《计算机视觉-openCV应用技术》

第五章 人脸检测和识别

李策

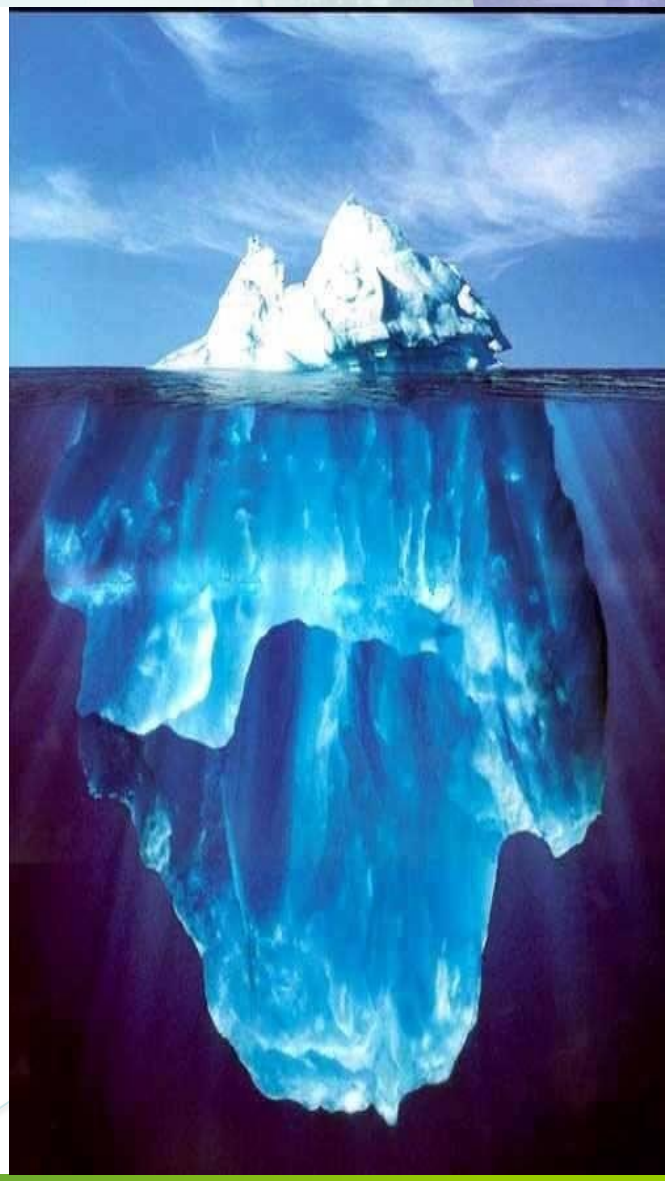
中国矿业大学（北京）计算机科学与技术系

E-mail: celi@cumtb.edu.cn

提纲

第五章 人脸检测和识别

- 5.1 Haar 级联的概念
- 5.2 获取Haar级联数据
- 5.3 使用OpenCV进行人脸检测
  - 5.3.1 静态图像中的人脸检测
  - 5.3.2 视频中的人脸检测
  - 5.3.3 人脸识别
- 5.4 总结



5.1 Haar 级联的概念

从图像数据中提取特征。

虽然任意像素都可能影响多个特征，但特征应该比像素少的多，两个图像的相似程度可以通过它们对应特征的欧式距离来度量。

对于给定的图像，特征可能会因区域大小（也称为窗口大小 window size）而有所不同。仅在尺度上不同的两幅图也应该有相似的特征。因此能为不同大小的窗口生成特征非常有用，这些特征集合称为**级联**。

5.1 Haar 级联的概念

性质：具有尺度不变性，即在尺度变化上具有鲁棒性。OpenCV 提供了尺度不变 Haar 级联的分类器和跟踪器，可将其保存成指定的文件格式。

注意：OpenCV 的 Haar 级联不具有旋转不变性；

如：Haar 级联不认为倒置的人脸图像和直立的人脸图像一样，而侧面的人脸图像和正面的人脸图像也不一样；更可通过多种图像变换和多种窗口大小来提高 Haar 级联的旋转鲁棒性，但这样会变得很复杂，而且会耗费更多计算资源。

5.2 获取Haar级联数据

1.找所有 OpenCV haarcascades 文件

C:\Users***\AppData\Local\Programs\Python\Python37-32\Lib\site-packages\cv2\data\ haarcascades\

