UNIVERSIDADE DE BRASÍLIA

Autores
Érica de Oliveira da Costa
Guilherme de Oliveira Costa
Heitor Moraes Couto
Leonardo Alves da Costa
Thales Portela

Introdução ao Processamento Digital de Imagens com Matlab

Organização
Marcelino Monteiro de Andrade

Brasília
1 de Julho de 2013

Lista de Figuras

1.1 Representação cartesiana de uma imagem. 9
1.2 Exemplo de redução de pixels. 10
1.3 Cálculo da quantidade de tons de cinza percebidos pelo observador 11
1.4 Imagens contínua e quantizada 12
1.5 Paletas a serem comparadas. 13
1.6 Comparação entre paleta com tons de cinza. 17
1.7 Demarcação da distância a ser medida. 18
1.8 Exemplo de reflexão simétrica. 18
1.9 Exemplo de reflexão na origem. 18
1.10 Exercício de transformações 2D. 19

2.1 Gráfico da trasnformação negativa 22
2.2 Exemplo de negativo da imagem. 22
2.3 Gráfico genérico de transformação de alargamento de contraste. 23
2.4 Exemplo de alargamento de contraste. 24
2.5 Imagem com correção de gamma 25
2.6 Exemplo de histograma normalizado. 27
2.7 Resultado da aplicação de equalização de histograma. 29
2.8 Exemplo de máscara 3×3. 29
2.9 Exemplo de máscaras espaciais para filtros passa-baixas. 30
2.10 Exemplo de filtro de suavização 31
2.11 Exemplo de máscara espacial para filtro passa altas 31
2.12 Imagens original e filtrada com filtro passa-altas. 31
2.13 Exemplo de filtragem alto-reforço. 32

3.1 Exemplo de segmentação de imagens. 33
3.2 Exemplo de dilatação. 34
3.3 Exemplo de erosão em uma imagem. 34
3.4 Exemplo da influência do elemento estruturante na dilatação. 35
3.5 Máscara 3×3 para o Laplaciano. 38
3.6 Exemplo de Operador Laplaciano da imagem 38
3.7 Exemplo de gradiente morfológico. 39
3.8 Imagem de antibiograma a ser segmentada. 41
3.9 Imagem gerada em escala de cinza. 41
3.10 Imagem binária gerada a partir do limiar de Otsu. 42
3.11 Resultado do fechamento com detalhe em destaque. 42
3.12 Resultado da abertura da imagem 43
3.13 Imagem final com disco isolado. 43
3.14 Imagem para detecção de borda. 44

4.1 Representação de uma imagem em matriz 45

A.1 GNU Octave 51
A.2 O ambiente de desenvolvimento da Verbosus. 52

B.1 Especro de cores visíveis ao olho humano. 53
B.2 Sistema de cor XYZ. 55
B.3 Sistema de cor RGB 56
B.4 Sistema de cor CMY 56
B.5 Gamma RGB projetado no CIELAB 57

C.1 Representação de um sistema bidimensional em espaço discreto 60

Lista de Algoritmos

1.1 Exemplo de redução de pixels. 10
1.2 Exemplo de comparação de paletas de cinza. 11
1.3 Cálculo da distância entre dois pontos. 14
1.4 Exemplo de expansão de uma imagem. 14
1.5 Exemplo de reflexão simétrica. 15
1.6 Exemplo de reflexão na origem. 16
2.1 Função que calcula o negativo da imagem. 22
2.2 Função que efetua o alargamento de contraste da imagem de entrada. . . . 23
2.3 Exemplo de correção de gamma. 25
2.4 Função que calcula o histograma normalizado. 26
2.5 Exemplo de equalização do histograma de uma imagem. 27
2.6 Função que filtra linearmente a imagem de acordo com a máscara fornecida. 30
2.7 Calcula o alto-reforço de uma imagem. 32
3.1 Função que realiza a erosão em uma imagem binária. 35
3.2 Função que realiza a dilatação em uma imagem binária 36
3.3 Abertura e fechamento de uma imagem de trilha de circuito impresso . . . 37
3.4 Exemplo de cálculo da detecção de bordas em uma imagem binária. 38
3.5 Calcula o limiar ótimo para binarização de uma imagem com o método de Otsu. 40
3.6 Exemplo de segmentação de uma placa de Petri. 42

Sumário

1 Fundamentos das Imagens Digitais **9**
1.1 Formação de imagens . . . 9
1.2 Amostragem e quantização . . . 10
1.3 Relacionamentos básicos entre pixels . . . 13
1.3.1 Vizinhos de um pixel . . . 13
1.3.2 Distância entre pixels . . . 13
1.4 Transformações lineares no plano . . . 14
1.4.1 Expansão ou contração uniforme . . . 14
1.5 Reflexão em torno de um eixo . . . 15
1.6 Exercícios propostos . . . 16

2 Transformações de intensidade **21**
2.1 Transformação da escala de cinza . . . 21
2.2 Equalização de histograma . . . 25
2.2.1 Filtragem no domínio espacial . . . 28
2.3 Filtragem no domínio de frequência . . . 32
2.4 Exercícios propostos . . . 32

3 Segmentação de Imagens **33**
3.1 Operadores morfológicos . . . 33
3.2 Detecção de bordas em imagens binárias . . . 37
3.3 Limiarização aplicando o método de Otsu . . . 39
3.4 Segmentação orientada a regiões . . . 40
3.5 Exercícios propostos . . . 44

4 Representação e Descrição de Imagens **45**
4.1 Descritores de imagens . . . 45
4.2 Códigos de Cadeia . . . 46
4.3 Descritores de fronteiras . . . 46
4.3.1 Comprimento do contorno . . . 46
4.3.2 Curvatura . . . 46
4.3.3 Número de formas . . . 47
4.4 Descritores regionais . . . 47
4.4.1 Descritores topológicos . . . 47
4.4.2 Momentos . . . 48
4.5 Exercícios Propostos . . . 49

A GNU Octave **51**
A.1 Método de instalação 52
A.1.1 Windows 52
A.1.2 Linux 52
A.1.3 Mac OS X 52

B Sistemas de Cores **53**
B.1 Modelo XYZ 54
B.2 Sistema RGB 55
B.3 Sistema CMY 56
B.4 Sistema HSL 56
B.5 Sistema CIELAB 57

C A Transformada de Fourier **59**
C.1 Sistemas multidimensionais em espaço discreto 60
C.2 Transformada de Fourier Bidimensional 60

Capítulo 1

Fundamentos das Imagens Digitais

Érica de Oliveira da Costa
Leonardo Alves da Costa

As imagens digitais nada mais são do que matrizes de elementos, chamados de pixels. Cada pixel é associado a uma posição e a ele é atribuído um valor que é relativo à cor que se deseja representar naquele ponto. Para facilitar a notação e manipulação matemática, a imagem é descrita como uma função $f(x, y)$, representada em um plano cartesiano, conforme a Figura 1.1.

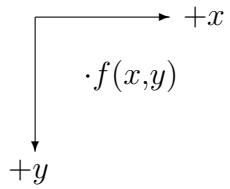


Figura 1.1: Representação cartesiana de uma imagem.

Do ponto de vista matemático, uma imagem é considerada uma função bidimensional $f(x,y)$ com as coordenadas planas x e y e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem no referido ponto. Quando as posições (x, y) e a amplitude de f fazem parte de um conjunto de valores finitos, ou discretos, a imagem é chamada de imagem digital [**?**].

Em outras palavras, uma imagem digital pode ser representada por meio de uma matriz $N \times M$ de modo que cada elemento da matriz corresponde à intensidade ou nível de cinza em um determinado ponto da imagem. Estes elementos de imagens digitais são os pixels, que, na prática, possuem uma intensidade que representa uma cor em um de terminado ponto da imagem.

1.1 Formação de imagens

A luz, como onda eletromagnética, é disposta em um espectro contínuo que varia de acordo com a frequência de propagação. Sabe-se que a imagem observada de um objeto é formada pela luz, portanto energia, que ele reflete ou irradia. Essa energia pode ser captada por um sensor, que gera um sinal contínuo. Por fim, para a representação digital do sinal, é necessário discretizar as informações obtidas. A discretização minimiza a quantidade de cores do espectro, no entanto, a visão humana não é capaz de diferenciar variações muito sutis de frequência.

Exercício de fixação

A quantidade de pixels em uma imagem define sua resolução. O Algoritmo 1.1 é capaz de diminuir a resolução de uma imagem utilizando a média das amplitudes dos pixels em regiões definidas de acordo com o passo escolhido. A Figura 1.2 apresenta o resultado deste exemplo.

```
im=imread('LENNA.bmp');
im1=im(:,:,1);
k=0;    % índice da imagem reduzida
p=0;    % índice da imagem reduzida
passo=2;  % = quantidade de pixels utilizados para a média

for i=1:passo:250-(passo+1)
   k=k+1;
     for j=1:passo:250-(passo+1)
       p=p+1;
        % Formação da imagem reduzida pela média das regiões
        im2(k,p)=mean2(im1(i:i+passo-1,j:j+passo-1));
     end
   p=0;
end

figure(gcf)
subplot(1,2,1)
imshow(im1);
subplot(1,2,2)
imshow(uint8(im2));
```

Algoritmo 1.1: Exemplo de redução de pixels.

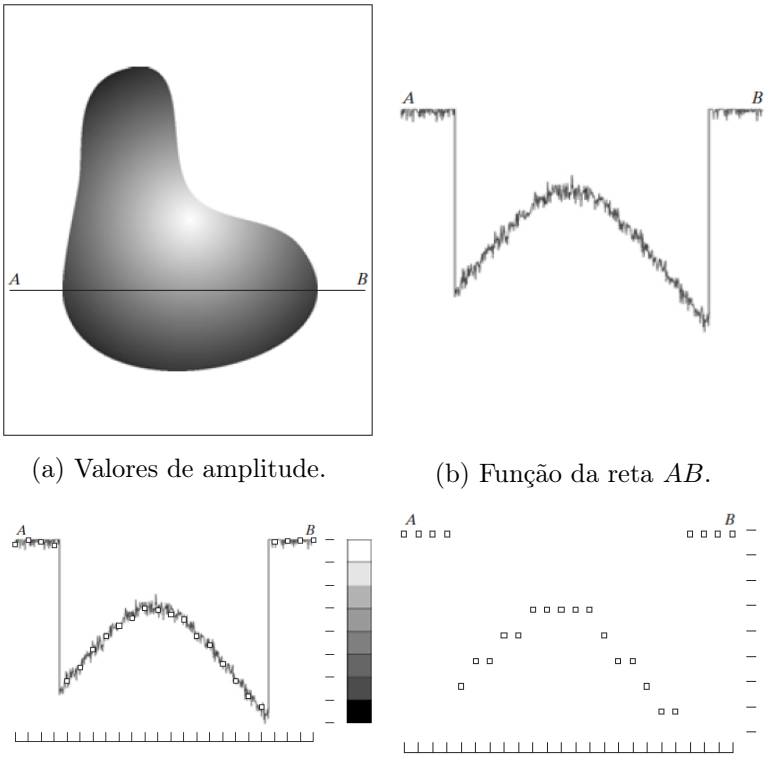
(a) Imagem original. (b) Imagem reduzida.

Figura 1.2: Exemplo de redução de pixels.

1.2 Amostragem e quantização

Uma imagem é continua com respeito às coordenadas (x,y) e à amplitude $f(x,y)$. O processo de digitalização consiste em discretizar ambas as características, também chamada de quantização. A Figura 1.3a é uma plotagem dos valores da amplitude para todo o objeto, enquanto a Figura 1.3b representa a função em apenas um seguimento de reta AB. A

amostragem das coordenadas é mostrada na Figura 1.3c, em que os quadrados brancos equidistantes, sobrepostos ao longo da função, são as amostras para uma cada posição, já em quantidade discreta. Por fim, a Figura 1.3d conclui o processo já com a quantização dos níveis de cinza. Neste caso, a amplitude $f(x,y)$ pode assumir um dos 8 níveis, e é ajustada conforme a proximidade vertical das demarcações específicas de cada nível. O resultado do processo é mostrado na Figura 1.4



(a) Valores de amplitude.

(b) Função da reta AB.

