

# Notas de aula de Processamento de imagens e Visão Computacional

Daniel Oliveira Dantas

24 de março de 2020

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	O que é processamento de imagens . . . . .	1
1.2	Origens do processamento digital de imagens . . . . .	2
1.3	Áreas que usam processamento digital de imagens . . . . .	2
1.4	Passos fundamentais no processamento digital de imagens . . . . .	2
1.5	Componentes de um sistema de processamento de imagens . . . . .	3
<b>2</b>	<b>Fundamentos de imagens digitais</b>	<b>4</b>
2.1	Elementos de percepção visual . . . . .	4
2.2	Luz e o espectro eletromagnético . . . . .	4
2.3	Aquisição de imagens . . . . .	5
2.4	Amostragem e quantização . . . . .	5
2.5	Relações entre pixels . . . . .	6
2.6	Ferramentas matemáticas . . . . .	7
<b>3</b>	<b><i>Image Enhancement</i> (melhoramento) no domínio do espaço</b>	<b>8</b>
3.1	Introdução . . . . .	8
3.2	<i>Gray-level transformations</i> simples . . . . .	9
3.3	Processamento de histograma . . . . .	11
3.4	Operações aritméticas e lógicas . . . . .	12
3.5	Filtragem espacial . . . . .	12
3.6	Suavização . . . . .	12
3.7	<i>Sharpening</i> . . . . .	13
<b>4</b>	<b><i>Image Enhancement</i> (melhoramento) no domínio da frequência</b>	<b>15</b>
4.1	Introdução . . . . .	15
4.2	A transformada de Fourier . . . . .	15
4.3	<i>Fast Fourier Transform</i> . . . . .	17

# Capítulo 1

## Introdução

Capítulo 1 de Gonzalez, *Digital Image Processing* [7].

### 1.1 O que é processamento de imagens

- O que é uma imagem?
  - Uma matriz
  - Uma função
  - Esses conceitos podem ser generalizados para mais dimensões
- Uma definição de processamento de imagens: processo em que tanto a entrada quanto a saída são imagens.
  - Essa definição não engloba, porém, processos como valor médio da imagem, extração de pontos característicos, alguns tipos de reconstrução 3D, aplicações de segurança que detectam atividades suspeitas, reconhecimento de gestos, reconhecimento de caracteres (OCR) e outras aplicações consideradas do campo de visão computacional.
- Uma definição mais abrangente: processos de baixo, médio e alto nível em que tanto a entrada quanto a saída são imagens.
  - Baixo nível: envolve operações primitivas, de redução de ruído, aumento de contraste, aumento de nitidez (*sharpening*), *thresholding* etc. Nos processos de baixo nível, tanto a entrada quanto a saída são imagens.
  - Médio nível: envolve tarefas como segmentação (particionamento), redução dos objetos a uma descrição ou formato apropriado para processamento, classificação (reconhecimento) de objetos. A entrada são imagens, e a saída são atributos extraídos dessas imagens, como bordas, contornos, características ou classe de objetos individuais.
  - Alto nível: envolve atividades cognitivas associadas com a visão. Cognição envolve atividades como memória, compreensão, aprendizado, raciocínio, atenção, resolução de problemas e tomada de decisão.

## 1.2 Origens do processamento digital de imagens

- 1920: *Bartlane cable picture transmission system*, 5 tons de cinza.
- 1929: idem, 15 tons de cinza.
- 1948: invenção do transistor.
- 1950 a 1960: invenção das linguagens de programação de alto nível, COBOL e Fortran.
- 1958: invenção do circuito integrado.
- 1964: primeira foto da Lua tirada de uma sonda.
- 1968 a 1971: invenção dos primeiros microprocessadores, CADC, TMS1000, 4004.
- 1969: invenção do CCD.
- 1971: primeira tomografia computadorizada

## 1.3 Áreas que usam processamento digital de imagens

- Medicina, astronomia, meteorologia, indústria, fotografia, editoração, segurança etc.
- Técnicas de obtenção de imagem:
  - Eletromagnética: luz, radiação UV, radiação IR, raios X, raios gama, microondas.
  - Eletrônica: microscopia eletrônica.
  - Mecânica: ondas acústicas, ultrassom.
  - Sintética: computação gráfica, fractais.

## 1.4 Passos fundamentais no processamento digital de imagens

- Aquisição
- Melhoramento (*image enhancement*)
- Restauração
- Processamento de cores
- *Wavelets* e processamento multirresolução
- Compressão
- Processamento morfológico
- Segmentação
- Representação e descrição
- Reconhecimento de objetos

## 1.5 Componentes de um sistema de processamento de imagens

- Sensores + *hardware* especializado
- Computador + GPU
- Armazenamento em massa
- *Software* de processamento de imagens
- Monitor de imagem
- Impressora
- Rede

## Capítulo 2

# Fundamentos de imagens digitais

Capítulo 2 de Gonzalez, *Digital Image Processing* [7].