这些文件可用于检测静止图像、视频和摄像头所得到的图像中的人脸。

2.创建一个项目文件夹，然后再创建一个子文件（命名为 cascades），将 haarcascades 文件夹中的所有文件复制到 cascades 文件夹中。

5.3 使用OpenCV进行人脸检测

人脸检测的图像来源可分为：

- 静态图像
- 视频

静态图像和视频中检测人脸的操作非常相似，基本理论也是一致的；实际上，视频人脸检测是从摄像头中读取每帧图像，然后采用静态图像中的人脸检测方法进行检测；不过，视频人脸检测还涉及诸如跟踪等静态图像中没有的概念。

5.3.1 静态图像中的人脸检测

创建一个基本的脚本实现人脸检测：

1. 导入所需的cv2模块，定义detect函数：

```
def detect(filename):
```

2. 函数声明一个变量，该变量为级联分类器CascadeClassifier对象，它负责人脸检测

```
face_cascade = cv2.CascadeClassifier('./cascades/haarcascade_frontalface_default.xml')
```

5.3.1 静态图像中的人脸检测

3.加载文件并将其转为灰度图，因为人脸检测需要这样的色彩空间

```
img = cv2.imread(filename)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

4.调节参数可以实现人脸的有效识别

```
faces = face_cascade.detectMultiScale(gray,1.3,5)
```

detectMultiScale函数检测操作返回人脸矩形数组

scaleFactor=1.3 - 人脸检测过程中每次迭代时图像的压缩率

minNeighbors=5 - 人脸检测过程中每次迭代时每个人脸矩形保留近邻数目的最小值

5.3.1 静态图像中的人脸检测

5. cv2.rectangle通过坐标绘制矩形（x和y表示左上角，w和h表示人脸矩形的宽度和高度）

注意：这是在原始图像而不是灰度图上进行绘制

```
for (x,y,w,h) in faces:
    img = cv2.rectangle(img, (x,y),(x+w,y+h),(255,0,0),2)
```

6. 创建nameWindow的实例，并显示处理后的图像。

```
cv2.namedWindow('Detected')
cv2.imshow('Detected',img)
cv2.imwrite('./Detected.jpg', img)
```

5.3.1 静态图像中的人脸检测

7. 加入waitKey函数，这样在按下任意键时才可关闭窗口

```
cv2.waitKey(0)
```

输出图像：

5.3.2 视频中的人脸检测

前面的内容可以实现静态图像上的人脸检测；实际上，在视频的帧上重复这个过程就能完成视频（摄像头的输入或视频文件）中的人脸检测。

1. 导入所需的cv2模块

```
import cv2
```

2. 定义detect函数：

```
face_cascade =
cv2.CascadeClassifier('./cascades/haarcascade_frontalface_default.
xml')
```

5.3.2 视频中的人脸检测

```
    eye_cascade =
cv2.CascadeClassifier('./cascades/haarcascade_eye.xml')
    camera = cv2.VideoCapture(0)
```

3. 打开一个VideoCapture目标（初始化摄像头），参数为摄像头ID，0表获取第一个摄像头

```
    while(True):
        ret, img = camera.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

5.3.2 视频中的人脸检测

4. 捕获帧。

read()函数返回两个值：

第一个值为布尔值，表明是否成功读取帧

第二个值为帧本身

捕获到帧后，将其转换为灰度图像：

```
faces = face_cascade.detectMultiScale(gray,1.3, 5)
```

5. 对具有灰度色彩空间的帧调用detectMultiScale

```
for (x,y,w,h) in faces:
    img = cv2.rectangle(img, (x,y),(x+w,y+h),(255,0,0),2)
    roi_gray=gray[y:y+h,x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray,1.3, 5)
```

5.3.2 视频中的人脸检测

6. 为人脸矩形框创建一个相对应的感兴趣区域，并在该矩形中进行 “眼睛检测”

```
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(img,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)
```

7. 循环输出检测眼睛的结果，在其周围绘制绿色的矩形框

```
cv2.imshow(“camera”, img)
    if cv2.waitKey(1000 / 12) & 0xff == ord(“q”):
        break
camera.release()
cv2.destroyAllWindows()
```

