878	-1.168211	1.190998	-1.230618	1.289227	-1.370580	1.480267	-1.626942
-1.167731	0.000664	-0.000694	0.000698	-0.000748	0.000774	-0.000837	0.000920
1.191004	-0.000694	0.000710	-0.000710	0.000751	-0.000801	0.000864	-0.000950
-1.230645	0.000687	-0.000721	0.000744	-0.000771	0.000837	-0.000891	0.000975
1.289146	-0.000751	0.000740	-0.000767	0.000824	-0.000864	0.000946	-0.001026
-1.370624	0.000744	-0.000820	0.000834	-0.000858	0.000898	-0.000998	0.001093
1.480278	-0.000856	0.000870	-0.000895	0.000944	-0.001000	0.001080	-0.001177
-1.626932	0.000933	-0.000940	0.000975	-0.001024	0.001089	-0.001175	0.001298