André Ricardo Backes
Jarbas Joaci de Mesquita Sá Junior

Introdução à Visão Computacional Usando MATLAB®

ALTA BOOKS
EDITORA
Rio de Janeiro, 2016

SUMÁRIO

PREFÁCIO

Com este livro pretendemos contribuir com o preenchimento de uma lacuna no mercado editorial brasileiro em relação à área de visão computacional. Atualmente, a maioria dos livros disponíveis sobre "processamento digital de imagens" possui um caráter muito mais teórico do que prático, e com pouca ênfase (se existente) na área de "reconhecimento de padrões". Assim, pretendemos fornecer um livro com as seguintes características: fácil entendimento por qualquer leitor com conhecimentos básicos de lógica de programação; foco na parte prática, porém mostrando a teoria essencial ao entendimento dos tópicos abordados; ênfase nos algoritmos de reconhecimento de padrões, que são fundamentais em qualquer sistema de visão computacional; e aplicação imediata dos exemplos do livro no software MATLAB®.

Escolhemos o MATLAB® por ser um software de linguagem de programação fácil e amplamente utilizado por estudantes e profissionais das mais diversas áreas do conhecimento, especialmente computação, engenharia, matemática, física e bioinformática. Ao escolher esse software, deparamo-nos com o problema de qual versão adotar, uma vez que versões antigas algumas vezes podem não reconhecer códigos de versões mais recentes e vice-versa. Para tentar resolver esse problema, preocupamo-nos em colocar no livro somente códigos que abrangessem uma longa faixa de tempo. Desse modo, acreditamos que a maioria dos leitores não terá problema algum em executar os algoritmos.

Acreditamos que este livro possa ser incluído na bibliografia principal da disciplina de processamento digital de imagens (e disciplinas similares), ministradas em cursos de graduação em engenharia de computação, engenharia elétrica, ciência de computação, informática etc., bem como na bibliografia complementar de disciplinas como inteligência computacional, inteligência artificial, redes neurais e computação gráfica (quando aborda processamento de imagens). Além disso, o livro pode ser utilizado em disciplinas de pós-graduação de visão computacional, processamento de imagens e reconhecimento de padrões, bem como pode ser bastante útil a qualquer leitor interessado na área de visão computacional.

Os autores

CAPÍTULO 1

INTRODUÇÃO

1.1 O que é visão computacional?

Podemos definir visão computacional como a área de estudo que tenta repassar para máquinas a incrível capacidade da visão. Quando falamos de visão, não estamos nos referindo apenas ao ato de captar imagens. Apesar de essa capacidade ser impressionante (basta que estudemos um pouco sobre o funcionamento de um olho para constatarmos isso), ela é apenas o início de um processo muito mais vasto e rico. A visão consiste em captar imagens, melhorá-las (por exemplo, com retirada de ruídos, aumento de contraste etc.), separar as regiões ou objetos de interesse de uma cena, extrair várias informações dependendo da imagem analisada, como, por exemplo, forma, cor e textura, e, finalmente, relacionar as imagens com outras vistas previamente.

