Reconhecimento de Padrões em Imagens

Victor Turrisi

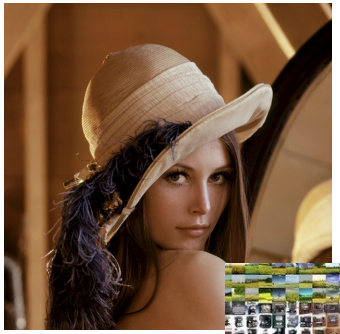
Roteiro

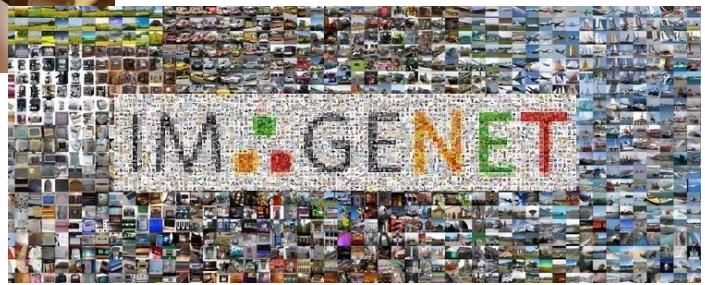
- ➔ Introdução
- ➔ Problema
- ➔ *Pipeline* para reconhecimento de padrões em imagens
  - ◆ Aquisição de imagens digitais
  - ◆ Pré-processamento
  - ◆ Extração de características
  - ◆ Criação de um modelo
  - ◆ Avaliação do modelo
- ➔ Conclusões

Introdução

➔ Imagens e imagens digitais permeiam o nosso cotidiano

➔ Alguns exemplos disso são: jogos digitais, interfaces gráficas, etc.





Introdução

- ➔ Seres humanos são excelentes em reconhecer padrões, sendo a visão o nosso sentido mais avançado

- ➔ Por exemplo, nós somos capazes de reconhecer se uma imagem contém um gato ou um cachorro

- ➔ A posição, cor, cenário, orientação, existência de ruído, ou iluminação de uma imagem raramente fará com que nós não sejamos capazes de distinguir entre um gato ou um cachorro em uma imagem

Introdução

https://github.com/aleju/imgaug

Introdução

- ➔ Apesar de sua eficiência (em detectar padrões em uma imagem), um ser humano tem um alto custo monetário, um baixa velocidade, e a incapacidade de visualizar diversas faixas de frequência

- ➔ Reconhecimento de padrões em imagens digitais se torna uma área de extrema importância para a computação uma vez que é muito interessante automatizar diversos processos que envolvem imagens digitais

- ➔ Algumas possíveis aplicações: veículos autônomos, identificação de pessoas e locais famosos em imagens, identificação de ações em vídeos, indexação automática de conteúdo, aplicações em indústria, entre outros