(c) Sampling.

(d) Quantização.

Figura 1.3: Cálculo da quantidade de tons de cinza percebidos pelo observador [?].

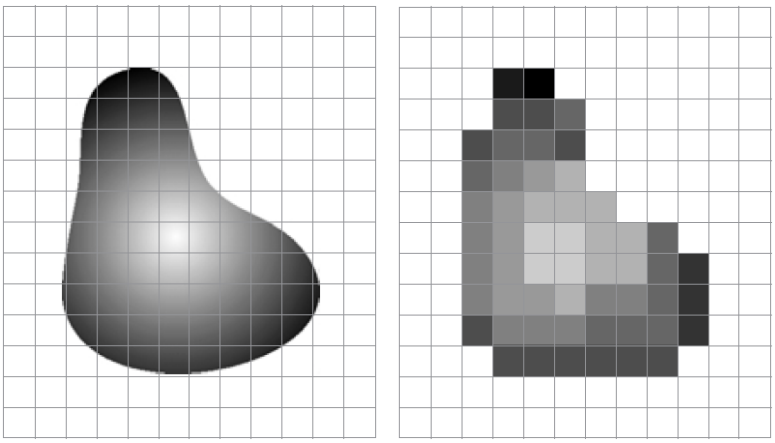
Exercício de fixação

A visão humana é incapaz de identificar pequenas variações de frequência da luz. O código abaixo permite constatar a quantidade de tons de cinza distinguíveis pelo observador.

```
imref(1:500, 1:500) = 0;
subplot(1,2,1)
imshow(uint8(imref))

im = imref;
nivel = 0;
subplot(1,2,2)
imshow(uint8(im))
vetor = 0;

while(nivel < 255)
  j = input('Houve diferença? 1 - Sim 2 - Não ');
  % armazena o nível de cinza diferenciado pelo observador
  if j == 1
```

```
    subplot(1,2,1);
    imshow(uint8(im));
    vetor = [vetor nivel];

  elseif j == 2
    im = im + 2;  % aumenta a amplitude dos pixels
    nivel = nivel + 2;

    subplot(1,2,2);
    imshow(uint8(im));

    imref = im;
  end
end

paleta_usr = vetor(1).*ones(10,10);
for i = 2 : length(vetor)
  paleta_usr = [paleta_usr vetor(i).*ones(10,10)];
end

paleta_ref = 0.*ones(10,1);
for i = 1:255
  paleta_ref = [paleta_ref i.*ones(10,1)];
end

figure(2)
subplot(2,1,1)
imshow(uint8(paleta_usr))

subplot(2,1,2)
imshow(uint8(paleta_ref))
```

Algoritmo 1.2: Exemplo de comparação de paletas de cinza.



(a) Imagem original. (b) Imagem quantizada.

Figura 1.4: Imagens contínua e quantizada [?].

(a) (b)

Figura 1.5: Paletas a serem comparadas.

(a) Paleta com diversos tons de cinza.

(b) Transição contínua entre o preto e branco.

Figura 1.6: Comparação entre paleta com tons de cinza.

1.3 Relacionamentos básicos entre pixels

1.3.1 Vizinhos de um pixel

Um pixel tem vizinhos verticais e horizontais, chamados vizinhos-de-4 de p, simbolizados por $N_4(p)$. Estes vizinhos são definidos pelo intervalo:

$$(x+1, y), (x-1,y), (x,y+1),(x,y-1) \tag{1.1}$$

O vizinhos diagonais, simbolizados por $N_D(p)$. Estes podem ser definidos como:

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1) \tag{1.2}$$

Assim, a junção de $N_4(p)$ e $N_D(p)$ é chamada vizinhos-de-8, simbolizada por $N_8(p)$.

1.3.2 Distância entre pixels

Considerando os pixels p e q com coordenadas (x, y) e (s, t), respectivamente, as distâncias entre p e q são dadas por:

i) Distância euclidiana:

$$D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2} \tag{1.3}$$

ii) Distância D_4:

$$D_4(p, q) = |x-s| + |y-t| \tag{1.4}$$

iii) Distância D_8:

$$D_8(p, q) = \max(|x-s|, |y-t|) \tag{1.5}$$

Exercício de fixação

Calcular a distância entre dois pontos em uma imagem.

```
im=imread('armario.jpg');
imshow(im)

while (1)

[x,y] = ginput(2);  % inserir os pontos atraves do mouse
rcp = 0.1377;  % relação de escala
d = sqrt((x(1)-x(2))^2+(y(1)-y(2))^2)  % distancia entre os pixels

D.real = [num2str(d*rcp) ' cm'] % distancia real entre os pontos

hold on
plot(x,y,'-*r')

pause
end
```

Algoritmo 1.3: Cálculo da distância entre dois pontos.

(a) Demarcação da distância (em vermelho).

```
d =

  1.5433e+003


D =

    real: '212.5175 cm'
```

(b) Resultado do cálculo.

Figura 1.7: Demarcação da distância a ser medida.

1.4 Transformações lineares no plano

Uma vez que uma imagem pode ser representada por uma matriz de pontos, ela também pode sofrer transformações baseadas em operações com matrizes.

1.4.1 Expansão ou contração uniforme

A transformação que permite ampliar ou contrair uma imagem a partir da multiplicação de seus índices por uma constante é chamada de transformação de expansão ou de contração.

A Equação 1.6 apresenta um exemplo desta transformação.

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow 2 \times \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.6}$$

Exercício de fixação

Efetuar a ampliação de uma imagem.

```
ima=imread('lei.jpg');
im(:,:)= ima(:,:,1);

im = imresize(im, [250 250]);
T=size(im);
D=T(1)*T(2)

im1=0;
o=2; % constante de ampliação/contração da imagem

for i=1:T(1)
  for j=1:T(2)
    im1(round(o*i),round(o*j),:)=im(i,j,:);  % amplia/contrai a imagem
  end
end

figure(gcf)
subplot(2,1,1)
imshow(uint8(im))

subplot(2,1,2)
imshow(uint8(im1))
```

Algoritmo 1.4: Exemplo de expansão de uma imagem.

1.5 Reflexão em torno de um eixo

A transformada de reflexão é usada caso seja necessário refletir uma imagem em torno de um eixo qualquer. Por exemplo:

i) Reflexão em torno do eixo x:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ -y \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.7}$$

ii) Reflexão em torno do eixo y:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -x \\ y \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.8}$$

iii) Reflexão na origem:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -x \\ -y \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.9}$$

Exercício de fixação

Reflexão para gerar de imagem simétrica

```
ima=imread('arvore.jpg');
im(:,:)= ima(:,:,1);

im = imresize(im, [250 250]);

T=size(im);
D=T(1)*T(2);

im1=0;
for i=1:T(1)
  for j=1:T(2)
    if j<=T(2)/2
      % reflete a imagem em torno do eixo que corta a imagem ao meio
      im1(i,1+T(2)/2-j)=im(i,T(2)/2+j);
    else
      im1(i,j)=im(i,j);
    end
  end
end

figure(gcf)
subplot(1,2,1)
imshow(uint8(im))

subplot(1,2,2)
imshow(uint8(im1))
```

Algoritmo 1.5: Exemplo de reflexão simétrica.

(a) Imagem original. (b) Reflexão em torno de uma linha vertical.

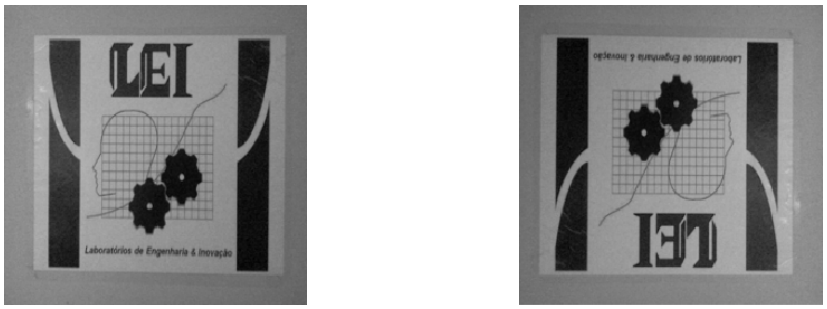
Figura 1.8: Exemplo de reflexão simétrica.

Reflexão na origem

```
ima=imread('lei.jpg');
im(:,:)= ima(:,:,1);

im = imresize(im, [250 250]);

T=size(im);
D=T(1)*T(2)

```

```
im1=0;

for i=1:T(1)
    for j=1:T(2)
      im1(251-i,251-j)=im(i,j);
      % reflete a imagem em torno da origem
    end
end

figure(gcf)
subplot(1,2,1)
imshow(uint8(im))

subplot(1,2,2)
imshow(uint8(im1))
```

Algoritmo 1.6: Exemplo de reflexão na origem.



(a) Imagem original. (b) Imagem refletida em torno do eixo x.

Figura 1.9: Exemplo de reflexão na origem.

1.6 Exercícios propostos

1. A Figura 1.10 mostra uma transformação 2D complexa aplicada a um quadrado unitário. A transformação total pode ser decomposta em termos de transformações simples. Assim, descreva-as e dê uma representação matricial a cada uma delas.

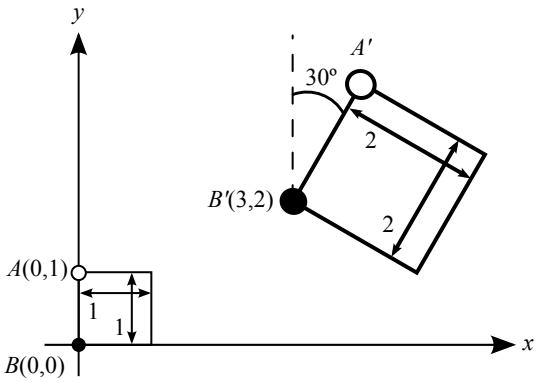


Figura 1.10: Exercício de transformações 2D.

2. Com o uso do Matlab, leia uma imagem qualquer e utilize o conceito de distância entre pontos para encontrar a projeção no eixo horizontal de uma distância transversal selecionada (vide Algoritmo 1.3 da Seção 1.3).

Capítulo 2

Transformações de intensidade e técnicas de filtragem

Heitor Moraes Couto
Leonardo Alves da Costa

As transformações de intensidade e as técnicas de filtragens são técnicas de realce de imagens, cujo objetivo é processar uma imagem para torná-la mais apropriada para uma aplicação específica. Essas técnicas são divididas em duas categorias, processamento no domínio espacial e filtragem no domínio da frequência. Os métodos utilizados no domínio espacial operam diretamente na matriz de pixels que define a imagem. Já os métodos utilizados no domínio da frequência consistem em modificar a imagem no domínio da frequência. Por exemplo, transformações com o resultado da Transformada de Fourier sobre a imagem[1].

Uma função de processamento de imagem no domínio espacial é definida pela Equação 2.1:

$$g(x, y) = T[f(x, y)] \tag{2.1}$$

De modo que:

- $f(x,y)$: imagem de entrada;
- $g(x,y)$: imagem de saída; e,
- T: um operador definido numa vizinhança do ponto (x,y).

2.1 Transformação da escala de cinza

Se a vizinhança do ponto (x,y) da Equação 2.1 é de tamanho 1×1, isto é, um único pixel, T é chamado função de transformação de intensidade, ou função de transformação da escala de cinza de uma imagem, definida pela Equação 2.2:

$$s = T(r) \tag{2.2}$$

De modo que:

[1] Veja mais informações sobre a Transformada de Fourier no Apêndice C.

- r: nível de cinza antes do processamento;
- s: nível de cinza depois do processamento; e,
- T: transformação que mapeia o nível de cinza r no novo nível de cinza s.

Existem diversas transformações de escala de cinza, como, por exemplo, negativo de imagens, alargamento de contraste e correção gamma. Estas técnicas apresentadas nos exemplos seguintes.