1.2 Uma brevíssima descrição da visão humana

O processo de visão humana pode ser dividido em algumas etapas básicas. Primeiramente, a luz refletida pelos objetos passa pela córnea e pupila, que é um orifício que regula a entrada de luz nos olhos por meio da variação de seu diâmetro. A seguir, a luz prossegue pelo cristalino, que funciona como uma lente biconvexa que focaliza a luz na retina. Alterações no cristalino causam problemas como miopia, hipermetropia, presbiopia etc. A retina, por sua vez, é uma camada de tecido nervoso no fundo dos olhos constituída por dois tipos de fotorreceptores: *bastonetes* e *cones*. Os bastonetes são células com alta sensibilidade à iluminação, pouca acuidade (ou seja, fornecem imagens de baixa resolução) e que não reconhecem diferenças de cor. Por outro lado, os cones são células que necessitam de maiores intensidades de luz, possuem maior acuidade e reconhecem cores (o daltonismo, que é um distúrbio visual que incapacita diferenciar certas cores, é justamente um problema relacionado aos cones de um indivíduo). A próxima etapa é a condução dos impulsos visuais produzidos pelos fotorreceptores pelo nervo óptico até uma região cerebral denominada *quiasma óptico*, que divide parte dos impulsos de um olho para o hemisfério cerebral de seu mesmo lado e parte dos impulsos para o hemisfério do lado oposto (desse modo, cada hemisfério recebe informações de ambos os olhos). Após essa etapa, os impulsos prosseguem até uma região do tálamo denominada *núcleo geniculado lateral*, que, por sua vez, envia as informações para o *córtex visual*. A Figura 1.1 mostra um esquema do sistema visual humano.

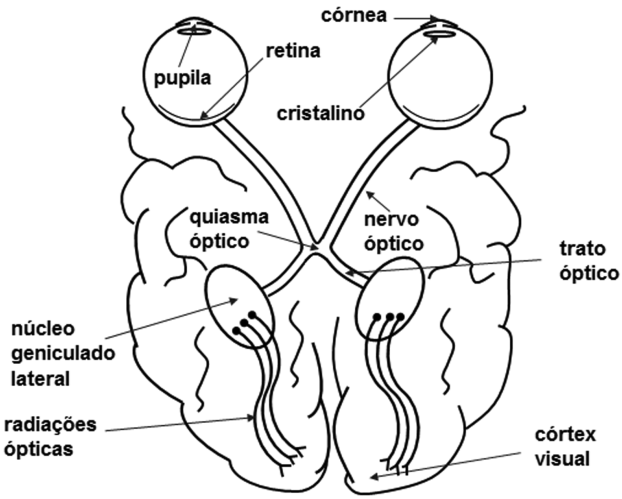

Figura 1.1: Esquema do sistema visual humano.

1.3 Um sistema de visão computacional

Podemos pensar em um sistema de visão computacional como constituído de várias fases. São elas:

- **Aquisição**: responsável pela captação das imagens, ou seja, tenta simular a função dos olhos. Os dispositivos que cumprem esse papel são os *scanners*, filmadoras, máquinas fotográficas etc.

- **Processamento de imagens**: responsável por "melhorar" a imagem, isto é, retirar ruídos, salientar bordas, suavizar a imagem etc. Essa etapa pode ser um fim em si mesma ou ter o propósito de fornecer uma imagem mais adequada para as próximas fases. É importante salientar que essa fase compreende tanto o que usualmente se denomina "pré-processamento", como rotação da imagem, equalização de histograma etc., quanto processamentos mais complexos, como, por exemplo, filtragens e aplicação de operadores morfológicos.
- **Segmentação**: responsável por particionar a imagem em regiões de interesse. Por exemplo, em uma imagem de paisagem, poderíamos estar interessados apenas na porção que representa o céu, ou a vegetação, ou apenas o lago, ou algum cisne nesse lago etc.
- **Extração de características/Análise de imagens**: responsável por obter um conjunto de características do objeto de interesse. Em outras palavras, essa fase é responsável por encontrar uma codificação numérica que represente determinada imagem, como uma espécie de "impressão digital" (analogia imperfeita) que permita identificá-lo.
- **Reconhecimento de padrões**: responsável por classificar ou agrupar as imagens com base em seus conjuntos de características. Por exemplo, se você vir a foto de uma única laranja, provavelmente saberá que aquele objeto pertence à classe "laranja" com base em atributos como cor, rugosidade da casca, formato, tamanho etc. É importante salientar que o objeto visto não é *igual* às laranjas vistas no passado, mas apenas *similar* (na verdade, segundo a filosofia, a igualdade é um conceito teórico que não existe na natureza). No entanto, mesmo com essa limitação conseguimos classificá-lo corretamente na maioria dos casos.