Problema

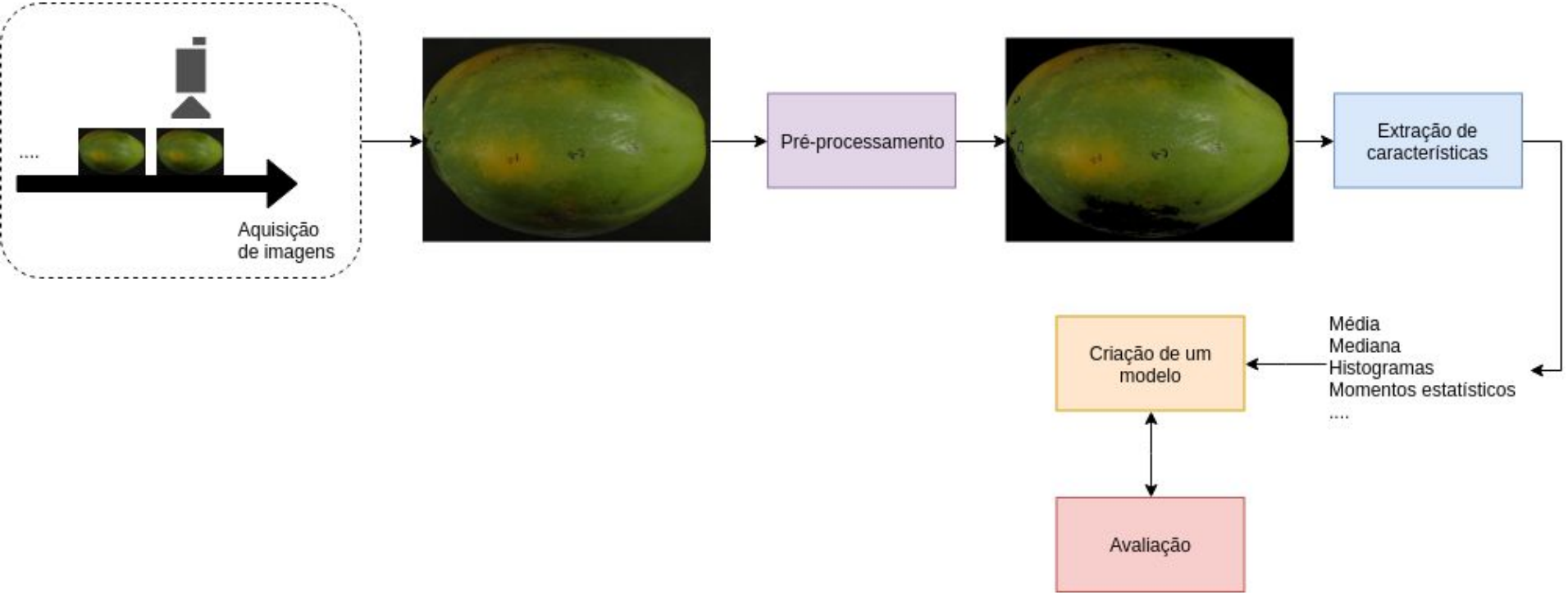
➔ Considere uma linha de produção de mamão que tem como objetivo identificar o grau de amadurecimento de cada fruta (de um a três) para uma tomada de decisão

Problema

Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Pipeline para reconhecimento de padrões



Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Aquisição de imagens digitais

- ➔ Aquisição de uma imagem é um processo de transformar, para o meio digital, uma imagem que antes se encontrava no meio analógico

- ➔ Considerando nosso problema, imagens digitais poderiam ser adquiridas por uma câmera posicionada acima de uma linha de processamento de mamão

- ➔ Para garantir uma boa qualidade da imagem adquirida, essa câmera deve possuir uma boa resolução e ser capaz de tirar fotos rapidamente para não comprometer a linha de produção

- ➔ Após adquirida uma imagem de um mamão, a mesma será utilizada para o próximo passo do *pipeline*: pré-processamento

Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Pré-processamento

- ➔ Técnicas de pré-processamento (em imagens) consistem em um conjunto de ferramentas e métodos para preparar imagens para processos subsequentes

- ➔ Regularizar problemas na aquisição de uma imagem (*e.g.*, iluminação), imagens de tamanhos diferentes ou remover ruído

- ➔ Realçar características de uma imagem, alterar representação dos canais de cores (*e.g.*, RGB para HSV), aplicação de filtros

Pré-processamento

➔ Considerando nosso problema, um pré-processamento simples seria remover efeitos da luz no fundo escuro

```python
import cv2
import numpy as np

img = cv2.imread('em1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img[gray < 37] = 0
cv2.imwrite('em1-fundo-preto.jpg', img)
```

Pré-processamento

- ➔ A função utilizada aplica apenas um valor de *threshold* para converter valores próximos a preto para preto. Entretanto, é possível perceber que, para o problema apresentado, essa técnica gera erros na imagem final

- ➔ Poderíamos utilizar outras técnicas de pré-processamento mais poderosas para segmentar a imagem, e.g., *watershed*, como vimos anteriormente

- ➔ Entretanto, como nosso problema é principalmente baseado na predominância da cor verde ou da cor amarela, as áreas removidas por uma segmentação defeituosa não irão prejudicar a resolução do problema

Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Extração de características

- ➔ Características, *features*, ou descritores, são medidas, nominais ou numéricas que representam algum determinado objeto de interesse, no caso uma imagem

- ➔ Características nominais: a imagem possui algum pixel de intensidade maior ou igual a 200 na cor vermelha

- ➔ Características numéricas: qual o tamanho da imagem? Qual o valor médio de intensidade dos pixels da imagem quando convertida para tons de cinza?