Exercício de fixação

Negativo de imagens

O negativo de uma imagem é o inverso dos tons de cinza, ou seja, o seu objetivo é reverter a ordem do preto para o branco. Uma imagem com L níveis de cinza é mostrada na Figura 2.1 e sua expressão é dada pela Equação 2.3

$$s = L - 1 - r \tag{2.3}$$

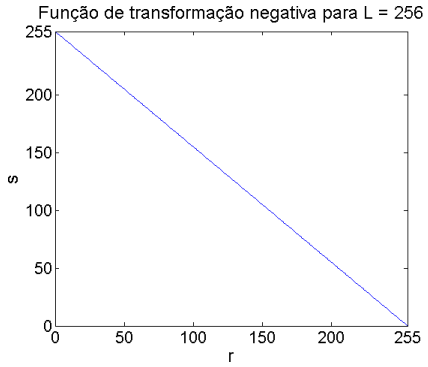


Figura 2.1: Gráfico da trasnformação negativa

O Algoritmo 2.1 aplica a transformação negativa em uma imagem. A Figura 2.2 exemplifica essa transformação.

(a) Imagem original. (b) Negativo da imagem.

Figura 2.2: Exemplo de negativo da imagem.

```
% Entrada: im: imagem que se deseja processar.
% Saída: IM: negativo da imagem de entrada.

function [IM] = NEGT_PDI(im)

```

```
IM = 255 - im;

subplot(1,2,1);
imshow(im);
title('Imagem original');

subplot(1,2,2);
imshow(IM);
title('Negativo da imagem');
end
```

Algoritmo 2.1: Função que calcula o negativo da imagem.

Alargamento de contraste

O objetivo dessa transformação é aumentar a escala dinâmica dos L níveis de cinza de uma imagem. A Figura 2.3 mostra uma típica transformação $T(r)$ para alargamento de contraste. Se $r_1 = s_1$ e $r_2 = s_2$, não há mudança no contraste. Se $r_1 = r_2$, $s_1 = 0$ e $s_2 = L - 1$, $T(r)$ será uma função de limiarização em r_1. Tal função cria um imagem binária onde a saída é 255 para níveis de cinza maiores ou iguais a r_1 e 0 para níveis abaixo desse valor.

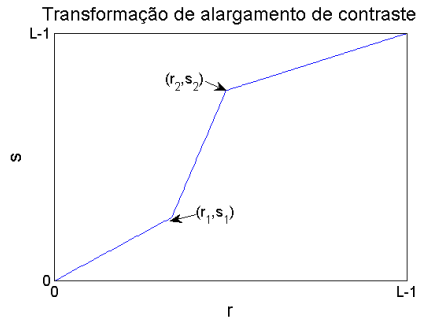


Figura 2.3: Gráfico genérico de transformação de alargamento de contraste.

O Algoritmo 2.2 aplica a transformação para alargamento de contraste. A Figura 2.4 mostra um exemplo da execução do código e a curva da função de transformação utilizada.

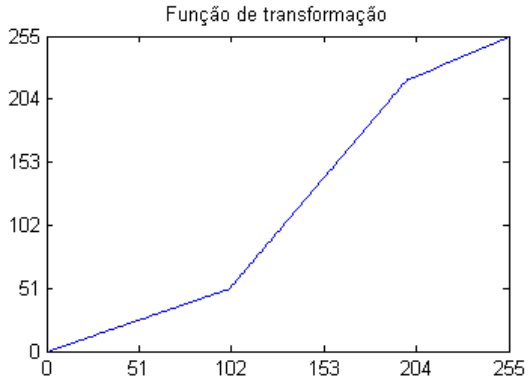
```
% Entrada: im: imagem e ft: função de transformação.
% Saída: IM: imagem com constraste alterado.

function [IM] = CONTRASTE_PDI(im,ft)

tam = size(im);

for i=1:tam(1)
    for j=1:tam(2)
        IM(i,j) = ft(im(i,j) + 1);
    end
end

subplot(1,2,1);
imshow(im);
title('Imagem original');

```

```
subplot(1,2,2);
imshow(uint8(IM));
title('Imagem com contraste alterado');

end
```

Algoritmo 2.2: Função que efetua o alargamento de contraste da imagem de entrada.

(a) Imagem original.

(b) Imagem com contraste alterado.



(c) Curva da função de transformação.

Figura 2.4: Exemplo de alargamento de contraste.

Correção de Gamma

Uma aplicação da correção gamma é a correção do brilho de imagens mostradas em telas de computadores. Estas apresentam resposta de intensidade para voltagem de acordo com uma função exponencial[2]. Por exemplo, se é necessária a apresentação de um pixel com intensidade r, ele será mostrado no display com intensidade $r^{2,5}$. Então, para corrigir esse problema, é utilizada a correção gamma. A imagem é pré-processada antes de ser enviada ao monitor de acordo com a Equação 2.4, com um gamma igual ao inverso do expoente da resposta do monitor. No caso de 2,5, o gamma utilizado para correção seria $\gamma = 1/2{,}5 = 0{,}4$.

$$s = cr^{\gamma} \tag{2.4}$$

- c: constante de correção da gamma;
- γ: fator de correção da gamma; e,

[2]Normalmente com um expoente de 2,5.

- r: valor de intensidade de um pixel.

O Algoritmo 2.3 aplica a correção gamma em uma imagem. Figura 2.5 ilustra a aplicação da correção gamma.

```
% Entrada: im: imagem, c: parâmetro c e gamma: parâmetro gamma.
% Saída: IM: imagem realçada segundo correção gama.

function [IM] = GAMMA_PDI(im,c,gamma)

IM = c.*im2double(im).^gamma;
figure();
title('CORREÇÃO GAMA');

subplot(1,2,1);
imshow(im);
title('Imagem original');

subplot(1,2,2);
imshow(IM);
title(['\gamma = ',num2str(gamma)]);
end
```

Algoritmo 2.3: Exemplo de correção de gamma.

(a) Imagem original. (b) $c = 1$ e $\gamma = 3$. (c) $c = 1$ e $\gamma = 0{,}6$.

Figura 2.5: Imagem com correção de gamma.

2.2 Equalização de histograma

O histograma é um vetor que contém a estatística de aparecimento dos tons de cinza em uma imagem. Assim, ele é dado pela Equação 2.5.

$$h(r_k) = n_k \tag{2.5}$$

De modo que:

- $h(r_k)$: histograma da imagem;
- n_k: o número de pixels que possui o nivel de cinza k; e,
- k: k-ésimo nível de cinza.

Os níveis de cinza da imagem variam no intervalo $[0,L-1]$. Se cada valor do histograma é divido pelo número total de pixels da imagem, diz-se que o histograma foi normalizado. Este é dado pela Equação 2.6.

$$p(r_k) = n_k/n, \quad k = 0, \ldots, L-1 \tag{2.6}$$

De modo que:

- $p(r_k)$: histograma normalizado;
- n: número total de pixels na imagem, dado por $x \times y$; e,
- L: número total de tons de cinza distintos.

Assim, a equalização de histograma, objetiva-se redistribuir os níveis de cinza de forma mais uniforme possível. Desse modo, obtém-se uma imagem com melhor contraste.

Exercício de fixação

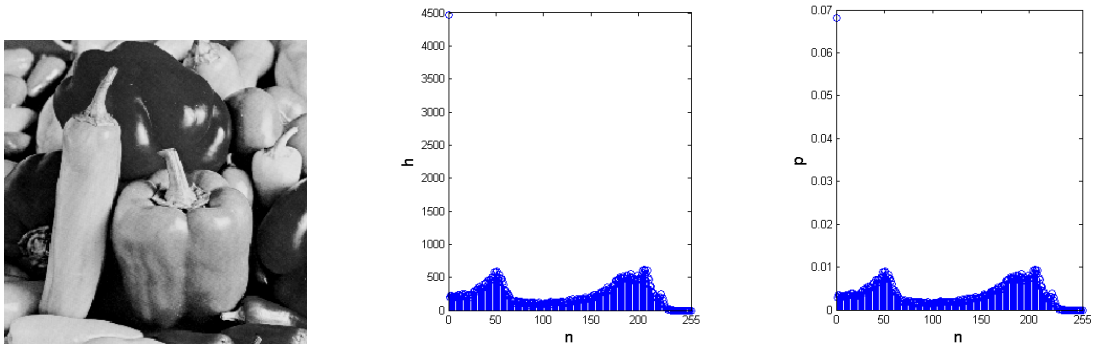
Histograma

O Algoritmo 2.4 calcula o histograma e o histograma normalizado de uma imagem. A Figura 2.6 ilustra um resultado gerado por este código.

```
% Entrada: im: imagem que se deseja processar.
% Saída: h: histogram da imagem e pb: histograma normalizado.

function [h,pb] = HIST_PDI(im)

tam = size(im);

h = zeros(1,256);

% Cálculo do histograma
for i=1:tam(1)
    for j=1:tam(2)
        h(im(i,j)+1) = h(im(i,j)+1) + 1;
    end
end

x = linspace(0,255,256);

figure();
subplot(1,3,1);
imshow(im);
title('Imagem original');

subplot(1,3,2);
stem(x,h);
title('Histograma');
xlim([0 255]);
ylabel('h','fontsize',14);
xlabel('n','fontsize',14);

% Normalização do histograma
```

```
pb = h./(tam(1)*tam(2));

subplot(1,3,3);
stem(x,pb);
title('Histograma normalizado');
xlim([0 255]);
ylabel('p','fontsize',14);
xlabel('n','fontsize',14);
```

Algoritmo 2.4: Função que calcula o histograma normalizado.



(a) Imagem original. (b) Histograma da imagem. (c) Histograma normalizado.

Figura 2.6: Exemplo de histograma normalizado.

O Algoritmo 2.5 calcula o histograma de uma imagem e faz sua equalização. A Figura 2.7 mostra o resultado de sua execução.

```
clear
close all
im = imread('lenna.bmp');

tam = size(im);

h = zeros(1,256);

% Calcula o histograma
for i=1:tam(1)
    for j=1:tam(2)
        h(im(i,j)+1) = h(im(i,j)+1) + 1;
    end
end

x = linspace(0,255,256);
figure();
subplot(2,2,1);
imshow(im);

subplot(2,2,2);
stem(x,h);

% Normaliza o histograma
pb = h./(tam(1)*tam(2));

% Equalizando o histograma
```

```
for i=1:256
    pbc(i) = sum(pb(1:i));
end
for i=1:tam(1)
    for j=1:tam(2)
        IM(i,j) = pbc(im(i,j) + 1);
    end
end

% Imagem com histogram equalizado
IM = IM*255;

h2 = zeros(1,256);

% Cálculo do histograma equalizado
for i=1:tam(1)
    for j=1:tam(2)
        h2(round(IM(i,j))+1) = h2(round(IM(i,j))+1) + 1;
    end
end

% Normalizando histograma equalizado
pb2 = h2./(tam(1)*tam(2));

subplot(2,2,3);
imshow(uint8(IM));

subplot(2,2,4);
stem(h2);
```

Algoritmo 2.5: Exemplo de equalização do histograma de uma imagem.

2.2.1 Filtragem no domínio espacial

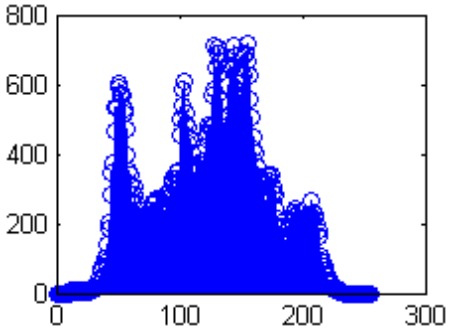
Filtrar uma imagem no domínio espacial significa aplicar uma máscara correspondente a um filtro em uma imagem diretamente em seus pixels. As máscaras são representadas por matrizes e suas operações consistem em mover a máscara pixel a pixel na imagem. A resposta do filtro em um determinado ponto (x,y) da imagem é determinada por uma relação predeterminada.