É importante enfatizar que não há uma descrição única para as fases de um sistema de visão computacional. Por exemplo, alguns autores chamam de "pré-processamento" a etapa que denominamos "processamento de imagens", e chamam de "processamento de imagens" todas as cinco fases apresentadas. O número de fases também varia de acordo com o ponto de vista dos autores. Além disso, algumas fases podem ser suprimidas dependendo do problema. Por exemplo, uma imagem pode seguir direto para a fase de extração de características sem passar pela fase de segmentação. Isso pode ocorrer tanto pela segmentação ter sido efetuada manualmente (às vezes esse procedimento é necessário para problemas mais difíceis) quanto porque a imagem original já representa toda a região de interesse para análise. A Figura 1.2 apresenta um esquema simples de um sistema de visão computacional.

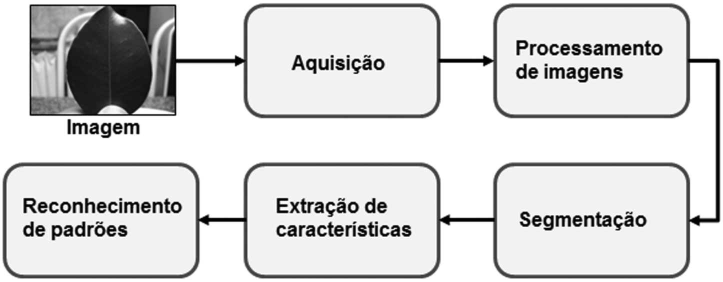

Figura 1.2: Esquema de um sistema de visão computacional.

1.4 Uma brevíssima história da área de visão computacional

Os trabalhos pioneiros sobre visão computacional foram propostos na mesma época do surgimento dos primeiros computadores e essa área de pesquisa evoluiu gradativamente à medida que mais recursos computacionais (por exemplo, memória e capacidade de processamento) se tornaram disponíveis. A seguir, apresentamos alguns trabalhos que consideramos representativos da evolução da área:

- Roberts (1963) propôs um dos primeiros detectores de bordas em sua tese defendida no Massachusetts Institute of Technology (MIT);
- Haralick et al. (1973) propuseram as matrizes de co-ocorrência para a classificação de texturas;
- Daugman (1980, 1985) apresentou modelos matemáticos bidimensionais (filtros de Gabor) para simular o comportamento de campos receptivos do córtex visual;
- Canny (1986) propôs um algoritmo pioneiro para detecção de bordas com alta tolerância a ruídos;
- Kass et al. (1988) propuseram os modelos de contorno ativo (*snakes*);

- Mallat (1987, 1989a) propôs a teoria multirresolução para análise de sinais usando *wavelets*;
- Mumford e Shah (1989) desenvolveram um algoritmo de segmentação em regiões por meio da minimização de um funcional;
- Trabalhos como os de Rowley et al. (1998) e Viola e Jones (2004) propuseram algoritmos para o desafiador problema de reconhecimento facial;
- Lowe (2004) desenvolveu um poderoso algoritmo para extração de características invariantes de uma imagem que permite o *matching* (casamento/combinação) entre objetos em diferentes perspectivas.

Atualmente, a pesquisa em visão computacional se caracteriza por uma grande variedade de algoritmos de alto desempenho designados para problemas específicos, como, por exemplo, reconhecimento facial e de íris, análise de formas, de texturas, segmentação em imagens médicas etc. Entretanto, em se tratando de algoritmos de âmbito genérico, ainda há um longo caminho para que surjam programas que emulem a capacidade da visão biológica.