### 2.1 Elementos de percepção visual

### 2.2 Luz e o espectro eletromagnético

- Frequência:  $f$
- Comprimento de onda:  $\lambda$
- Velocidade da luz:  $c = 3 \times 10^8 \text{ m/s}$
- Constante de Planck:  $h = 6.6 \times 10^{-34} \text{ Js}$  ( $\text{m}^2\text{kg/s}$ ),  $E = hf$ .

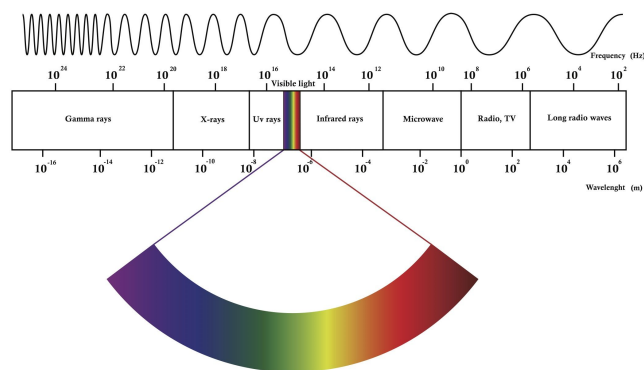


Figura 2.1: Espectro eletromagnético

## 2.3 Aquisição de imagens

- Normalmente é feita através de componentes sensíveis a alguma faixa específica do espectro eletromagnético
  - Sensores simples: fotodiodo, fototransistor.
  - Sensores lineares: CCD linear.
  - Sensores em matriz: CCD de câmeras fotográficas, mouse ótico.

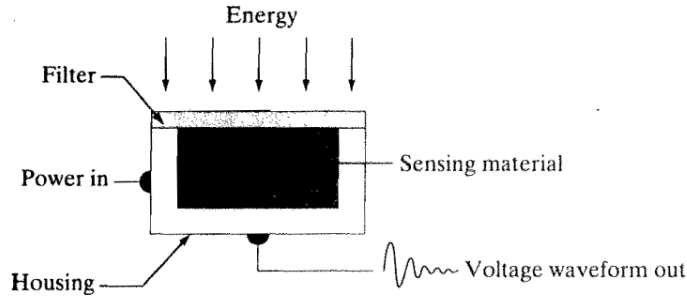


Figura 2.2: Esquema de um sensor

## 2.4 Amostragem e quantização

- Amostragem: digitalização dos valores das coordenadas, tanto no espaço quanto no tempo.
- Quantização: digitalização dos valores das amplitudes
- Representação:
  - Matriz

$$f(i, j) = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \dots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1, 0) & f(M-1, 1) & \dots & f(M-1, N-1) \end{bmatrix}$$

- Função

$$f : \mathbb{Z}^2 \rightarrow \mathbb{Z}$$

$$f : \{0, \dots, M-1\} \times \{0, \dots, N-1\} \rightarrow \{0, \dots, L-1\}$$

onde  $L = 2^k$ .  $L$  é o intervalo dinâmico do sensor e  $k$  é o número de bits necessário para representar  $L$ .

- Número de bits necessários para armazenar uma imagem  $M \times N$ :

$$b = MNk$$

- Número de bytes necessários para armazenar uma imagem  $M \times N$ :

$$B = MNk/8$$

- Redimensionamento de imagem: *zoom* ou *resize*
  - *Nearest neighbor*: ampliar uma imagem replicando cada pixel várias vezes.
  - *Bilinear*: ampliar uma imagem inserindo pixels calculados da interpolação linear entre os pixels mais próximos.

## 2.5 Relações entre pixels

- 4-adjacência: os vizinhos de  $(x, y)$  são  $(x, y - 1)$ ,  $(x, y + 1)$ ,  $(x - 1, y)$  e  $(x + 1, y)$ .
- D-adjacência: os vizinhos de  $(x, y)$  são  $(x - 1, y - 1)$ ,  $(x - 1, y + 1)$ ,  $(x + 1, y - 1)$  e  $(x + 1, y + 1)$ .
- 8-adjacência: os vizinhos de  $(x, y)$  são a união da 4-adjacência e da D-adjacência.

$(x - 1, y - 1)$	$(x, y - 1)$	$(x + 1, y - 1)$
$(x - 1, y)$	$(x, y)$	$(x + 1, y)$
$(x - 1, y + 1)$	$(x, y + 1)$	$(x + 1, y + 1)$

Tabela 2.1: Disposição dos pixels na vizinhança de  $(x, y)$ .

- Distância: é uma relação entre dois pixels  $p$  e  $q$  com as propriedades listadas abaixo.