Para filtros espaciais lineares, a resposta é a soma dos produtos dos elementos da máscara e o correspondente pixel da imagem na vizinhança coberta pela máscara. Para uma imagem de tamanho $M \times N$ e uma máscara $m \times n$, essa resposta é dada pela seguinte relação

$$\begin{aligned} a &= (m-1)/2, \\ b &= (n-1)/2, \\ g(x,y) &= \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s,y+t) \end{aligned} \tag{2.7}$$

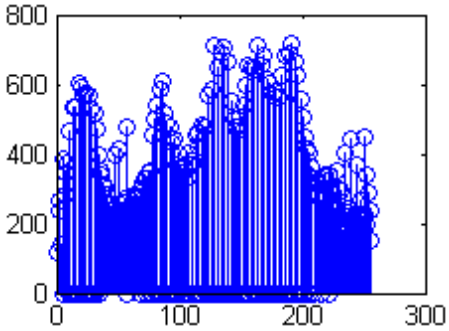
Neste caso, para filtrar a imagem completamente, a Equação 2.7 deve ser aplicada para $x = 0,...,M-1$ e $y = 0,...,N-1$.

(a) Imagem original.



(b) Histograma original

(c) Imagem equalizada.



(d) Histograma equalizado.

Figura 2.7: Resultado da aplicação de equalização de histograma.

$w_{(-1,-1)}$	$w_{(-1,0)}$	$w_{(-1,1)}$
$w_{(0,-1)}$	$w_{(0,0)}$	$w_{(0,1)}$
$w_{(1,-1)}$	$w_{(1,0)}$	$w_{(1,1)}$

Figura 2.8: Exemplo de máscara 3×3.

Já os filtros espaciais não-lineares baseiam-se apenas nos valores dos pixels da vizinhança coberta pela máscara que não seguem a relação descrita pela Equação 2.7. Por exemplo, um filtro que calcule a média dos níveis de cinza dos pixels posicionados sob a máscara pode ser utilizado para redução de ruído de uma imagem.

Os filtros podem ainda ser divididos em filtros de suavização e filtros de aguçamento. O primeiro tipo é utilizado para borramento, supressão de bordas e detalhes finos. Estes são definidos pelos componentes de alta frequência de uma imagem, logo, o borramento é conseguido por meio de um filtro passa-baixas. Também é utilizado para redução de ruído.

Filtros de aguçamento visam aguçar detalhes finos e bordas de uma imagem. Isso corresponde a filtrar a imagem com um filtro passa altas, que atenua os componentes de baixa frequência, responsáveis pelas características que variam lentamente em uma imagem.

Exercício de fixação

O Algoritmo 2.6 é uma função genérica para filtrar uma imagem com filtros lineares. Recebe como entrada a imagem a ser filtrada e a máscara do filtro a ser utilizada.

```
% Entradas: im: imagem e mask: mascára do filtro.
```

```
% Saída: IM: imagem filtrada

function [IM] = FILTERS_PDI(im,mask)

tam = size(im);
tam2 = size(mask);

for i = 1:tam(1) - (tam2(1)-1)
    for j = 1:tam(2) - (tam2(2)-1)
       temp = mask.*double(im(i:i + tam2(1) - 1, j:j + tam2(2) - 1));

       IM(i,j) = sum(sum(temp));
    end
end

figure;
subplot(1,2,1);
imshow(im);
title('Imagem original');

subplot(1,2,2);
imshow(uint8(IM));
title('Imagem filtrada');

end
```

Algoritmo 2.6: Função que filtra linearmente a imagem de acordo com a máscara fornecida.

Filtro de suavização

Como já apresentado, o filtro de suavização é um filtro passa-baixas. A Figura 2.9 mostra exemplos de máscaras espaciais correspondentes a este filtro.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

(a) Máscara 3×3.

$$\frac{1}{25} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

(b) Máscara 5×5.

$$\frac{1}{49} \times \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

(c) Máscara 7×7.

Figura 2.9: Exemplo de máscaras espaciais para filtros passa-baixas.

A Figura 2.10 mostra resultados obtidos com a execução do Algoritmo 2.6 com as máscaras da Figura 2.9. Neste caso, percebe-se que a imagem fica cada vez mais borrada com o aumento da máscara utilizada.

Filtro de aguçamento

É relativo ao filtro passa-altas. Desta forma, a Figura 2.11 mostra uma máscara correspondente a este filtro. A Figura 2.12 mostra o resultado da execução do código de filtragem

(a) Imagem original. (b) Máscara 3×3. (c) Máscara 5×5. (d) Máscara 7×7.

Figura 2.10: Exemplo de filtro de suavização com várias máscaras.

com tal máscara. Assim, é possível notar que as bordas foram realçadas na imagem filtrada.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Figura 2.11: Exemplo de máscara espacial para filtro passa altas

(a) Imagem original. (b) Imagem filtrada.

Figura 2.12: Imagens original e filtrada com filtro passa-altas.

Alto-reforço

Quando se somam a imagem original com a imagem filtrada por um filtro passa-altas, os componentes de baixa frequência atenuados na filtragem são parcialmente restaurados. O resultado disso é uma imagem semelhante a original, porém com as bordas realçadas.

O Algoritmo 2.7, que utiliza a função de filtragem usada anteriormente, realiza o processo de filtragem alto reforço. Com a execução do Algoritmo 2.7 percebe-se que as bordas da imagem original foram realçadas.

```
im =imread('LENNA.bmp');

%Criação da máscara
mask = -ones(3);
mask(2,2) = 8;
mask = (1/9)*mask;

%Filtra a imagem
```

```
IM = FILTERS_PDI(im,mask);

%Calcula o alto reforço
altoref = im(2:255,2:255) + uint8(IM);

close all;

figure;

subplot(1,2,1);
imshow(im);
title('Imagem original');

subplot(1,2,2);
imshow(altoref);
title('Alto reforo');
```

Algoritmo 2.7: Calcula o alto-reforço de uma imagem.

(a) Imagem original. (b) Imagem filtrada.

Figura 2.13: Exemplo de filtragem alto-reforço.

2.3 Filtragem no domínio de frequência

<pendente>

Exercício de fixação

<pendente>

2.4 Exercícios propostos

<pendente>

Capítulo 3

Segmentação de Imagens

Guilherme de Oliveira Costa
Leonardo Alves da Costa

A segmentação consiste no processo de repartir uma imagem, esta técnica possui aplicação direta no processamento morfológico. Assim, imagem é manipulada conforme uma série de passos para permitir a análise em termos de formatos especificos contidos na imagem processada. Um exemplo pode ser visto na Figura 3.1.

(a) Imagem original. (b) Imagem segmentada.

Figura 3.1: Exemplo de segmentação de imagens.

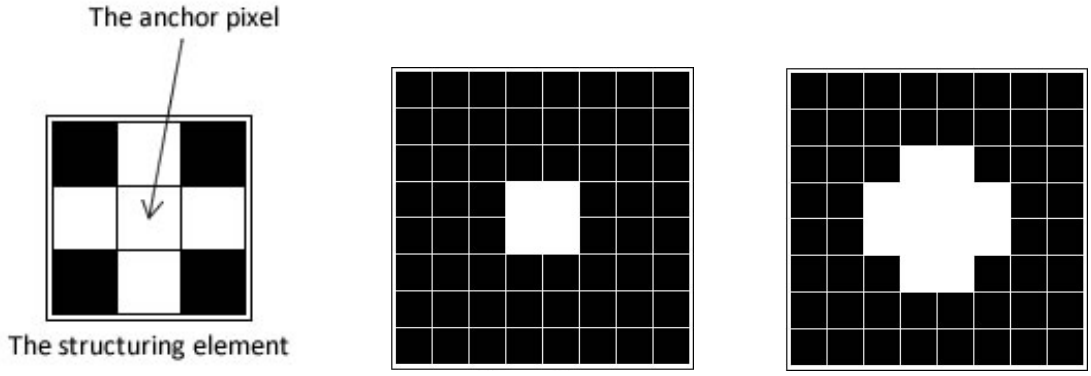
3.1 Operadores morfológicos

Existem quatro operadores morfológicos: erosão, dilatação, abertura e fechamentoo. Estes operadores baseiam-se em uma única premissa: uma pequena máscara (elemento estruturante) é transladada sobre a imagem binária. Assim, as ações podem ser tomadas de acordo com o padrão da máscara. O padrão da máscara é o mesmo dos pixels sobre a máscara (*hit*) ou não (*miss*).

Assim, pode-se, então, definir as operações morfológicas básicas da seguinte forma:

i) **Dilatação:** consiste em aumentar o número de pixels de mesma amplitude nas regiões de interesse. Por exemplo, uma imagem que contém um círculo de raio R pixels, após a dilatação, deverá ter um raio maior do que R em um ou mais pixels.

   No caso mostrado na Figura 3.2, a técnica utilizada para a dilatação consistiu em "carimbar" com elemento estruturante (Figura 3.2a) em todos os pixels brancos da imagem oringial (Figura 3.2b), o que gerou a Figura 3.2c.



(a) Elementro estruturante. (b) Imagem original. (c) Imagem dilatada.

Figura 3.2: Exemplo de dilatação.

ii) **Erosão:** a imagem final deve possuir menos pixels nas bordas dos objetos em comparação a imagem original. Por exemplo, uma imagem que contém um círculo de raio R, e, após a erosão, esse círculo terá seu raio diminuido em um ou mais pixels.

A erosão é feita ao inverso da dilatação, ou seja, os pixels da imagem são comparados a um elemento estruturante, se ele for encontrado, substitui-se por um elemento menor. Caso contrário, não há alterações nos pixels.

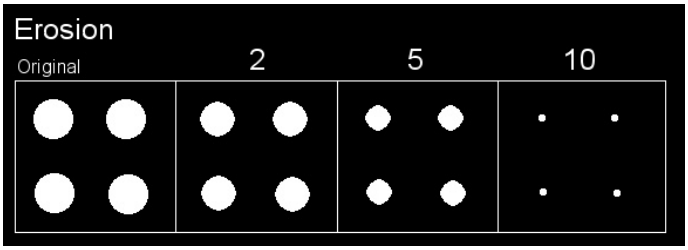
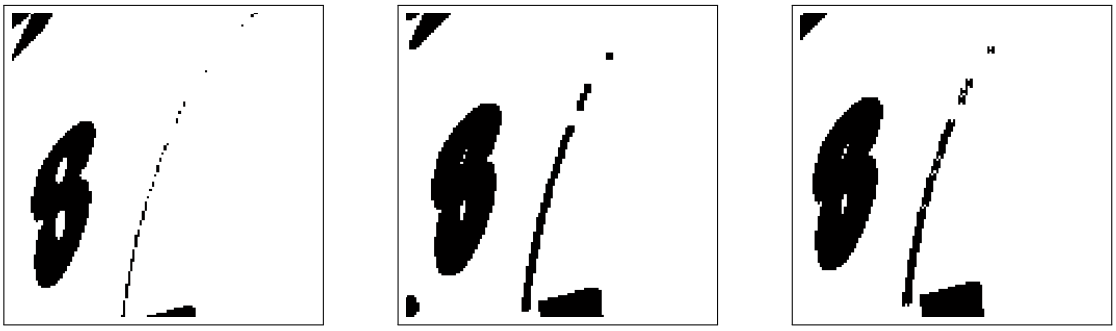


Figura 3.3: Exemplo de erosão em uma imagem.

A Figura 3.3 apresenta a aplicação da operação de erosão diversas vezes seguidas. Neste ponto, vale comentar que as operações de erosão e dilatação não são complementares, isto é, nem sempre a aplicação sucessiva dessas operações nos levará de volta à imagem original. Na realidade, as outras duas operações morfológicas básicas se baseiam justamente nesse fato. A **abertura** consiste na dilatação de uma imagem erodida e é utilizada para remover pequenos objetos. Este objetos são retirados da imagem durante a erosão e a dilatação restaura os elementos restante ao seu tamanho original. O **fechamento** consiste na erosão de uma imagem dilatada, que remove pequenos "furos" da imagem (menores que o elemento estruturante) na fase da dilatação, e a erosão restaura os elementos de volta ao seu tamanho original.

Outro fator importante para essas operações morfológicas, que merece atenção especial, é o formato do elemento estruturante. É visível o efeito de diferentes elementos estruturantes aplicados a uma mesma operação sobre a imagem, o que pode ser visto na Figura 3.4.

Assim, as operações morfológicas são complementares no que diz respeito à amplitude dos pixels, isto é, dilatar os pixels brancos de uma imagem é igual a erodir os pixels pretos da mesma imagem com o uso de um elemento estruturante refletido. Portanto, caso o resultado final de alguma das operações morfológicas mostradas seja o oposto do esperado,



(a) Imagem original. (b) Dilatada com um quadrado. (c) Dilatada com um "H".