1.5 Alguns periódicos e eventos da área de visão computacional

Nesta seção apresentamos alguns periódicos que têm trazido contribuições significativas para a área de visão computacional. São eles: *IEEE Transactions on Pattern Analysis and Machine Intelligence*; *IEEE Transactions on Image Processing*; *Pattern Recognition*; *IEEE Transactions on Signal Processing* (antes se chamava *IEEE Transactions on Acoustics, Speech and Signal Processing*); *Pattern Recognition Letters*; *Journal of the Optical Society of America A*; *Signal Processing*; *Information Sciences*; *IEEE Transactions on Systems, Man, and Cybernetics*; *International Journal of Computer Vision*; *Computer Vision and Image Understanding*; *IEEE Transactions on Medical Imaging*; *Medical Image Analysis*; *Journal of Electronic Imaging*; *Journal of Mathematical Imaging and Vision*; *Expert Systems with Applications*; *Machine Vision and Applications*; *International Journal of Pattern Recognition and Artificial Intelligence*; e *International Journal of Imaging Systems and Technology.*

Alguns eventos importantes que contemplam a área de visão computacional são: ICCV (*IEEE International Conference on Computer Vision*), CVPR (*IEEE Conference on Computer Vision and Pattern Recognition*), ICIP (*IEEE International Conference on Image Processing*), ECCV (*European Conference on Computer Vision*), ICPR (*International Conference on Pattern Recognition*), BMVC (*British Machine Vision Conference*), SIBGRAPI (*Conference on Graphics, Patterns and Images*), ICIAP (*International Conference on Image Analysis and Processing*), ICVS (*International Conference on Computer Vision Systems*), WACV (*IEEE Winter Conference on Applications of Computer Vision*), CIARP (*Iberoamerican Congress on Pattern Recognition*), CAIP (*International Conference on Computer Analysis of Images and Patterns*), VISAPP (*International Conference on Computer Vision Theory and Applications*), IWSSIP (*International Conference on Systems, Signals and Image Processing*), WVC (Workshop de Visão Computacional), ACCV (*Asian Conference on Computer Vision*) e IbPRIA (*Iberian Conference on Pattern Recognition and Image Analysis*).

CAPÍTULO 2

FUNDAMENTOS DE MATLAB®

O MATLAB® (do inglês, **MAT**rix **LAB**oratory) é um software de altíssimo desempenho desenvolvido para a realização de cálculos em matrizes. De forma diferente de outras linguagens de programação, o MATLAB® utiliza comandos que são mais próximos da forma como escrevemos expressões algébricas, o que torna mais simples o seu uso. Além disso, ele possui uma vasta quantidade de funções já prontas e agrupadas em *toolboxes* para as mais diversas áreas do conhecimento.

Neste capítulo iremos abordar os fundamentos necessários para utilizar o software MATLAB® como suporte computacional para os estudos na área de processamento de imagens.

2.1 Comandos Iniciais

2.1.1 Criando variáveis e atribuindo valores

Uma das operações mais utilizadas em programação é a operação de atribuição (=). É ela que permite que armazenemos um valor em uma variável. No exemplo abaixo, o valor 5 é atribuído à variável x:

```
>> x = 5
x =
     5
```

Importante: no MATLAB® não é necessário criar uma variável com antecedência. Sempre que atribuímos um valor a uma variável, se ela não existir, o MATLAB® irá criá-la.

O nome de uma variável é um conjunto de caracteres que podem ser letras ou números. Porém, esse nome deve sempre iniciar com uma letra, nunca com um número. Além disso, o MATLAB® é **case-sensitive**, ou seja, uma palavra escrita utilizando caracteres maiúsculos é diferente da mesma palavra escrita com caracteres minúsculos.

Importante: apesar de não termos definido, toda variável possui um **tipo**. É ele quem determina o conjunto de valores e de operações que uma variável aceita, ou seja, que ela pode executar.

Os tipos mais comuns no MATLAB® são:

- **double**: valores numéricos em geral. São valores reais de dupla precisão (64 bits). Podem assumir valores de 10^{-308} a 10^{308}, com 15 a 16 algarismos significativos.

```
>> x = 5
x =
     5
>> y = 2.1
y =
    2.1000
```

  Note que em números reais a parte decimal usa ponto, e não vírgula.
- **char**: são valores escalares de 16 bits, representando um caractere simples.

```
>> letra = 'a'
letra =
a
>> texto = 'matlab'
texto =
matlab
```

  Note que os caracteres sempre ficam entre aspas simples.