1.  $D(p, q) \geq 0$
2.  $D(p, q) = 0 \Leftrightarrow p = q$
3.  $D(p, q) = D(q, p)$  simetria
4.  $D(p, q) \leq D(p, z) + D(z, q)$  desigualdade triangular

- Distância Euclidiana:  $D_e(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$
- Distância *city-block*:  $D_4(p, q) = |p_x - q_x| + |p_y - q_y|$
- Distância *chessboard*:  $D_8(p, q) = \max(|p_x - q_x|, |p_y - q_y|)$



## 2.6 Ferramentas matemáticas

- Operações lineares: seja  $H$  uma operação cuja entrada e saída são imagens, sejam  $f$  e  $g$  duas imagens, e  $a$  e  $b$ , dois escalares. A operação  $H$  é dita linear se segue a relação abaixo

$$H(af + bg) = aH(f) + bH(g)$$

- Operações não-lineares: uma operação é dita não-linear se não é linear.

## Capítulo 3

# *Image Enhancement* (melhoramento) no domínio do espaço

Capítulo 3 de Gonzalez, *Digital Image Processing* [7].

### 3.1 Introdução

- Métodos que operam no domínio do espaço podem ser denotados pela fórmula

$$g(x, y) = T(f(x, y))$$

onde  $g(x, y)$  é a imagem de saída,  $f(x, y)$  é a imagem de entrada, e  $T$  é uma operação sobre  $f$ , definida sobre alguma vizinhança de  $(x, y)$ .  $T$  pode operar sobre uma imagem ou um conjunto de imagens de entrada.

- A forma mais simples de  $T$  é quando a vizinhança tem tamanho  $1 \times 1$ , ou seja, um único pixel. Nesse caso,  $T$  é chamado *gray-level transformation function*, ou *intensity* ou *mapping transformation function*, da forma

$$s = T(r)$$

onde  $s$  denota a intensidade de  $g(x, y)$ , e  $r$ , a intensidade de  $f(x, y)$  em qualquer ponto  $(x, y)$ .

- Um exemplo de transformação para aumentar o contraste da imagem de entrada pode ser visto na Figura 3.1a.
- *Thresholding*: é como a transformação anterior levada ao extremo. Também aumenta o contraste, e a imagem de saída fica com apenas duas intensidades. Um exemplo dessa transformação pode ser visto na Figura 3.1b.

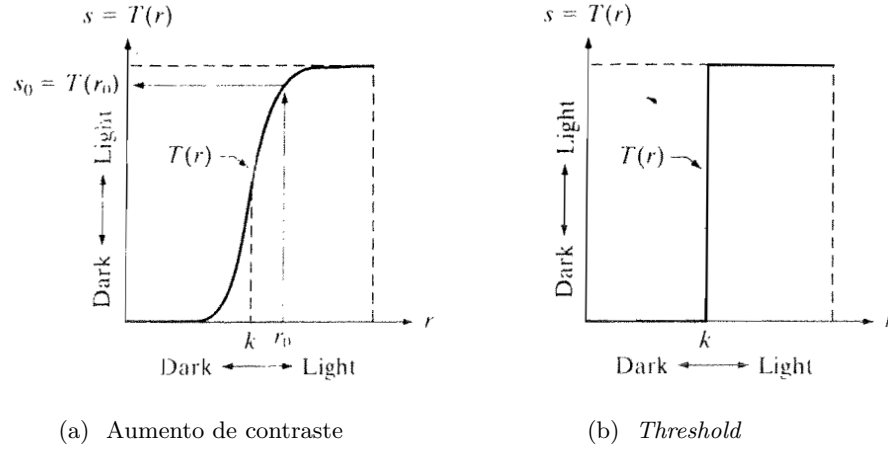


Figura 3.1: *Gray-level transformations*

### 3.2 *Gray-level transformations* simples

— Negação: ver Figura 3.2.

$$s = L - 1 - r$$

— Transformações logarítmicas: ver Figura 3.2.

$$s = c \log(1 + r)$$

— Transformações por potenciação: ver Figura 3.3.

$$s = cr^\gamma$$

— *Piecewise linear transformations*: ver Figura 3.4.

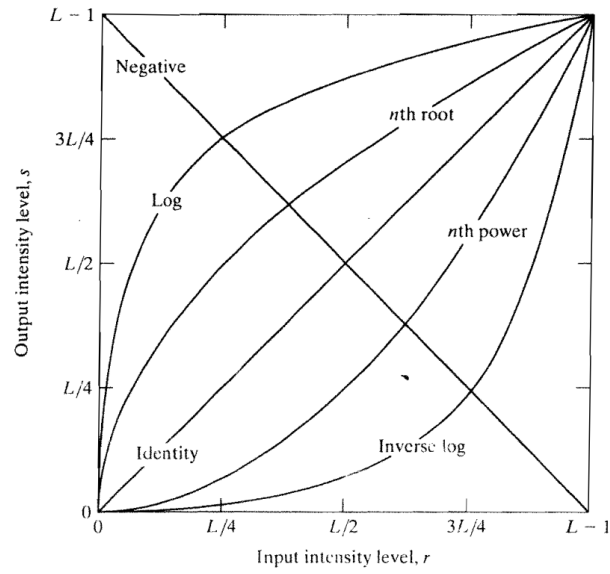


Figura 3.2: Exemplos de *gray-level transformations* simples.