Figura 3.4: Exemplo da influência do elemento estruturante na dilatação.

o desenvolvimento dessas funções foram ter sido feitas com a lógica oposta à imaginada por seu criador.

Exercício de fixação

Erosão

Criar uma função que realize a erosão de uma imagem binária baseada em uma máscara fornecida pelo usuário.

```
% Entrada: B: imagem binária e mask: elemento estruturante.
% Saída: R: imagem erodida.

function [R] = erode(B, mask)

  WHITE = 255; %Definição do valor para um pixel branco.
  BLACK = 0; %Definição do valor para um pixel preto

  R = zeros(size(B)); %Alocação da variável final
  sz = size(B);
  sz_mask = size(mask) - 1;

  for i = 2 : sz(1) - sz_mask(1)
    for j = 2 : sz(2) - sz_mask(2)
      %Obtenção de uma amostra da imagem original
      sample = double(B(i:i + sz_mask(1), j:j + sz_mask(2)));
      sample(sample == BLACK) = 1; %Transforma os pixel pretos em 1
      sample(sample == WHITE) = 0; %Transforma os pixels brancos em 0

      match = mask.*sample; %Verificação da correlação entre máscara e amostra

      %Se a correlação entra a máscara e a amostra for menor que 100%
      if(sum(sum(match)) ~= sum(sum(mask)))
        R(i+1,j+1)= WHITE; %O pixel central da iamgem erodida é branco
      end
    end
  end

  R = uint8(R); %conversão da imagem resultante para o o tipo uint8
end
```

Algoritmo 3.1: Função que realiza a erosão em uma imagem binária.

Dilatação

Criar uma função que realize a dilatação de uma imagem binária baseada em uma máscara fornecida pelo usuário.

```
% Entrada: B: imagem binária e mask: elemento estruturante.
% Saída: R: imagem dilatada.

function [R] = dilate(B, mask)

  WHITE = 255; %Definição do valor para um pixel branco.
  BLACK = 0; %Definição do valor para um pixel preto

  R = zeros(size(B)); %Alocação da variável final
  sz = size(B);
  sz_mask = size(mask) - 1;

  for i = 2 : sz(1) - sz_mask(1)
    for j = 2 : sz(2) - sz_mask(2)
      %Obtenção de uma amostra da imagem original
      sample = double(B(i:i + sz_mask(1), j:j + sz_mask(2)));
      sample(sample == BLACK) = 1; %Transforma os pixel pretos em 1
      sample(sample == WHITE) = 0; %Transforma os pixels brancos em 0

      match = mask.*sample; %Verificação da correlação entre máscara e amostra

      %Se pelo menos um dos pixels da amostra "bater" com a máscara
      if(sum(sum(match)) > 0)
        R(i+1,j+1)= BLACK; %O pixel central da imagem dilata é preto
      end
    end
  end

  R = uint8(R); %conversão da imagem resultante para o o tipo uint8
end
```

Algoritmo 3.2: Função que realiza a dilatação em uma imagem binária

Abertura e Fechamento

Criar uma rotina que realize a abertura e fechamento de uma figura com o uso das rotinas desenvolvidas e as equivalentes do Matlab e comparar os resultados.

```
clear all
close all

im = imread('schematic.gif'); %Abrindo o arquivo que contém a imagem
R = zeros(size(im)); %Alocação para a imagem resultante

subplot(131)
imshow(im);
title('Original')

mask = ones(3);

%Abertura
```

```
R = erode(im,mask);
R = dilate(R,mask);

subplot(132)
imshow(R);
title('Abertura')

%Fechamento
R = dilate(im,mask);
R = erode(R,mask);

subplot(133)
imshow(R);
title('Fechamento')

figure(2)

subplot(131)
imshow(im);
title('Original')

mask = ones(3);

%Abertura
R = imopen(im,mask);

subplot(132)
imshow(R);
title('Abertura')

%Fechamento
R = imclose(im,mask);

subplot(133)
imshow(R);
title('Fechamento')
```

Algoritmo 3.3: Abertura e fechamento de uma imagem de trilha de circuito impresso

3.2 Detecção de bordas em imagens binárias

As bordas são definidas em termos de variação, isto é, um objeto tem as bordas bem definidas quando estas variam fortemente em relação aos seus arredores, portanto as bordas podem ser extraídas a partir da análise da variação dos pixels. Uma boa forma de realizar esta operação é com o Laplaciano da imagem, apresentado na Figura 3.6.

O Operador Laplaciano pode ser definido conforme a Equação 3.1. Contudo, ele pode ser aproximado por uma máscara 3×3, conforme a Figura 3.5.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.1}$$

A Equação 3.2 apresenta a equação para o caso discreto:

$$\nabla^2 f = \nabla_x^2 f(x,y) + \nabla_y^2 f(x,y)$$

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y-1) + f(x,y+1)] - 4f(x,y) \tag{3.2}$$

0	1	0
1	-4	1
0	1	0

Figura 3.5: Máscara 3×3 para o Laplaciano.

(a) Imagem original. (b) Laplaciano da imagem.

Figura 3.6: Exemplo de Operador Laplaciano da imagem

Outra maneira de destacar as bordas de uma imagem consiste em utilizar o chamado *gradiente morfológico*, que é nada mais do que a diferença entre a erosão e a dilatação de uma imagem, produzindo um resultado com forma similar à original, porém com bordas destacadas devido às diferenças inseridas pelos operadores morfológicos. O resultado deste procedimento é mostrado na Figura 3.7.

(a) Imagem original. (b) Gradiente morfológico.

Figura 3.7: Exemplo de gradiente morfológico.

Exercício de fixação

Criar uma rotina de detecção de borda que apresente o Laplaciano da imagem e o gradiente morfológico.

```
im = imread('schematic.gif');

D = dilate(im,ones(3));
E = erode(im,ones(3));
```

```
5
6  G = E - D;
7
8  laplace = [0 1 0; 1 -4 1; 0 1 0];
9
10 R = filter_image(im, laplace);
11
12 subplot(121)
13 imshow(G);
14 title('Gradiente Morfológico');
15
16 subplot(122)
17 imshow(R)
18 title('Laplaciano');
```

Algoritmo 3.4: Exemplo de cálculo da detecção de bordas em uma imagem binária.

3.3 Limiarização aplicando o método de Otsu

De acordo com o histograma e contraste, apresentado na Seção 2.2, é possível realizar uma binarização da imagem a partir de um limiar escolhido baseado no histograma. Porém, realizar esta escolha analíticamente é uma tarefa razoavelmente complicada, especialmente em imagens com maior quantidade de detalhes. Para isso, pode-se utilizar o Método de Otsu, que visa determinar um limiar ótimo baseado em técnicas estatísticas.

O método de Otsu baseia-se em uma ideia bem simples, definir um limiar que divide o histograma em duas partes (classes): um *background* e um *foreground*. A partir daí o código itera em busca do limiar que minimiza a *variância ponderada intra-classe* ou, equivalentemente, maximiza a *variância entre classes*. A partir deste limiar ótimo, realiza-se a binarização: todos os valores abaixo deste limiar tornam-se pretos e todos acima do limiar tornam-se brancos.

O método de Otsu funciona melhor se aplicado a imagens cujo histograma é bi-modal, pois isso usualmente caracteriza dois "planos" na imagem, um *foreground* (primeiro plano) e um *background* (fundo). Nos casos em que o histograma não é bi-modal, pode-se resultar em imagens com um dos elementos parcialmente "fundidos" com o fundo da imagem, o que normalmente não é desejável. Nestes casos, uma estratégia melhor seria binarizar a imagem com base em uma pequena amostra, chamado método de Otsu local (em contraste com o global, que utiliza a imagem inteira).

Exercício de fixação

Criar uma função que receba o histograma normalizado de uma imagem e retorne o limiar de Otsu.

```
1  % Entrada: H: histograma normalizado de uma imagem.
2  % Saída: L: Valor do limiar ótimo para binarização.
3
4  function [L] = OTSU_PDI(H)
5
6    %Cria 255 pontos igualmente espaçados de 1 a 256
7    x = linspace(0,255,256);
8
```

```
  for(t=1:256)
    %Calculo da variância ponderada da primeira classe
    var1 = var(x(1:t),H(1:t));
    w1 = sum(H(1:t)); %Cálculo do peso da primeira classe

    if t==256
      w2 = sum(H(t:256)); %Cálculo do peso da segunda classe
      %Cálculo da variância ponderada da segunda classe
      var2 = var(x(t:256),H(t:256));
    else
      %Cálculo do peso da segunda classe
      w2 = sum(H(t + 1:256));
      %Cálculo da variância ponderada da segunda classe
      var2 = var(x(t + 1:256),H(t + 1:256));
    end

    sig(t) = w1*var1 + w2*var2; %Cálculo da variância intra-classe
  end

  %Localizar o índice no qual o valor das variâncias cálculadas é mínimo
  L=find(sig==min(sig))-1;
end
```

Algoritmo 3.5: Calcula o limiar ótimo para binarização de uma imagem com o método de Otsu.

3.4 Segmentação orientada a regiões

A segmentação é uma tarefa bem comum, ela identifica um objeto de interesse e o extrái para que ele possa ser tratado isoladamente. Desta forma, com o uso das técnicas mostradas até aqui, é possível realizar um tarefa de maneira simples. Para isso, a Figura 3.8 apresenta uma placa de Petri, que é uma imagem de um antibiograma, que será utilizada para avaliar a eficácia de antibióticos sobre bactérias. O objetivo é gerar uma imagem de saída com o disco de Petri isolado.

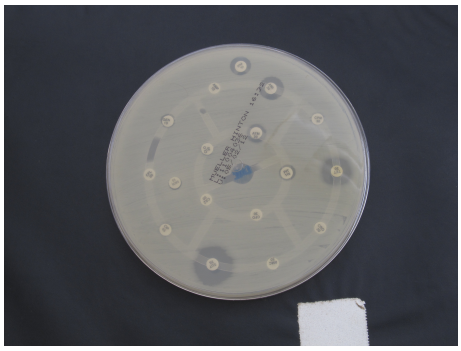


Figura 3.8: Imagem de antibiograma a ser segmentada.

Primeiramente, por se tratar de uma imagem RGB, deve-se transformá-la em níveis de cinza, o que pode ser obtido com a análise dos valores de apenas uma de suas "camadas"[1],

[1] As imagens com o sistema de cor RGB são matrizes tridimensionais, cada camada possui os valores de amplitudes dos componentes vermelha, verde e azul que compõe a imagem.

ou com a intensidade retirada de um modelo conceitual[2]. Uma vez realizado este processo, tem-se a imagem da Figura 3.9.

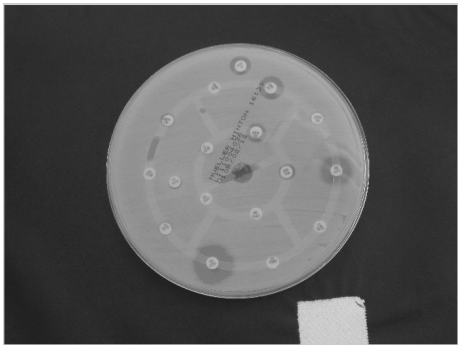


Figura 3.9: Imagem gerada em escala de cinza.

Agora, de posse da imagem em escala de cinza, é utilizado o método de Otsu para determinar um limiar ideal para binarização, que é apresentada na Figura 3.10.

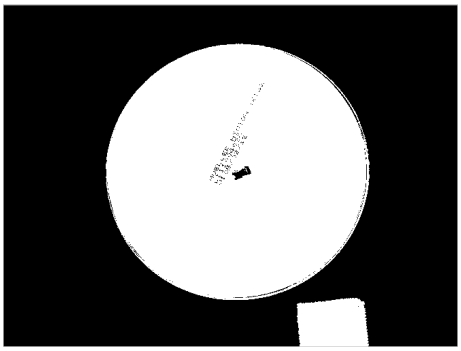


Figura 3.10: Imagem binária gerada a partir do limiar de Otsu.