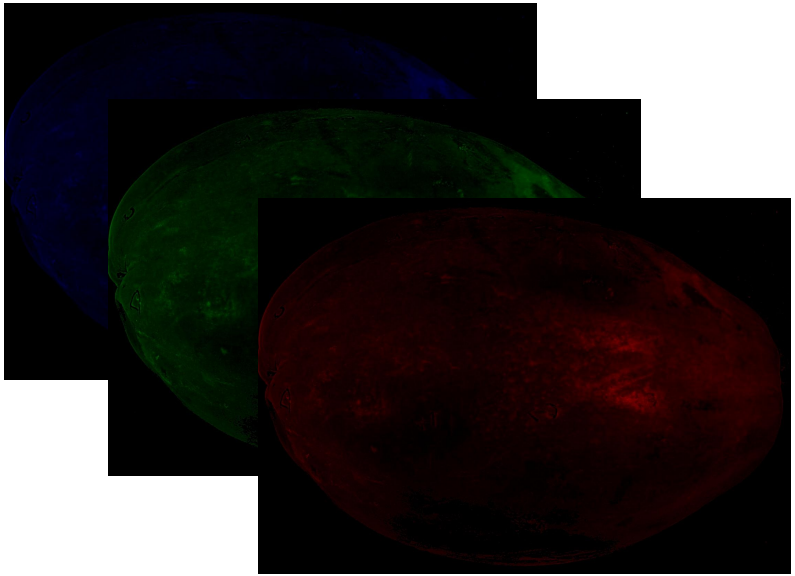
Extração de características

- ➔ Considerando nosso exemplo, iremos extrair algumas características estatísticas com base nos canais RGB

- ➔ Primeiramente, é necessário separar cada canal da imagem

- ➔ Lembrando que uma imagem digital é representada por uma matriz na forma de altura x largura x canais de cores, devemos isolar a terceira dimensão da matriz

Extração de características



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# criação das imagens por canal

img = cv2.imread('em1-fundo-preto.jpg')

new_img = np.zeros(img.shape)
new_img[:, :, 0] = img[:, :, 0]
cv2.imwrite('em1-blue.jpg', new_img)

new_img = np.zeros(img.shape)
new_img[:, :, 1] = img[:, :, 0]
cv2.imwrite('em1-green.jpg', new_img)

new_img = np.zeros(img.shape)
new_img[:, :, 2] = img[:, :, 0]
cv2.imwrite('em1-red.jpg', new_img)
```

Extração de características

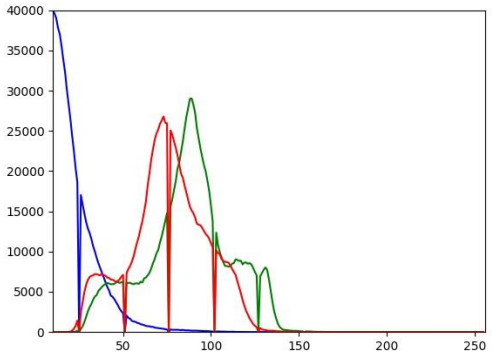
➔ Após isso, podemos computar um histograma para cada canal de cor

➔ Como vimos nas aulas anteriores, um histograma é uma contagem da quantidade de pixels de uma determinada intensidade

Extração de características

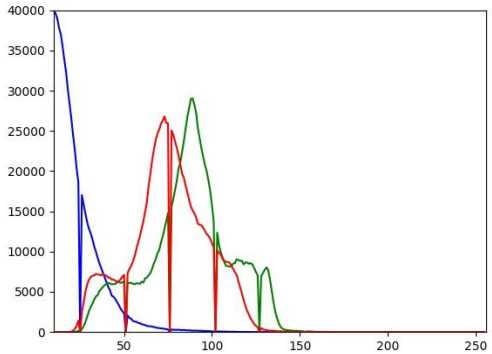
➔ Após isso, podemos computar um histograma para cada canal de cor

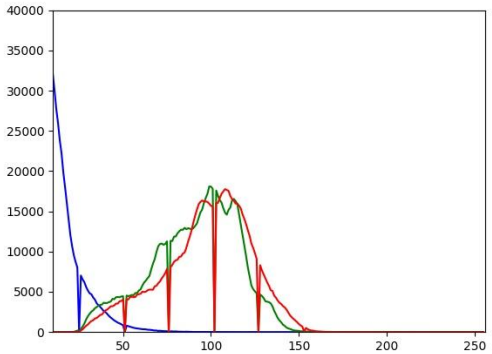
➔ Como vimos nas aulas anteriores, um histograma é uma contagem da quantidade de pixels de uma determinada intensidade

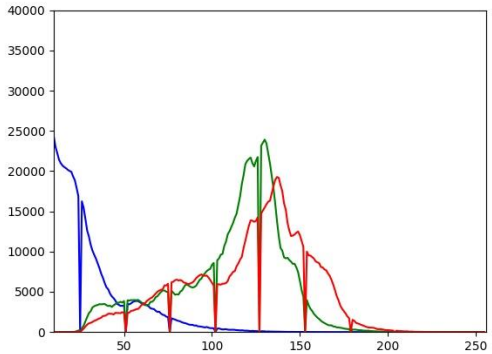