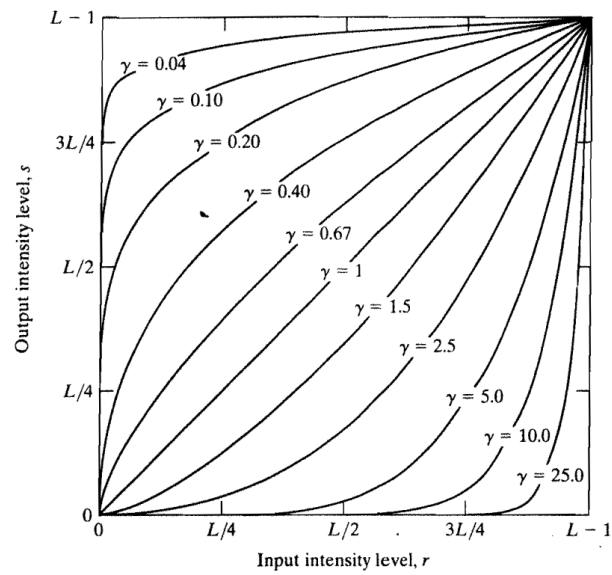


Figura 3.3: Transformações por potenciação com diferentes expoentes

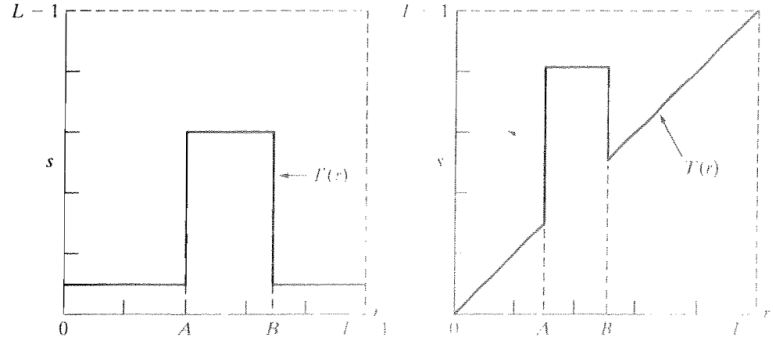


Figura 3.4: *Piecewise linear transformations*

### 3.3 Processamento de histograma

- Histograma: de uma imagem digital com tons de cinza no intervalo  $[0, L-1]$  é uma função discreta  $h(r_k) = n_k$ , onde  $k$  é o  $k$ -ésimo tom de cinza,  $r_k = k/(L-1)$  e  $n_k$  é o número de pixels na imagem com tom de cinza igual a  $k$ .
- Histograma normalizado: é dado por  $p(r_k) = n_k/n$ , onde  $n$  é o total de pixels da imagem.
- Equalização de histograma: é uma transformação que distribui uniformemente as escalas de cinza pelo intervalo dinâmico.

$$s = T(r), 0 \leq r \leq 1$$

- $T(r)$  tem valor único e é monotonicamente crescente no intervalo  $0 \leq r \leq 1$ .
- $0 \leq T(r) \leq 1$  para  $0 \leq r \leq 1$ .

A probabilidade de ocorrência de um tom de cinza na imagem é

$$p(r_k) = n_k/n, k \in [0, L-1]$$

$$T(r_k) = \sum_{j=0}^k p(r_j) = \sum_{j=0}^k \frac{n_j}{n}$$

### 3.4 Operações aritméticas e lógicas

- Interseção ou *AND* lógico ou mínimo entre duas imagens: útil para aplicar máscaras.
- Subtração: útil em imagens médicas com contraste radioativo.
- Média: útil para diminuir o ruído quando é possível tirar várias fotos de um mesmo ponto de vista. Aplicada em astronomia.

### 3.5 Filtragem espacial

- Filtragem espacial linear: é dada pela soma de pixels multiplicados por pesos
  - Correlação:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

onde  $g$  é a imagem de saída,  $(2a + 1)(2b + 1)$  é o tamanho do filtro, máscara, *kernel* ou janela, e  $f$  é a imagem de entrada.

- Convolução:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x - s, y - t)$$

- Filtragem espacial não-linear: mediana, variância, filtros morfológicos dentre outros.