Para a identificação dos limites de cada objeto, foi utilizada a técnica de rotulagem. Porém, como é possível ver na Figura 3.10, a imagem possui informações indesejadas, como a escrita dentro do disco. Para retirar estes ruídos, pode-se realizar o fechamento da imagem, cujo resultado é mostrado na Figura 3.11.

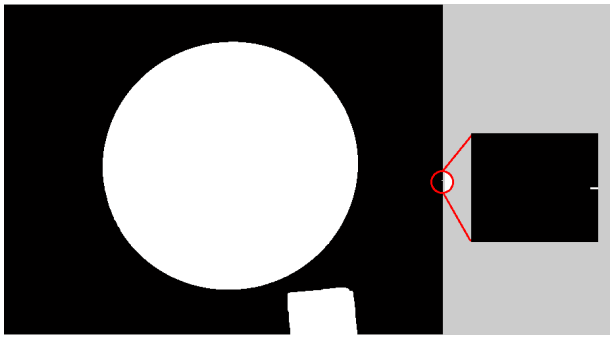


Figura 3.11: Resultado do fechamento com detalhe em destaque.

Entretanto, como é possível ver no detalhe da Figura 3.11, o processo de fechamento introduz pequenas "falhas" na nova imagem, e que podem ser retiradas realizando agora

[2]Veja mais informações sobre os sistemas de cores conceituais no Apêndice B, mais especificamente nos modelos HSL e CIELAB.

a abertura da imagem. Nota-se que para realização destas operações foram utilizadas as funções *imclose* e *imopen* do Matlab por serem mais rápidas do que as desenvolvidas ao longo do capítulo.

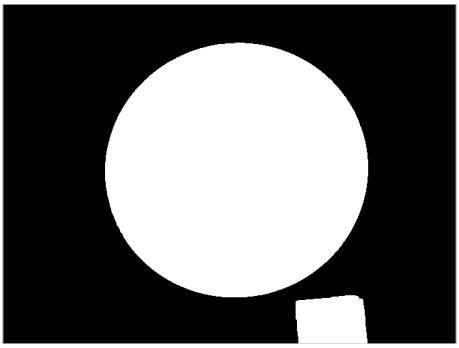


Figura 3.12: Resultado da abertura da imagem da Figura 3.11.

Finalmente, com todos os ruídos minimizados, foi utilizada a rotulagem para identificar os limites do disco. Aqui, foi utilizada a função *bwlabel* do Matlab, que retorna a imagem rotulada. De posse da imagem rotulada, deve-se identificar qual dos rótulos é o do disco, o que pode ser feito de maneira simples se analisarmos quantas vezes cada rótulo foi utilizado e escolhermos o segundo maior destes, o que pode ser feito com uma pequena análise. Neste caso, ele é o segundo maior rótulo da imagem.

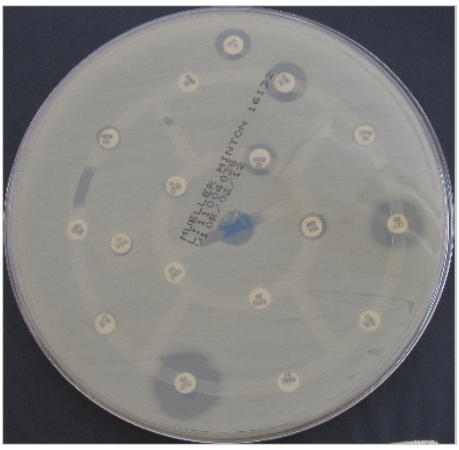


Figura 3.13: Imagem final com disco isolado.

Exercício de fixação

O Algoritmo 3.6 apresenta os passos necessários para isolar os discos, como descrito na Seção 3.4.

```
clear all;
close all;

im_rgb=imread('IMG1.JPG');
imshow(im_rgb)
pause();

```

```
%Cópia de uma das camadas da imagem colorida para obtenção de imagem em
%escala de cinza.
im = im_rgb(:,:,1);
imshow(im)
pause();

%Binarização da imagem
L = graythresh(im)*255;%Obtenção do limiar de Otsu
B = zeros(size(im));
B(im<=L) = 0;
B(im>L) = 255;
imshow(B)
pause();

se = strel('square',50); %Elemento estruturante a ser utilizado nas
                         %funções de abertura e fechamento
R = imclose(B,se); %Fechamento da imagem

imshow(R)
pause();

R = imopen(R,se); %Abertura da imagem

imshow(R)
pause();

L = bwlabel(R,4); %Rotulagem com vizinha de 4

%Obtendo o número de vezes que cada rótulo apareceu
q0 = max(find(L == 0)); %Quantidade de vezes que o rótulo 0 aparece
q1 = max(find(L == 1)); %Quantidade de vezes que o rótulo 1 aparece
q2 = max(find(L == 2)); %Quantidade de vezes que o rótulo 2 aparece

quantities = sort([q0 q1 q2],'descend'); %Organização dos valores de cada
                                         %rótulo em ordem descrescente
                                         %para obter o segundo maior

disk_label = [];

switch quantities(2) %quantities(2) será o segundo maior elemento do vetor
    case(q0)
        disk_label = 0;

    case(q1)
        disk_label = 1;

    case(q2)
        disk_label = 2;
end

[x,y] = find(L == disk_label); %Obtenção das posições do rótulo identificado como
    segundo maior

cut = im_rgb(min(x):max(x),min(y):max(y),:); %Obtenção do corte da imagem do disco

figure(1)
imshow(im_rgb);

```

```
figure(2)
imshow(cut);
```

Algoritmo 3.6: Exemplo de segmentação de uma placa de Petri.

3.5 Exercícios propostos

1. Caso seja necessário segmentar a Figura 3.14 e isolar apenas o triângulo inscrito, a técnica mostrada ao final do capítulo (Algoritmo 3.6) seria eficiente? Por quê? Caso não seja, desenvolva uma solução para este problema.

Figura 3.14: Imagem para detecção de borda.

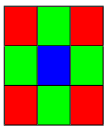
Capítulo 4

Representação e Descrição de Imagens

Leonardo Alves da Costa
Thales Portela

Existem várias formas de representar uma cor, normalmente, os tons de cinza são representados por valores no intervalo $[0, 1]$. Para imagens coloridas, há sempre um sistema de cor associado a três valores distintos dentro do mesmo intervalo. Na computação, normalmente é utilizado o sistema de cor RGB com valores no intervalo $[0, 255]$[1].

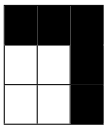
A Figura 4.1, apresenta a disposição de três imagens compostas por 9 pixels (3 linhas e 3 colunas) bem como sua respectiva matriz de representação digital.



(a)

255,0,0	0,255,0	255,0,0
0,255,0	0,0,255	0,255,0
255,0,0	0,255,0	255,0,0

(b)



(c)

1	1	1
0	0	1
0	0	1

(d)

Figura 4.1: Representação de uma imagem em matriz

4.1 Descritores de imagens

Um descritor de imagem é caracterizado por um procedimento de extração de um atributo da imagem [**?**]. Assim, eles podem ser definidos como um par composto por um vetor de características extraídas e uma função de distância.

O vetor de características representa um conjunto de propriedades de uma determinada imagem (cor, textura e forma, por exemplo) e a função de distância diz respeito a dissimilaridades entre duas imagens que tem como base suas características. Estas distâncias podem ser mensuradas por meio de métricas como, por exemplo, a Euclidiana [**?**].

[1] Veja mais sobre os sistemas de cores no Apêndice B.

4.2 Códigos de Cadeia

O código de cadeia é usado na representação da fronteira por uma sequencia conectada de segmentos de linha reta de determinado tamanho e direção. Essa representação baseia-se tipicamente nos vizinhos-de-4 ou nos de-8 dos segmentos [**?**].

Como as imagens digitais são adquiridas em uma grade com espaçamentos iguais nas coordenadas x e y, a cadeia pode ser gerada seguindo a fronteira. Contudo, esta abordagem pode ser problemática por dois motivos [**?**]:

1) A cadeia de códigos resultante é usualmente muito longa; e,

2) Qualquer pequena perturbação ao longo da fronteira, devido à ruído ou à segmentação imperfeita, causa mudanças no código que podem não estar necessariamente relacionadas a forma da fronteira.

A abordagem frequentemente utilizada baseia-se na reamostragem da fronteira. Nesta abordagem, seleciona-se uma grade com o espaçamento maior do que o da imagem, quando a fronteira é percorrida, cada ponto da fronteira é atribuído a cada nó da grade maior, de acordo com a proximidade da fronteira original em relação àquele nó [**?**].

Exercício de fixação

4.3 Descritores de fronteiras

4.3.1 Comprimento do contorno

O *comprimento* de um contorno é um de seus descritores mais simples. Uma simples operação de contagem dos pixels ao longo do contorno fornece uma aproximação do comprimento. No caso de uma curva representada pelo código da cadeia com espaçamento unitário em ambas as direções[2], o número de componentes horizontais e verticais mais $\sqrt{2}$ vezes o número de componentes diagonais fornece o tamanho exato [**?**].

O diâmetro de uma fonteira B é definida como:

$$\text{Diametro}(B) = \max_{i,j}[D(p_i,p_j)] \tag{4.1}$$

sendo que D é a medida de distância e p_i e p_j são pontos no fronteira. O valor do diâmetro e a orientação da linha que conecta da linha que conecta os dois pontos extremos que definem o diâmetro (essa linha é chamada de *maior eixo* da fronteira) são descritores úteis da fronteira [**?**].

4.3.2 Curvatura

A *curvatura* é definida pela taxa de mudança de inclinação. Em geral, a obtenção de medidas confiáveis da curvatura de um ponto em uma fronteira digital é difícil, visto que tais fronteiras tendem a ser localmente "áspera". Entretanto, a utilização da diferença entre as inclinações de segmentos adjacentes de fronteira (que tenham sido representados como segmentos de retas) como um descritor da curvatura no ponto de interseção dos segmentos pode ser útil. Na medida que a fronteira é atravessada no sentido horário, diz-se que um

[2] Direções horizontais e verticais

vértice p é parte de um segmento *convexo* se a mudança de inclinação em p for não-negativa; caso contrário, diz-se que p pertence a um segmento *côncavo*. A descrição da curvatura em um ponto pode ser adicionalmente refinada fazendo-se a mudança da inclinação por intervalos. Por exemplo, p pode ser parte de um segmento quase reto se a mudança for menos que 10° ou um canto (vértices) se a mudança exceder 90°. Note, no entanto, que esses descritores devem ser usados com cuidado, uma vez que sua interpretação depende do comprimento dos segmentos individuais em relação ao comprimento total da fronteira [**?**].

4.3.3 Número de formas

A primeira diferença de uma fronteira representada pelo código da cadeia dependente do ponto inicial. O *número de forma* de tal fronteira, baseada na conexão-de-4 da <*apresentar a figura aqui*>, é definido como a primeira diferneça de menor magnitude. A *ordem* n de um número de forma é definido como o número de dígitos em sua representação. Além disso, n é par para uma fronteira fechada, sendo que seus valores limitam o número de diferentes formas possíveis [**?**].

Como indicado previamente, o *maior eixo* de uma fronteira é o segmento de linha reta que une os dois pontos mais distantes nela. O *menor eixo* é perpendicular ao maior eixo, de maneira que uma caixa retangular que envolve a fronteira pode ser definida em função desses eixos. A razão entre o maior e o menor eixo é a *excentricidade* da fronteira, enquanto que a caixa acima descrita é chamada de *retângulo básico*. Na maioria dos casos, um único número de formas é obtido alinhando-se a grade do código da cadeia com os lados do retângulo básico.

Exercício de fixação

<*pendente*>

4.4 Descritores regionais

A *área* de uma região é definida pelo número de pixels contidos dentro de sua fronteira. O *perímetro* é o tamanho de sua fronteira. Embora a área e o perímetro possam ser usados como descritores, eles são geralmente aplicados a situações em que o tamanho dos objetos não varie. Um uso mais frequente desses descritores é na medida da *compacidade* de uma região, definida como (perímetro)^2/área. A compacidade é a quantidade sem dimensão (portanto, invariante à mudanças de escala), sendo mínima para uma região em forma de disco. Com exceção dos erros introduzidos pela rotação em uma região digital, a compacidade é também invariante à orientação [**?**].