```
for em in range(1, 4):
    img = cv2.imread(f'em{em}-fundo-preto.jpg')
    color = ('b', 'g', 'r')
    for i, col in enumerate(color):
        histr = cv2.calcHist([img], [i], None, [256], [10, 256])
        plt.plot(histr, color=col)
        plt.xlim([10, 256])
        plt.ylim([0, 40000])
    plt.savefig(f'histogram-em{em}.jpg')
    plt.close()
```

Extração de características







Extração de características

➔ Além dos histogramas, podemos ainda extrair características estatísticas adicionais, como média, mediana, moda, assimetria (*skewness*) - terceiro momento estatístico - e curtose (*kurtosis*) - quarto momento estatístico

$$\mathrm{E}[X] = \sum_{i=1}^{k} x_i \, p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k$$

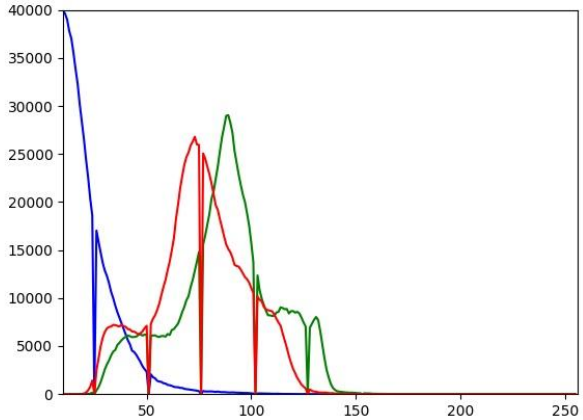
$$\mu_n(x) = \sum p_i (x_i - \mu)^n$$

$$\frac{\mu_n}{\sigma^n} = \frac{\mathrm{E}[(X-\mu)^n]}{\sigma^n}$$

Extração de características

```python
from scipy import stats

for em in range(1, 4):
    img = cv2.imread(f'em{em}-fundo-preto.jpg')
    color = ('b', 'g', 'r')
    for i, col in enumerate(color):
        histr = cv2.calcHist([img], [i], None, [256], [10, 256])

        mean = np.mean(img)
        std = np.std(img)
        mode = np.argmax(histr)
        skew = stats.skew(histr)
        kurt = stats.kurtosis(histr)

        print(f'EM - {em} (color={col})')
        print('mean', mean)
        print('std', std)
        print('mode', mode)
        print('skewness', float(skew))
        print('kurtosis', float(kurt))
        print()
```

Extração de características

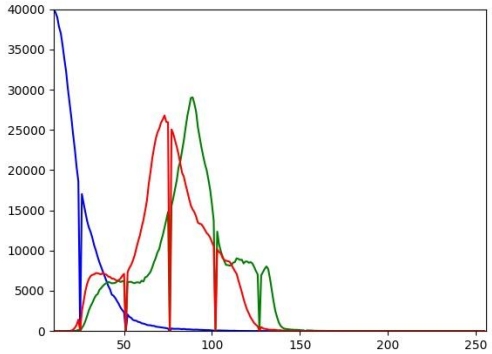


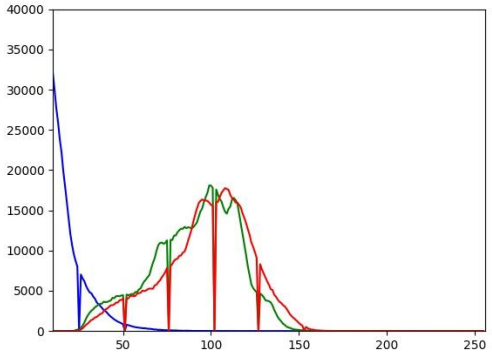