### 3.6 Suavização

- Filtro de média:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Filtro Gaussiano ou suavização Gaussiana ou *blur*:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

- Filtro de estatísticas de ordem: mediana, máximo, mínimo etc.

### 3.7 *Sharpening*

— Derivada de primeira ordem:

$$\frac{\partial f}{\partial x} = \frac{df}{dx} = f(x+1) - f(x)$$

— Derivada de segunda ordem:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

— Laplaciano:

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

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

— *Sharpening*:

$$g(x, y) = f(x, y) - \nabla^2 f$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

— *Unsharp mask*:

$$g(x, y) = (1 + \alpha)f(x, y) - \alpha\bar{f}(x, y), \alpha \in [0.0, 1.0]$$

onde  $\bar{f}(x, y)$  denota a imagem  $f(x, y)$  após uma operação de *blur*.

— O uso do gradiente:

- Gradiente de um campo escalar é um campo vetorial que aponta para a direção de sua maior taxa de crescimento. Sua magnitude é a taxa de crescimento.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- A magnitude do gradiente é dada por

$$\text{mag}(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

mas na prática, para evitar os quadrados e a raiz quadrada, se usa apenas

$$\text{mag}(\nabla f) = \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right|$$

- Considere a vizinhança a seguir:

$p_1$	$p_2$	$p_3$
$p_4$	$p_5$	$p_6$
$p_7$	$p_8$	$p_9$

O exmplo mais simples de cálculo do gradiente é

$$\text{mag}(\nabla f) = |p_6 - p_5| + |p_8 - p_5|$$

- *Robert's cross gradient operator:*

$$\text{mag}(\nabla f) = \sqrt{(p_9 - p_5)^2 + (p_8 - p_6)^2}$$

ou, para simplificar,

$$\text{mag}(\nabla f) = |p_9 - p_5| + |p_8 - p_6|$$



## Capítulo 4

# *Image Enhancement* (melhoramento) no domínio da frequência

Capítulo 4 de Gonzalez, *Digital Image Processing* [7].  
Seção 7 de Ramirez, *The FFT: Fundamentals and Concepts* [10].

### 4.1 Introdução

- Transformada de Fourier é uma maneira de representar um sinal como uma integral de senos e cossenos multiplicados por uma função de peso. Um sinal pode ser convertido para sua representação transformada e reconvertido de volta para o domínio original sem perda de informação. Também é possível realizar operações como filtragens na representação transformada e reconverter de volta para o domínio original.

### 4.2 A transformada de Fourier

- Transformada de Fourier  $g(u)$  de uma função contínua de uma única variável é dada pela fórmula

$$g(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux} dx$$

onde  $j = \sqrt{-1}$ , De maneira análoga, dado  $g(u)$ , podemos obter  $f(x)$  através da transformada inversa de Fourier

$$f(x) = \int_{-\infty}^{\infty} g(u)e^{j2\pi ux} du$$

— Essas operações possuem uma versão para duas variáveis

$$g(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

e sua inversa

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(u, v) e^{j2\pi(ux+vy)} dx dy$$

— Muitas vezes é mais fácil manipular as versões contínuas da transformada de Fourier, mas para trabalhar com imagens, usamos suas versões discretas

$$g(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \text{ para } u \text{ de } 0 \text{ a } M-1$$

e sua inversa

$$f(x) = \sum_{u=0}^{M-1} g(u) e^{j2\pi ux/M} \text{ para } x \text{ de } 0 \text{ a } M-1$$

— O conceito de domínio da frequência segue a fórmula de Euler

$$e^{j\theta} = \cos \theta + j \sin \theta$$

— Substituindo na transformada direta discreta, temos

$$g(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) (\cos(2\pi ux/M) - j \sin(2\pi ux/M))$$

— e na transformada inversa discreta, temos

$$f(x) = \sum_{u=0}^{M-1} g(u) (\cos(2\pi ux/M) + j \sin(2\pi ux/M))$$

— A transformada de Fourier discreta bidimensional pode ser calculada pela fórmula

$$g(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)}$$

Nesse caso, a inversa pode ser calculada pela fórmula

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} g(u, v) e^{j2\pi(ux/M+vy/N)}$$

- Substituindo a fórmula de Euler na transformada discreta bidimensional de Fourier, obtemos

$$g(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos \left( 2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right) \right) - j \sin \left( 2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right) \right)$$

e sua inversa

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} g(u, v) \cos \left( 2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right) \right) + j \sin \left( 2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right) \right)$$

### 4.3 *Fast Fourier Transform*

- A complexidade do algoritmo da transformada de Fourier é quadrática no número de pixels da imagem. Para calcular cada pixel da imagem de saída, é necessário consultar todos os pixels da imagem de entrada. Se  $P$  é o número de pixels da imagem, a complexidade da transformada de Fourier é  $O(P^2)$ . Felizmente, é possível acelerar o cálculo através do uso da Fast Fourier Transform (FFT), cuja complexidade é  $O(P \log(P))$ . A seguir veremos o algoritmo de Sande-Tukey descrito por Ramirez [10].
- O algoritmo requer como entrada um vetor unidimensional cujo tamanho é uma potência de 2. Para processar imagens bidimensionais, é necessário primeiro aplicar o algoritmo em todas as linhas da imagem de entrada, e em seguida, em todas as colunas da imagem resultante.
- A expressão para calcular a DFT de um sinal unidimensional é dada por

$$g(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M}$$

para  $u$  de 0 a  $M-1$ .