4.4.1 Descritores topológicos

As propriedades topológicas são úteis para descrições globais de regiões no plano da imagem. Assim, elas não são afetadas por deformações.

<*pendente de exemplos*>

Exercício de fixação

<pendente>

4.4.2 Momentos

No caso de uma função contínua bidimensional $f(x,y)$ (também chamado de momento puro), o momento de orgem $(p+q)$ é definido como

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{4.2}$$

para $p,q = 0,1,2,\cdots$ Ao adaptar esta equação para uma imagem escalar (escala de cinza) com as intensidades dos pixels em $I(x,y)$, o momento puro da imagem Mij é calculado por:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \tag{4.3}$$

Em alguns casos, pode ser considerado o cálculo da função de densidade de probabilidade, por exemplo, dividir a funlção Equação 4.3 por:

$$\sum_x \sum_y I(x,y) \tag{4.4}$$

Momento central

Os momentos centrais são invariantes à transladação. Ele é definido como:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{4.5}$$

de modo que $\bar{x} = \frac{M_{10}}{M_{00}}$ e $\bar{y} = \frac{M_{01}}{M_{00}}$ são componentes da centroide. Se $f(x,y)$ é uma imagem digital, então a Equação 4.2 se torna:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y) \tag{4.6}$$

Os momentos centrais até a 3° ordem são:

$$\begin{aligned}
\mu_{00} &= M_{00} \\
\mu_{01} &= 0 \\
\mu_{10} &= 0 \\
\mu_{11} &= M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10} \\
\mu_{20} &= M_{20} - \bar{x}M_{10} \\
\mu_{02} &= M_{02} - \bar{y}M_{01} \\
\mu_{21} &= M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2 M_{01} \\
\mu_{12} &= M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2 M_{10} \\
\mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2 M_{10} \\
\mu_{03} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2 M_{01}.
\end{aligned}$$

E pode ser mostrado que:

$$\mu_{pq} = \sum_{m}^{p} \sum_{n}^{q} \binom{p}{m} \binom{q}{n} (-\bar{x})^{(p-m)} (-\bar{y})^{(q-n)} M_{mn} \tag{4.7}$$

Momentos invariantes de escala

Momentos η_{ij} que $i + j \geqslant 2$ podem ser construídos para serem invariantes tanto de transladação quanto alterações em escala pela divisão do momento central pelo momento 00 propriamente escalonado, conforme a Equação 4.8:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}} \tag{4.8}$$

Momentos invariantes de rotação

É possível calcular momentos invariantes em transladação, escala, e de rotação. Os mais utilizados são o conjunto de Hu de momentos invariantes:

$$\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].
\end{aligned}$$

Exercício de fixação

<pendente>

4.5 Exercícios Propostos

<pendente>

Referências Bibliográficas

[1] John W. EATON. Gnu octave, June 2012. Acesso em: 13 jun. 2013.

[2] R.C. GONZÁLEZ and R.E. WOODS. *Digital image processing*. Prentice Hall, 2002.

[3] A. D. P. F. F. SOUSA. Análise da influência dos descritores visuais mpeg-7 no processamento de consultas por similaridade em cbir aplicado à biblioteca digital, 2009.

[4] R. TORRES, A. FALCÃO, and I. DINSTEIN. Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, 13(2):161–185, 2006.

Apêndice A

GNU Octave

Leonardo Alves da Costa

O GNU Octave é uma linguagem interpretada de alto nível, que se destina principalmente para computações numéricas. Ele fornece recursos para a solução numérica de problemas lineares, não-lineares e para a realização de outros experimentos numéricos. Ele também fornece recursos gráficos abrangentes quem permitem a visualização e manipulação de dados. Octave é usado normalmente por meio de sua interface de linha de comando interativa, mas também pode ser usado para escrever programas não-interativos. A linguagem Octave é bastante semelhante ao Matlab para que a maioria dos programas sejam facilmente portáveis [**?**].

O GNU Octave é distrbuído sobre os temos da licença GNU General Public License e está disponível em http://www.gnu.org/software/octave.

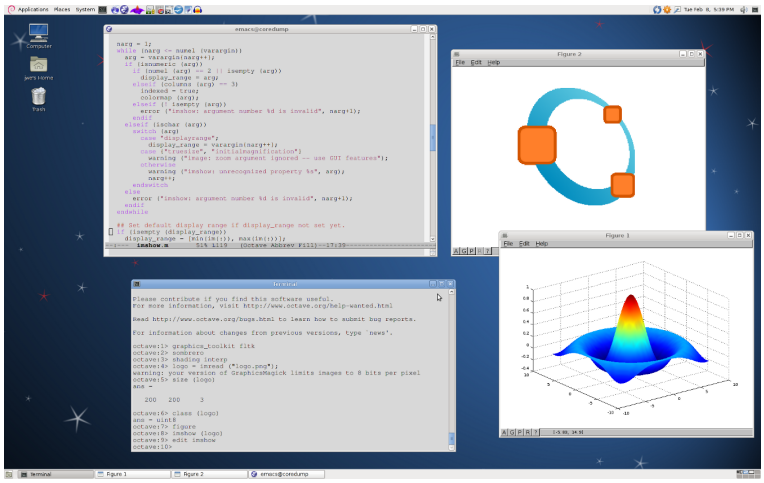


Figura A.1: GNU Octave [**?**].

Para aqueles que não podem instalar programas em seus computadores, ou não se sentem seguros para realizar uma instalação de software, uma boa solução seria o site Verbosus, disponível em https://www.verbosus.com. Ele fornece um ambiente para o GNU Octave em seu site de forma gratuita. A única necessidade de seus usuários é um cadastro gratuito.

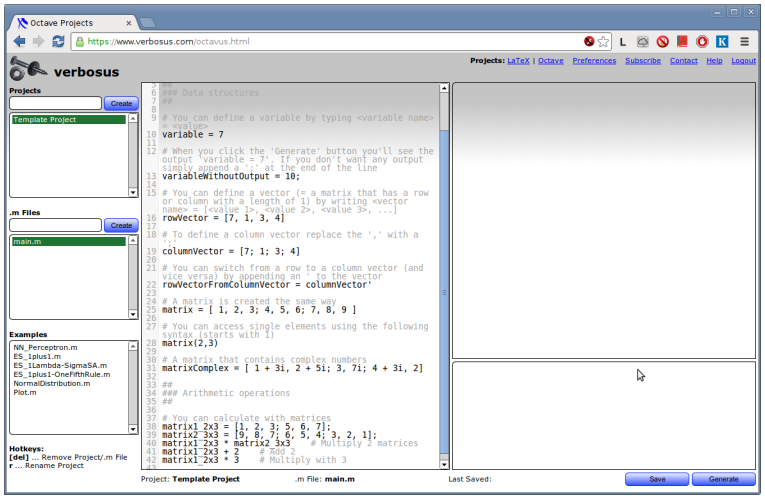


Figura A.2: O ambiente de desenvolvimento da Verbosus.

A.1 Método de instalação

O GNU Octave armazena os seus arquivos de instalação e código fonte no seu projeto hospedado no site sourceforge.net. Os arquivos indicados nesta seção foram retirados dele, ou seja, do site http://sourceforge.net/projects/octave.

A.1.1 Windows

A.1.2 Linux

Existe uma quantidade grande de versões do GNU Linux, o que impede que haja um padrão na instalação do Octave neste sistema operacional. Contudo, todas as distribuições atuais possuem pacotes dele em sua própria instalação.

A.1.3 Mac OS X

Não existe uma compilação própria do GNU Octave em seu site oficial. Contudo, o Octave-Forge fez um pacote para isso e pode ser encontrado em seu site, hospedado no SourceForge.

- Efetue o *download* do arquivo octave-3.4.0-i386.dmg.

Apêndice B

Sistemas de Cores

Leonardo Alves da Costa

Embora amplamente estudado, a percepção das cores pelo cérebro humano ainda não é completamente conhecido. Desta forma, a natureza física das cores é expressa em uma base formal suportada por resultados de experimentais e teóricos [**?**, p. 157].

Basicamente, as cores que os seres humanos percebem são determinadas pela natureza da luz refletida em um objeto. Assim, a Figura B.1 demostra o espectro de luz visível pelos seres humanos [**?**, p. 157].

Figura B.1: Especro de cores visíveis ao olho humano.

Os sistemas de cores normalmente são representadas pela intensidade de três valores distintos, chamados de *triestímulo*. Nos sistemas *subtrativos* e *aditivos*, são três cores distintas. Contudo, elas podem ser representadas pelo seu *brilho*, *matiz* e *saturação*. O brilho está relacionada a intensidade da cor, a matiz ao comprimento de onda dominante da cor e a saturação a pureza da cor. Vale ressaltar que a combinação da matiz com a saturação também é chamada de *cromaticidade*.

Ao referir-se as cores, elas podem ser divididas em sistemas de cores *adivitas* e *subtrativas*.

O sistema aditivo é aquele formado pelas três cores primárias da luz: azul-violeta, vermelho e verde, decompostas a partir da luz branca solar. Chama-se aditivo porque a adição das três cores primárias formam a luz branca.

Em contrapartida, o sistema subtrativo possui as cores primárias ciano, magenta e amarelo como cores primárias. Ele é um sistema que subtrai as cores de acordo com a combinação de suas cores primárias. Assim, a soma das três cores primárias originam-se o preto, ao invés do branco no sistema aditivo.

O olho humano consegue perceber cores que possuem comprimentos de onde que vão de 380nm (nanômetros) a 780nm. Abaixo de 380nm estão os raios ultravioleta e acima de 780nm a radiação infravermelha.

O preto e o branco não são cientificamente consideradas cores. O branco é o resultado da soma de todos os comprimentos de onda e o preto é a ausência completa da luz, portanto da cor.

B.1 Modelo XYZ

O CIE 1931, ou somente XYZ, foi criado pelo Comissão Internacional de Iluminação (CIE) e é um sistema aditivo que descreve as cores por meio de 3 cores primárias virtuais X, Y e Z. Esse foi criado devido à inexistência de um conjunto finito de cores primárias que produza todas as cores visíveis possíveis.

Nesse sistema, as cores C_l podem ser expressas pela seguinte equação:

$$C_l = xX + yY + zZ \tag{B.1}$$

em que x, y e z especificam as quantidades das primárias padrão necessárias para descrever uma cor espectral. A normalização dessa quantidade em relação à luminância $(X + Y + Z)$ possibilita a caracterização de qualquer cor. As cores desse sistema podem ser expressas pela forma normalizadas, conforme a Equação B.2:

$$x + y + z = 1$$

$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad z = \frac{Z}{X+Y+Z} \tag{B.2}$$

Assim, qualquer cor pode ser definida por meio de X e Y que, por dependerem apenas do matiz e da saturação, são chamadas de coordenadas de cromaticidade. Essas coordenadas descrevem a direção do raio produzido por uma fonte de luz real e podem ser aplicadas a qualquer sistema de 3 estímulos. Esse raio deve estar contido no cone de cores reais e, por isso, deve intersectar o plano unitário $X + Y + Z = 1$. A descrição completa de uma cor é dada pelas coordenadas de cromaticidade e pelo valor de um dos 3 estímulos originais, normalmente do Y, que contém a informação de luminância. Essa descrição possibilita a obtenção das quantidades de X e Z por meio da Equação B.3:

$$X = \frac{x}{y}Y \quad Z = \frac{z}{y}Y, \quad \text{onde} \quad z = 1 - x - y \tag{B.3}$$

O sistema XYZ é formado por cores imaginárias definidas matematicamente. Nesse sistema, as combinações de valores negativos e outros problemas relacionados à seleção de um conjunto de primárias reais são eliminados. Com as coordenadas de cromaticidade X e Y é possível representar todas as cores num gráfico bidimensional. O traçado das quantidades de X e Y normalizadas para as cores no espectro visível resulta na curva apresentada na Figura B.2.