```
EM - 1 (color=r)
mean 45.766601552388124
std 43.28884718885949
mode 73
skewness 1.6352922916412354
kurtosis 1.7245368957519531
```

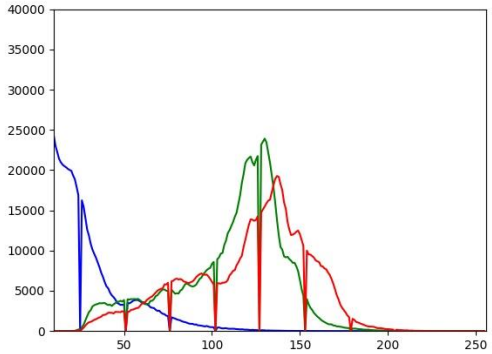
```
EM - 1 (color=g)
mean 45.766601552388124
std 43.28884718885949
mode 89
skewness 1.8341237306594849
kurtosis 2.8446178436279297
```

```
EM - 1 (color=b)
mean 45.766601552388124
std 43.28884718885949
mode 8
skewness 2.599670648574829
kurtosis 5.474271774291992
```

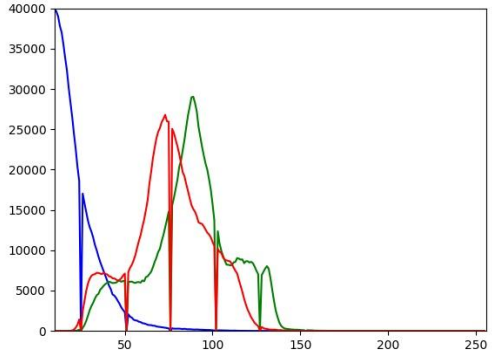
Extração de características



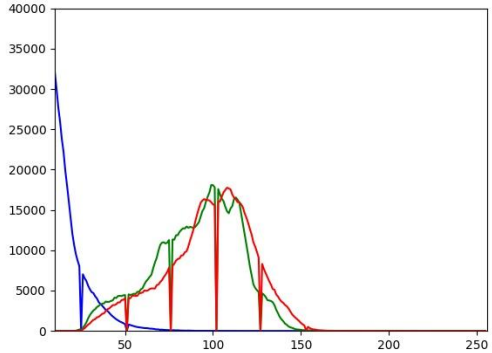




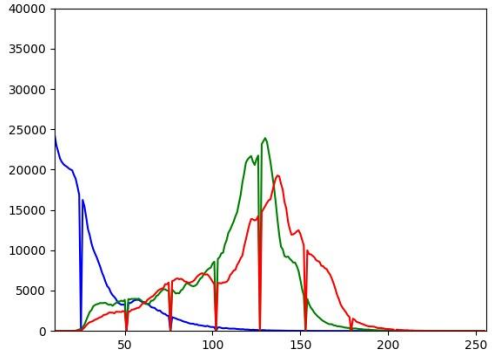
Extração de características




```
EM - 1 (color=r)
mean 45.766601552388124
std 43.28884718885949
mode 73
skewness 1.6352922916412354
kurtosis 1.7245368957519531
```




```
EM - 2 (color=r)
mean 51.08989740673107
std 49.92298583374931
mode 108
skewness 1.374550223350525
kurtosis 0.5523767471313477
```