- Seja  $W = e^{-j2\pi/M}$ . Omitindo o fator  $1/M$ , que pode ser aplicado no final do cálculo, e trocando  $f$  por  $f_0$ , onde o índice subscrito denota o estágio do cálculo, teremos

$$g(u) = \sum_{x=0}^{M-1} f_0(x) W^{-ux}$$

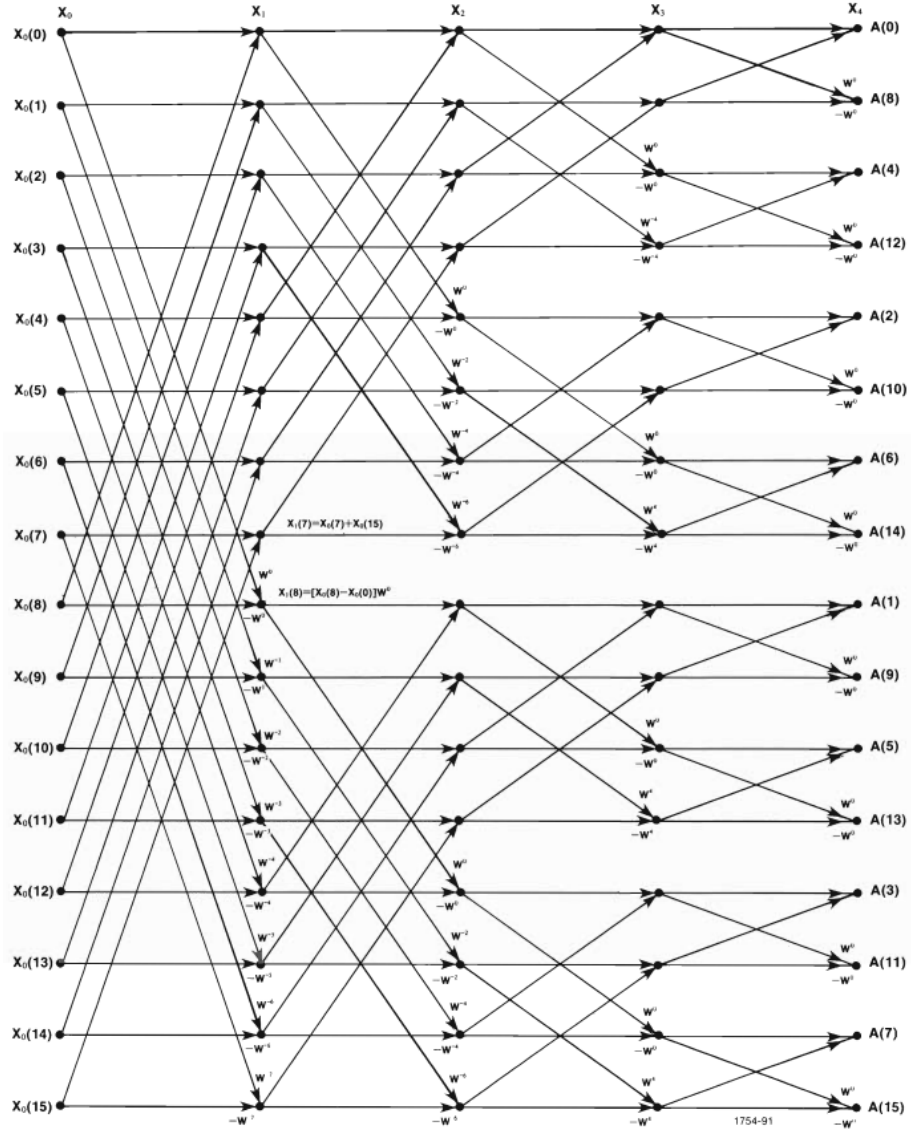


Figura 4.1: Esquema de uma FFT de um vetor de tamanho 16.

- O cálculo da FFT consiste de  $\log_2 M = K$  estágios. Cada estágio requer pares de cálculos da forma

$$f_{k+1}(r) = f_k(r) + f_k(s)$$

e

$$f_{k+1}(s) = (f_k(r) + f_k(s))W^{-p}$$

para inteiros  $r, s, p$  entre 0 e  $M - 1$ , e para  $k$  de 0 a  $K$ .