Importante: se colocarmos um ponto e vírgula (;) após um comando, esse comando será executado, mas nenhuma saída será apresentada na Janela de Comando (**Command window**).

```
>> x = 4;
>> y = 3.2
y =
    3.2000
```

2.1.2 A variável ans

Vimos que uma variável é criada sempre que atribuímos um valor a ela. Porém, podemos querer calcular uma operação, saber o seu resultado, mas não armazená-lo. Sempre que realizamos uma operação, mas não definimos a variável na qual o resultado será armazenado, o MATLAB® salva o resultado na variável **ans**, como mostra o exemplo a seguir:

```
>> 2 + 3
ans =
     5

>> [1 2; 3 4]
ans =
     1     2
     3     4
```

2.1.3 Trabalhando com matrizes

Definindo uma matriz usando uma lista de valores

Podemos definir uma matriz utilizando uma lista explícita de valores delimitada por colchetes ([ ]). Na matriz, os elementos de uma mesma linha são separados por espaços ou vírgulas (,). Duas linhas são separadas utilizando um ponto e vírgula (;).

Exemplo de matriz 4×3

```
>> A = [16 3 13; 5 11 8; 9 6 7; 4 14 1]
A =
    16     3    13
     5    11     8
     9     6     7
     4    14     1
```

Exemplo de matriz 1×5

```
>> B = [16 3 2 13 1]
B =
    16     3     2    13     1
```

Exemplo de matriz 2×1

```
>> C = [2; 1]
C =
     2
     1
```

Funções para a criação de uma matriz

O MATLAB® possui várias rotinas que permitem a criação automática de uma matriz com valores pré-definidos. Abaixo são apresentadas algumas dessas funções:

zeros(M,N,P,...): cria uma matriz preenchida com "0s". As dimensões da matriz são definidas pela quantidade de parâmetros informados. Exemplos:

```
>> m = zeros(2,3)
m =
     0     0     0
     0     0     0
>> m = zeros(2)
m =
     0     0
     0     0
>> m = zeros(2,1)
m =
     0
     0
```

ones(M,N,P,...): cria uma matriz preenchida com "1s". As dimensões da matriz são definidas pela quantidade de parâmetros informados. Exemplos:

```
>> m = ones(2,3)
m =
     1     1     1
     1     1     1
>> m = ones(2)
m =
     1     1
```

```
      1      1
>> m = ones(2,1)
m =
      1
      1
```

rand(M,N,P,...): cria uma matriz preenchida com números aleatórios seguindo uma distribuição **uniforme**. As dimensões da matriz são definidas pela quantidade de parâmetros informados. Exemplos:

```
>> m = rand(1,4)
m =
    0.9572    0.4854    0.8003    0.1419
>> m = rand(2,4)
m =
    0.4218    0.7922    0.6557    0.8491
    0.9157    0.9595    0.0357    0.9340
>> m = rand(3,1)
m =
    0.6787
    0.7577
    0.7431
```

randn(M,N,P,...): cria uma matriz preenchida com números aleatórios seguindo uma distribuição **normal**. As dimensões da matriz são definidas pela quantidade de parâmetros informados. Exemplos:

```
>> m = randn(1,5)
m =
   -0.3034    0.2939   -0.7873    0.8884   -1.1471
>> m = randn(2,5)
m =
   -1.0689   -2.9443    0.3252    1.3703   -0.1022
   -0.8095    1.4384   -0.7549   -1.7115   -0.2414
>> m = randn(2)
m =
    0.3192   -0.8649
    0.3129   -0.0301
```

eye(N): cria uma matriz identidade de ordem N. Exemplos:

```
>> m = eye(2)
m =
```