5.3.3 人脸识别

人脸检测是OpenCV的一个不错的功能，它是人脸识别的基础。

人脸识别就是程序能识别给定图像或视频中的人脸；可通过一系列分好类的图像（人脸数据库）来“ 训练 ”程序, 并基于这些图像进行识别。

人脸识别所需的人脸获取方式

- 自己获得图像
- 从人脸数据库免费获得可用的人脸图像

5.3.3 人脸识别

1. 生成人脸识别数据

人脸识别模块的一项重要特征：每个识别都具有转置信（confidence）评分。所以在实际应用中通过对其设置阈值来进行筛选。

样本具有如下特征：

- 图像为灰度格式，后缀名为.pgm
- 图像为正方形
- 图像大小要一样（这里保存为200*200，大多是免费图像集都比这个小）

5.3.3 人脸识别

2. 人脸识别

OpenCV 3 有三种人脸识别的方式，对应于三种不同的算法：

- Eigenfaces - Eigenfaces 是通过 PCA 来处理的，PCA 是计算机视觉中提到最多的数学概念。PCA 的本质是识别某个训练集上（比如人脸数据库）的主成分，并计算出训练集（图像或帧中检测到的人脸）相对于数据库的发散程度，并输出一个值。该值越小，表明人脸数据库和检测到的人脸之间的差别就越小；0 值表示完全匹配。

5.3.3 人脸识别

- Fisherfaces - Fisherfaces 是从PCV 衍生并发展起来的概念，它采用更复杂的逻辑。尽管计算更加密集，但比Eigenfaces 更容易得到准确的效果。
- Local Binary Pattern Histogram（LBPH） - LBPH粗略地（在非常高的层次上）将检测到的人脸分成小单元，并将其与模型中的对应单元进行比较，对每个区域的匹配值产生一个直方图。由于这种方法的灵活性，LBPH 是唯一允许模型样本人脸和检测到的人脸在形状、大小上可以不同的人脸识别算法。

5.3.3 人脸识别

所有的方法都有类似的过程，即都使用了分好类的训练数据集（人脸数据库，每个人都有很多样本）来进行“训练”，对图像或视频中检测到的人脸进行分析，并从两方面来确定：

- 是否识别到目标
- 目标真正被识别到的置信度的度量，这也称为置信度评分

5.3.3 人脸识别

3. 准备训练数据

将已有的样本图像加载到人脸识别算法中。

一般而言，人脸识别算法在它们的 train() 函数中都有两个参数：

- 图像数组
- 标签数组

5.3.3 人脸识别

基于上述要求创建一个 逗号分隔值（comma-separated value，CSV）的文件，并根据ID记录样本图像的路径。CSV文件内容如下所示：

jm/1.pgm; 0

jm/2.pgm; 0

jm/3.pgm; 0

...

jm - 文件夹

1.pgm、2.pgm - 样本图像

0 - 人脸ID，假如A的人脸ID为0，B的人脸ID为1。标签

5.3.3 人脸识别

4. 加载数据并识别人脸

将 **图像数组** 和 **CSV文件** 加载到人脸识别的算法中，以训练人脸识别算法。

- 需要创建函数用来逐行读取 CSV 文件。
- 将对应路径的图像加载到图像数组中。
- 将 ID 加载到标签数组中。

5.3.3 人脸识别

5. 基于Eigenfaces的人脸识别

- 声明数组名称，用IDO来标识
- 加载图像
- 创建人脸识别模型
- 通过图像数组和标签数组来训练模型
- 重复与人脸检测操作类似的过程

5.3.3 人脸识别

6. 基于Fisherfaces的人脸识别

Fisherfaces 的实现过程并没有太大变化，只需要采用不同的算法。模型变量的声明如下：

```
model = cv2.face.createFisherFaceRecognizer()
```

该函数参数与 Eigenfaces 参数相同，保留 Fisherfaces 的参数以及置信度阈值；置信度高于该阈值的人脸将被丢弃。

5.3.3 人脸识别

7. 基于 LBPH 的人脸识别

该运算过程与前面的识别过程很类似

该算法的参数依次表示radius、neighbors、grid_x、grid_y 以及置信度阈值；默认值为：1，8，8，8，123.0。模型声明如下

```
model = cv2.face.createLBPHFaceRecognizer()
```

对于 LBPH 不需要调整图像大小，因为网格中的分割允许在每个单元中比较识别到的模式。

5.3.3 人脸识别

8. 通过置信度评分来丢弃结果

predict() 函数返回含有两个元素的数组：

- 被识别个体的标签
- 置信度评分

所有的算法都有一个置信度评分阈值，置信度评分用来衡量所识别人脸与原模型的差距，0 表示完全匹配。

5.4 总结

现在，对人脸检测和人脸识别是如何工作的以及如何用Python和OpenCV3实现人脸检测和识别有了深入的了解

但是，检测和识别的精度大部分取决于训练数据的质量。因此，为了得到满意的结果，需要确保向应用程序提供高质量的人脸数据库。