- A Figura 4.1 mostra o esquema do cálculo da FFT de um vetor de tamanho  $M = 16$ . Cada conjunto de 4 setas saindo de um par  $(f_k(r), f_k(s))$  e chegando em um par  $(f_{k+1}(r), f_{k+1}(s))$ , pelo seu formato, é chamado *butterfly*.
- A diferença entre  $s$  e  $r$  diminui com o aumento de  $k$  e é dada por  $2^{K-k-1}$ . Já o expoente  $p$  começa sempre como 0 em cada grupo de *butterflies* que se sobrepõem. O incremento do valor de  $p$  depende de  $k$  e é dado por  $2^{k+1}$ . A tabela 4.1 mostra os cálculos necessários para obter a FFT de um vetor de tamanho  $M = 16$ .

$k = 0$ $s - r = 8$ $\Delta p = 1$	$k = 1$ $s - r = 4$ $\Delta p = 2$
$f_1(0) = (f_0(0) + f_0(8))$ $f_1(1) = (f_0(1) + f_0(9))$ $f_1(2) = (f_0(2) + f_0(10))$ $f_1(3) = (f_0(3) + f_0(11))$ $f_1(4) = (f_0(4) + f_0(12))$ $f_1(5) = (f_0(5) + f_0(13))$ $f_1(6) = (f_0(6) + f_0(14))$ $f_1(7) = (f_0(7) + f_0(15))$ $f_1(8) = (f_0(0) - f_0(8))W^{-0}$ $f_1(9) = (f_0(1) - f_0(9))W^{-1}$ $f_1(10) = (f_0(2) - f_0(10))W^{-2}$ $f_1(11) = (f_0(3) - f_0(11))W^{-3}$ $f_1(12) = (f_0(4) - f_0(12))W^{-4}$ $f_1(13) = (f_0(5) - f_0(13))W^{-5}$ $f_1(14) = (f_0(6) - f_0(14))W^{-6}$ $f_1(15) = (f_0(7) - f_0(15))W^{-7}$	$f_2(0) = (f_1(0) + f_1(4))$ $f_2(1) = (f_1(1) + f_1(5))$ $f_2(2) = (f_1(2) + f_1(6))$ $f_2(3) = (f_1(3) + f_1(7))$ $f_2(4) = (f_1(0) - f_1(4))W^{-0}$ $f_2(5) = (f_1(1) - f_1(5))W^{-2}$ $f_2(6) = (f_1(2) - f_1(6))W^{-4}$ $f_2(7) = (f_1(3) - f_1(7))W^{-6}$ $f_2(8) = (f_1(8) + f_1(12))$ $f_2(9) = (f_1(9) + f_1(13))$ $f_2(10) = (f_1(10) + f_1(14))$ $f_2(11) = (f_1(11) + f_1(15))$ $f_2(12) = (f_1(8) - f_1(12))W^{-0}$ $f_2(13) = (f_1(9) - f_1(13))W^{-2}$ $f_2(14) = (f_1(10) - f_1(14))W^{-4}$ $f_2(15) = (f_1(11) - f_1(15))W^{-6}$
$k = 2$ $s - r = 2$ $\Delta p = 4$	$k = 3$ $s - r = 1$ $\Delta p = 8$
$f_3(0) = (f_2(0) + f_2(2))$ $f_3(1) = (f_2(1) + f_2(9))$ $f_3(2) = (f_2(0) - f_2(2))W^{-0}$ $f_3(3) = (f_2(1) - f_2(3))W^{-4}$ $f_3(4) = (f_2(4) + f_2(6))$ $f_3(5) = (f_0(5) + f_0(7))$ $f_3(6) = (f_2(4) - f_2(6))W^{-0}$ $f_3(7) = (f_2(5) - f_2(7))W^{-4}$ $f_3(8) = (f_2(8) + f_2(10))$ $f_3(9) = (f_2(9) + f_2(11))$ $f_3(10) = (f_2(8) - f_2(10))W^{-0}$ $f_3(11) = (f_2(9) - f_2(11))W^{-4}$ $f_3(12) = (f_2(12) + f_2(14))$ $f_3(13) = (f_2(13) + f_2(15))$ $f_3(14) = (f_2(12) - f_2(14))W^{-0}$ $f_3(15) = (f_2(13) - f_2(15))W^{-4}$	$f_4(0) = (f_3(0) + f_3(1))$ $f_4(1) = (f_3(0) - f_3(1))W^{-0}$ $f_4(2) = (f_3(2) + f_3(3))$ $f_4(3) = (f_3(2) - f_3(3))W^{-0}$ $f_4(4) = (f_3(4) + f_3(5))$ $f_4(5) = (f_3(4) - f_3(5))W^{-0}$ $f_4(6) = (f_3(6) + f_3(7))$ $f_4(7) = (f_3(6) - f_3(7))W^{-0}$ $f_4(8) = (f_3(8) + f_3(9))$ $f_4(9) = (f_3(8) - f_3(9))W^{-0}$ $f_4(10) = (f_3(10) + f_3(11))$ $f_4(11) = (f_3(10) - f_3(11))W^{-0}$ $f_4(12) = (f_3(12) + f_3(13))$ $f_4(13) = (f_3(12) - f_3(13))W^{-0}$ $f_4(14) = (f_3(14) + f_3(15))$ $f_4(15) = (f_3(14) - f_3(15))W^{-0}$

Tabela 4.1: Etapas do cálculo da FFT de  $f$ .