A conversão entre o sistema de cor RGB e o XYZ pode ser feito por meio de matrizes. Entretanto, não há uma conversão geral, pois o sistema RGB é dependente de dispositivo. Assim, segue abaixo uma tabela de matrizes conhecidas:

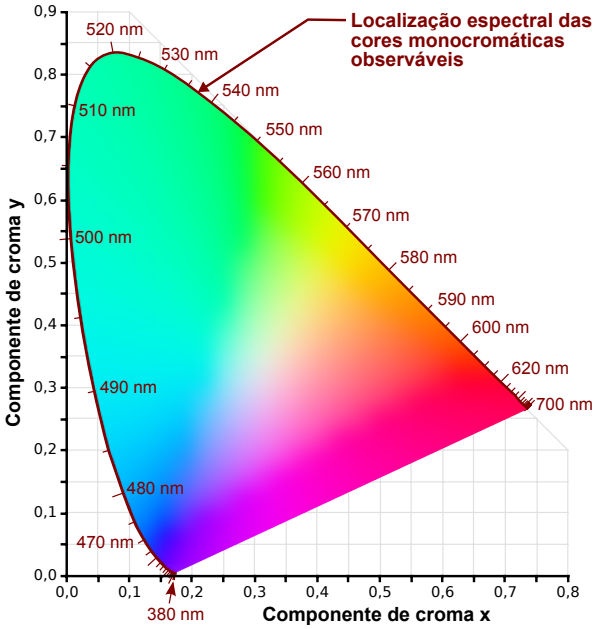


Figura B.2: Sistema de cor XYZ.

$$
\begin{aligned}
\text{CIE RGB} &= \begin{bmatrix} 0.4887180 & 0.3106803 & 0.2006017 \\ 0.1762044 & 0.8129847 & 0.0108109 \\ 0.0000000 & 0.0102048 & 0.9897952 \end{bmatrix} \\
\text{CIE RGB}^{-1} &= \begin{bmatrix} 2.3706743 & -0.9000405 & -0.4706338 \\ -0.5138850 & 1.4253036 & 0.0885814 \\ 0.0052982 & -0.0146949 & 1.0093968 \end{bmatrix} \\
\text{RGB D50} &= \begin{bmatrix} 0.6326696 & 0.2045558 & 0.1269946 \\ 0.2284569 & 0.7373523 & 0.0341908 \\ 0.0000000 & 0.0095142 & 0.8156958 \end{bmatrix} \\
\text{RGB D50}^{-1} &= \begin{bmatrix} 1.7552599 & -0.4836786 & -0.2530000 \\ -0.5441336 & 1.5068789 & 0.0215528 \\ 0.0063467 & -0.0175761 & 1.2256959 \end{bmatrix}
\end{aligned} \tag{B.4}
$$

E a conversão entre os sistemas de cores pode ser dada conforme a Equação B.5. Considere $[M]$ como a matriz de conversão Equação B.4:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{B.5}
$$

B.2 Sistema RGB

É um sistema aditivo mais conhecido ultimamente devido a sua grande utilização na informática. Ele possui as cores primárias vermelho, verde e azul, conforme a Figura B.3.

Vale ressaltar que as cores resultantes são chamadas de cores secundárias da luz e são ao mesmo tempo as cores primárias do sistema subtrativo.

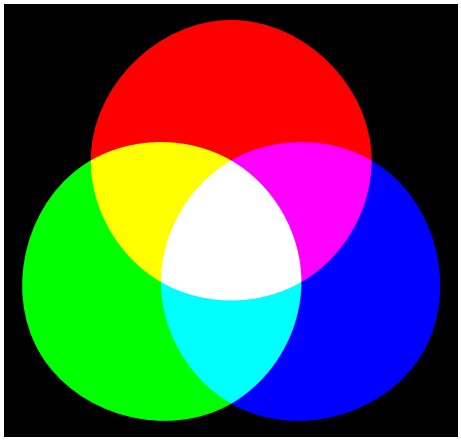


Figura B.3: Sistema de cor RGB

B.3 Sistema CMY

É um sistema subtrativo muito utilizado em pigmentos e tintas. Suas cores primárias são o ciano, magenta e amarelo. Seu modelo é apresentado na Figura B.4.

Figura B.4: Sistema de cor CMY

Em alguns casos, o sistema CMY é apresentado como CMYK. O componente "K" representa o preto (*black*) e é utilizado para a economia de tinta. É mais barato imprimir tons de cinza, ou para mudar a saturação de uma cor, do que a mistura dos três componentes primários. Assim, ele não é efetivamente um sistema de cor.

A equação de conversão entre o sistema RGB e o CMY segue conforme a Equação B.6.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1{,}0 \\ 1{,}0 \\ 1{,}0 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{B.6}$$

B.4 Sistema HSL

Gonzalez p. 310

B.5 Sistema CIELAB

O espaço de cor Lab é baseado no sistema CIE XYZ. O componente L é a luminância e a junção dos componentes a e b são a croma.

Também é chamado de CIE 1976 (L^*, a^* e b^*), ou simplesmente CIELAB. Ele é independente de dispositivo e foi projetado para se aproximar a visão humana, ou seja, ele leva em consideração a luminância de cada cor na sua conversão. Em sua definição, o $L^* = 0$ é preto e o componente $L^* = 100$ é branco difuso, o branco especular pode ter valores maiores. O componente a^* com valores negativos está associado a cor verde enquanto que valores positivos indicam o magenta. Por último, o componente b^* com valores negativos indicam a cor azul e os valores positivos indicam a cor amarela.

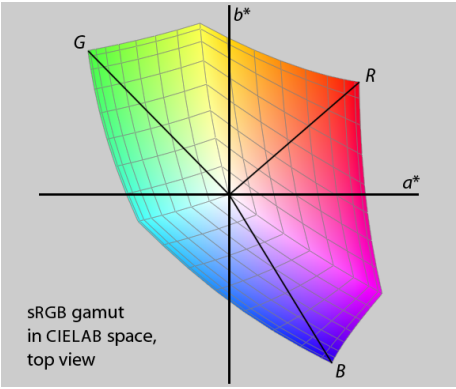


Figura B.5: Gamma RGB projetado no CIELAB

CIEXYZ para CIELAB

O X_n, Y_n e Z_n são valores triestímulos CIE XYZ referentes ao ponto branco (o n sugere "normalizado").

$$\begin{aligned} L^* &= 116 f(Y/Y_n) - 16 \\ a^* &= 500[f(X/X_n) - f(Y/Y_n)] \\ b^* &= 200[f(Y/Y_n) - f(Z/Z_n)] \\ f(t) &= \begin{cases} t^{1/3}, & \text{se } t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t, & \text{caso contrário} \end{cases} \end{aligned} \tag{B.7}$$

A divisão do domínio da função $f(t)$ em duas partes foi feita para prevenir o declive infinito ao $t = 0$. O $f(t)$ foi assumido para ser linear abaixo de $t = t_0$ e é assumido que $t^{1/3}$ seja parte da função ao t_0 em ambos valores do declive. Em outras palavras:

$$\begin{aligned} t_0^{1/3} &= a t_0 + b, && \text{combina em valor} \\ \tfrac{1}{3} t_0^{-2/3}, & && \text{combina com o declive} \end{aligned} \tag{B.8}$$

A interseção $f(0) = b$ foi escolhida para que L^* seja 0 para $Y = 0$ e $b = 16/116 = 4/29$. A Equação B.8 podem ser resolvidas para a e t_0 conforme a Equação B.10:

$$\begin{aligned} a &= \frac{1}{3}\delta^{-2} = 7{,}787037\cdots \\ t_0 &= \delta^3 = 0{,}00856\cdots \end{aligned} \tag{B.9}$$

com $\delta = 6{,}29$.

Transformação inversa

A transformação inversa é mais facilmente expressada com o uso da função inversa de f:

$$\begin{aligned} Y =& Y_n f^{-1}\left(\frac{1}{116}(L^* + 16)\right) \\ X =& X_n f^{-1}\left(\frac{1}{116}(L^* + 16)\right) + \frac{1}{500}a^* \\ Z =& Z_n f^{-1}\left(\frac{1}{116}(L^* + 16)\right) - \frac{1}{200}b^* \end{aligned} \tag{B.10}$$

de modo que,

$$f^{-1}(t) = \begin{cases} t^3, & \text{se } t > \frac{6}{29} \\ 3\left(\frac{6}{29}\right)^2\left(t - \frac{4}{29}\right), & \text{caso contrário} \end{cases} \tag{B.11}$$

Apêndice C

A Transformada de Fourier

Marcelino Monteiro de Andrade
Leonardo Alves da Costa

No espaço vetorial de Hilbert é definida a transformada de Fourier. Logo, considerando inicialmente uma função f, contida no espaço de Hilbert das funções quadraticamente integráveis $L_2(R)$, e estando a energia da função f limitada ao longo de todo o eixo dos reais é definida conforme a Equação C.1:

$$\int_{-\infty}^{+\infty} |f(t)|^2 \, \mathrm{d}t < \infty \tag{C.1}$$

O produto escalar e a norma para o espaço $L_2(R)$ são dados pela Equação C.2:

$$\langle f, \psi \rangle = \int_{-\infty}^{+\infty} f(t)\psi^* \mathrm{d}t < \infty \text{ e } \|\psi\| = \sqrt{\langle \psi, \psi \rangle} \tag{C.2}$$

O resultado do produto escalar entre f e ψ corresponde a operação de projeção de f sobre ψ. Que assume um conjunto ortogonal e com norma unitária de funções $\psi_l(t)$, $l \in \mathbb{N}$ no espaço $L_2(R)$, chamado de conjunto ortonomal, tem-se a Equação C.3:

$$\langle \psi_m, \psi_n \rangle = \begin{cases} 0, & \text{para } m \neq n \\ 1, & \text{para } m = n \end{cases} \tag{C.3}$$

A função f pode ser definida como uma expansão em série temporal ortonormal, representada por uma combinação linear de funções ortonormais de base $\psi_l(t)$, ponderadas pelas projeções de f sobre cada uma delas, isto é:

$$f(t) = \sum_l \langle f, \psi_l \rangle \, \psi_l(t) = \sum_{k=-\infty}^{+\infty} a_l \psi_l(t) \tag{C.4}$$

onde $a_l = \langle f, \psi_l \rangle$ com $l \in \mathbb{N}$, define a transformada da função f. Na Transformada de Fourier, pode-se constatar que as funções de base $\psi_l(t)$ são exponenciais complexas obtida a partir de dilatações em freqüência,

$$\mathcal{F}(w) = \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-jwt}\mathrm{d}t \tag{C.5}$$

Essa abordagem implica que a função base $\psi_l(t)$ possui suporte igual a toda extensão do eixo real, isto é, cada função de base decai a zero somente nos limites $\pm\infty$.

C.1 Sistemas multidimensionais em espaço discreto

Um sistema multidimensional em espaço discreto é caracterizado por um operador que efetua o mapeamento de um sinal de entrada x, com mais de uma dimensão, em outro de saída y. A Figura C.1 ilustra esse efeito para um sistema em duas dimensões, no qual um sinal $x(n_1,n_2)$ de entrada sofre uma transformação $T[\cdot]$ e origina como saída um sinal $y(n_1, n_2)$.

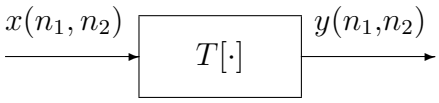


Figura C.1: Representação de um sistema bidimensional em espaço discreto

O operador $T[\cdot]$ representa a regra ou um conjunto de regras de mapeamento do sinal de entrada para o sinal de saída, o que pode, por exemplo ser a representação de um filtro passa-baixas aplicado em Processamento Digital de Imagens (PDI), de acordo com:

- $x(n_1,n_2)$: a imagem original;
- $y(n_1,n_2)$: a imagem filtrada; e,
- $T[\cdot]$: o operador do filtro.

C.2 Transformada de Fourier Bidimensional

É similar a Equação C.5. Contudo, é definida em domínio discreto. Ela pode associar-se uma função complexa de $x(n_1,n_2)$, expressa por:

$$X(\omega_1,\omega_2) = \sum_{n_1}\sum_{n_2} x(n_1,n_2)\mathrm{e}^{-j\omega_1 n_1 - j\omega_2 n_2} \tag{C.6}$$

Assim, a Equação C.6 é chamada de transformada direta de Fourier em espaço discreto.