```
EM - 3 (color=r)
mean 55.93424040732118
std 60.414718826968844
mode 137
skewness 1.200707197189331
kurtosis 0.5930571556091309
```

Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Criação de um modelo

- ➔ Um modelo de reconhecimento de padrões consiste em um conjunto de suposições estatísticas sobre um conjunto de dados

- ➔ Diversas técnicas podem ser utilizadas para a criação de um modelo: medidas estatísticas somadas a um *threshold*, comparação de histogramas (utilizando alguma medida de distância, e.g. distância Euclidiana), técnicas de aprendizado de máquina (regressão linear, árvore de decisão, DL, ...)

Criação de um modelo

➔ Para nosso problema, utilizaremos um valor de *threshold* na métrica de *skewness* selecionado com base no histograma da cor vermelha. Para computar esse valor, poderíamos utilizar o valor que melhor separa as diversas classes do problema considerando todas as imagens disponíveis no momento da criação desse modelo

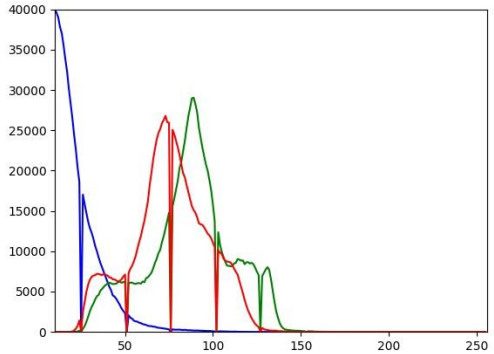
Criação de um modelo

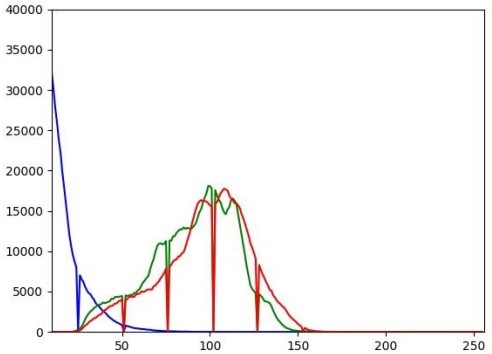
➔ Exemplo de um modelo

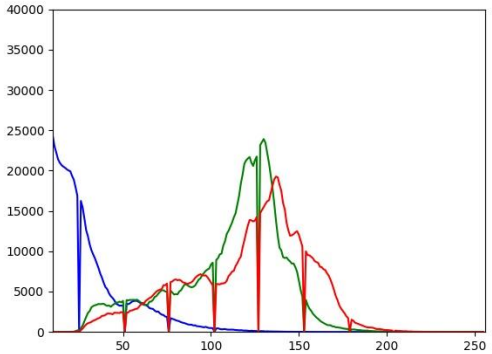
```
from scipy import stats

img = cv2.imread('image.jpg')
color = ('b', 'g', 'r')
histr = cv2.calcHist([img], [2], None, [256], [10, 256])

skew = stats.skew(histr)
if skew < threshold_c1:
    print('Classe 1')
elif skew < threshold_c2:
    print('Classe 2')
else:
    print('Classe 3')
```

Criação de um modelo

➔ Um exemplo considerando uma regressão linear








```
EM - 1 (color=r)
mean 45.766601552388124
std 43.28884718885949
mode 73
skewness 1.6352922916412354
kurtosis 1.7245368957519531
```

```
EM - 2 (color=r)
mean 51.08989740673107
std 49.92298583374931
mode 108
skewness 1.374550223350525
kurtosis 0.5523767471313477
```

```
EM - 3 (color=r)
mean 55.934240407321118
std 60.414718826968844
mode 137
skewness 1.200707197189331
kurtosis 0.59305715560913093
```

Pipeline para reconhecimento de padrões

1. Aquisição de imagens digitais
2. Pré-processamento
3. Extração de características
4. Criação de um modelo
5. Avaliação do modelo

Avaliação do modelo

- ➔ Após a criação de um modelo, é necessário que a performance do mesmo seja aferida

- ➔ Para tal, devemos reservar um conjunto de imagens que não foram utilizadas durante a criação desse modelo e avaliar a performance do mesmo nesse conjunto

- ➔ Após isso, podemos computar métricas como acurácia e uma matriz de confusão

Conclusão

- ➔ Foi apresentado uma visão geral de reconhecimento de padrões em imagens

- ➔ Além disso, é possível perceber que nas diferentes fases do *pipeline* apresentado, fica a critério de um especialista tomar diversas decisões:
  - ◆ Como adquirir as imagens?
  - ◆ Quais métodos de pré-processamento utilizar?
  - ◆ Quais características devem ser extraídas?
  - ◆ Qual o melhor modelo?
  - ◆ Como avaliar esse modelo?

Referências

- Gonzalez, R. C.; Woods, R. E. - Processamento Digital de Imagens , 3a edição, Pearson Prentice Hall, 2010.

- Castleman, K. R. - Digital Image Processing. Prentice Hall 1996.

- Umbaugh, S. E. - Digital image processing and analysis: human and computer vision applications with CVIPtools. [S.l.]: CRC press, 2010

- Krig, S. (2014). Computer Vision Metrics: Survey, Taxonomy, and Analysis.