- Após a execução dos passos descritos, teremos em  $f_K$ , ou seja,  $f_4$  no exemplo, os valores necessários para obter o resultado fora de ordem. É necessário executar a etapa de *bit reversal* para obter o vetor resultante  $g$  na ordem correta. Essa etapa consiste em copiar para  $g(u)$  o valor de  $f_K(R(u))$  onde a operação  $R$  denota a reversão dos bits do parâmetro de entrada.

$g(0)$	$=$	$g(0b0000)$	$=$	$f_4(0b0000)$	$=$	$f_4(0)$
$g(1)$	$=$	$g(0b0001)$	$=$	$f_4(0b1000)$	$=$	$f_4(8)$
$g(2)$	$=$	$g(0b0010)$	$=$	$f_4(0b0100)$	$=$	$f_4(4)$
$g(3)$	$=$	$g(0b0011)$	$=$	$f_4(0b1100)$	$=$	$f_4(12)$
$g(4)$	$=$	$g(0b0100)$	$=$	$f_4(0b0010)$	$=$	$f_4(2)$
$g(5)$	$=$	$g(0b0101)$	$=$	$f_4(0b1010)$	$=$	$f_4(10)$
$g(6)$	$=$	$g(0b0110)$	$=$	$f_4(0b0110)$	$=$	$f_4(6)$
$g(7)$	$=$	$g(0b0111)$	$=$	$f_4(0b1110)$	$=$	$f_4(14)$
$g(8)$	$=$	$g(0b1000)$	$=$	$f_4(0b0001)$	$=$	$f_4(1)$
$g(9)$	$=$	$g(0b1001)$	$=$	$f_4(0b1001)$	$=$	$f_4(9)$
$g(10)$	$=$	$g(0b1010)$	$=$	$f_4(0b0101)$	$=$	$f_4(5)$
$g(11)$	$=$	$g(0b1011)$	$=$	$f_4(0b1101)$	$=$	$f_4(13)$
$g(12)$	$=$	$g(0b1100)$	$=$	$f_4(0b0011)$	$=$	$f_4(3)$
$g(13)$	$=$	$g(0b1101)$	$=$	$f_4(0b1011)$	$=$	$f_4(11)$
$g(14)$	$=$	$g(0b1110)$	$=$	$f_4(0b0111)$	$=$	$f_4(7)$
$g(15)$	$=$	$g(0b1111)$	$=$	$f_4(0b1111)$	$=$	$f_4(15)$

Tabela 4.2: *Bit reversal* de  $f_4$  para obtenção de  $g$ .

- Finalmente, é necessário dividir o vetor  $g$  por  $M$ , para que  $g(0)$  seja igual à média de  $f$  como convencionado.
- O algoritmo para o cálculo da inversa, ou seja, da IFFT, é quase idêntico, com duas diferenças. Uma é que não é necessário dividir o resultado por  $M$ . Também é preciso remover o sinal de menos do expoente de  $W^{-p}$ , obtendo assim

$$f_{k+1}(s) = (f_k(r) + f_k(s))W^p$$

# Referências Bibliográficas

- [1] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.
- [2] Luciano da F. Costa and R. M. Cesar Jr. *Shape Analysis and Classification*. CRC Press, 2 edition, 2001.
- [3] E. R. Dougherty and R. A. Lotufo. *Hands-on Morphological Image Processing (SPIE Tutorial Texts in Optical Engineering Vol. TT59)*. SPIE Publications, July 2003.
- [4] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2 edition, 2001.
- [5] Olivier Faugeras. *Three-dimensional computer vision: Geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.
- [6] Olivier Faugeras, Quang-Tuan Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.
- [7] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., USA, 3 edition, 2006.
- [8] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, New York, NY, USA, 2000.
- [9] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine vision*. McGraw-Hill Science/Engineering/Math, 1 edition, 1995.
- [10] Robert W. Ramirez. *The FFT: Fundamentals and Concepts*. Tektronix, Inc., 1975.
- [11] Pierre Soille. *Morphological image analysis: principles and applications*. Springer, 1999. pag 57: Composition.
- [12] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.