人脸识别

OpenCV 简介

OpenCV 的全称是 Open Source Computer Vision Library，是一个跨平台的计算机视觉库。OpenCV 是由英特尔公司发起并参与开发，以 BSD 许可证授权发行，可以在商业和研究领域中免费使用。OpenCV 可用于开发实时的图像处理、计算机视觉以及模式识别程序。该程序库也可以使用英特尔公司的 IPP 进行加速处理。

OpenCV 用 C++语言编写，它的主要接口也是 C++语言，但是依然保留了大量的 C 语言接口。该库也有大量的 Python、Java and MATLAB/OCTAVE（版本 2.5）的接口。这些语言的 API 接口函数可以通过在线文档获得。如今也提供对于 C#、Ch、Ruby、GO 的支持。

安装 OpenCV 模块

OpenCV 已经支持 python 的模块了，直接使用 pip 就可以进行安装，命令如下：

```
pip install opencv-python
```

OpenCV 基本使用

读取图片

显示图像是 OpenCV 最基本的操作之一，imshow()函数可以实现该操作。如果使用过其他 GUI 框架背景，就会很自然第调用 imshow()来显示一幅图像。imshow()函数有两个参数：显示图像的帧名称以及要显示的图像本身。直接调用 imshow()函数图像确实会显示，但随即会消失。要保证图片一直在窗口上显示，要通过 waitKey()函数。waitKey()函数的参数为等待键盘触发的时间，单位为毫秒，其返回值是-1（表示没有键被按下）

```
image = cv2.imread(imagepath)
```

【示例】读取图片

```
import cv2 as cv
img=cv.imread('lena.jpg') #注意读取图片的路径不能有中文，不然数据读取不出来
cv.imshow('input image',img)
cv.waitKey(0) #等待键盘的输入    单位是毫秒    传入 0  无限等待
cv.destroyAllWindows()    #C++语言      使用完内存必须释放
```

图片灰度转换

OpenCV 中有数百种关于在不同色彩空间之间转换的方法。当前，在计算机视觉中有三种常用的色彩空间：灰度、BGR、以及 HSV（Hue，Saturation，Value）。

（1）灰度色彩空间是通过去除彩色信息来将其转换成灰阶，灰度色彩空间对中间处理特别有效，比如人脸识别。

（2）BGR 及蓝、绿、红色彩空间，每一个像素点都由一个三元数组来表示，分别代表蓝、绿、红三种颜色。网页开发者可能熟悉另一个与之相似的颜色空间：RGB 它们只是颜色顺序上不同。

（3）HSV，H（Hue）是色调，S（Saturation）是饱和度，V（Value）表示黑暗的程度（或光谱另一端的明亮程度）。

灰度转换的作用就是：转换成灰度的图片的计算强度得以降低。示例如下：

【示例】将图片灰度

```
import cv2 as cv
src=cv.imread('lena.jpg')
cv.imshow('input image',src)
#cv2 读取图片的通道是 BGR（蓝绿红）
#PIL 读取图片的通道是 RGB
gray_img=cv.cvtColor(src,code=cv.COLOR_BGR2GRAY)
cv.imshow('gray_image',gray_img)
cv.waitKey(0)
cv.destroyAllWindows()
#保存图片
cv.imwrite('gray_lena.jpg',gray_img)
```

修改图片尺寸

【示例】修改图片尺寸

```
import cv2 as cv
img=cv.imread('lena.jpg')
cv.imshow('input image',img)
#修改图片的尺寸
# resize_img=cv.resize(img,dsize=(110,160))
resize_img=cv.resize(img,dsize=(400,360))
print(resize_img.shape)
cv.imshow('resize_img',resize_img)
#如果键盘输入的是 q 时候  退出
while True:
    if ord('q') == cv.waitKey(0):
            break
cv.destroyAllWindows()
```

画图

OpenCV 的强大之处的一个体现就是其可以对图片进行任意编辑，处理。 下面的这个函数最后一个参数指定的就是画笔的大小。

【示例】画图

```
import cv2 as cv
img=cv.imread('lena.jpg')
#画矩形
x,y,w,h=50,50,80,80
cv.rectangle(img,(x,y,x+w,y+h),color=(0,255,0),thickness=2) #color=BGR
cv.circle(img,center=(x+w//2,y+h//2),radius=w//2,color=(0,0,255),thickness=2)
cv.imshow('result image',img)
cv.waitKey(0)
cv.destroyAllWindows()
```

人脸检测

Haar 级联的概念

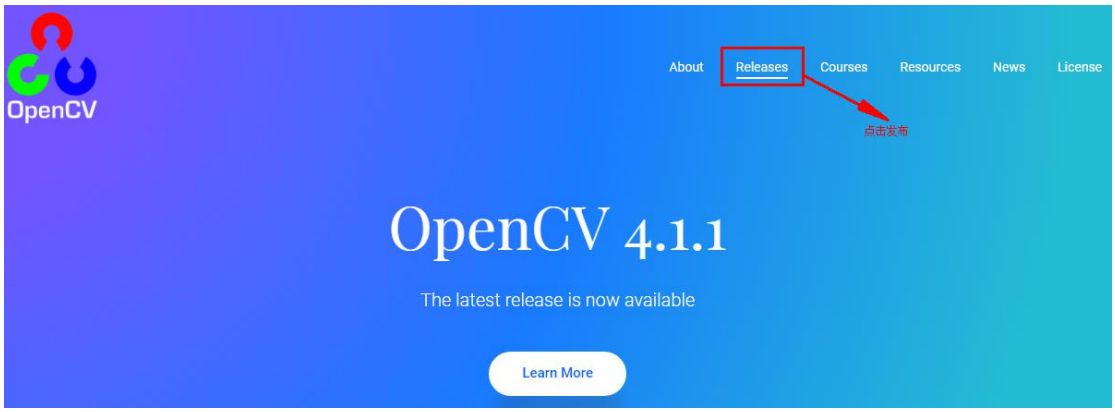
摄影作品可能包含很多令人愉悦的细节。但是，由于灯光、视角、视距、摄像头抖动以及数字噪声的变化，图像细节变得不稳定。人们在分类时不会受这些物理细节方面差异的影响。以前学过，在显微镜下没有两片看起来很像的雪花。幸运的是，作者生长在加拿大，已经学会如何不用显微镜来识别雪花。

因此，提取出图像的细节对产生稳定分类结果和跟踪结果很有用。这些提取的结果被称为特征，专业的表述为：从图像数据中提取特征。虽然任意像素都可以能影响多个特征，但特征应该比像素少得多。两个图像的相似程度可以通过它们对应特征的欧氏距离来度量。

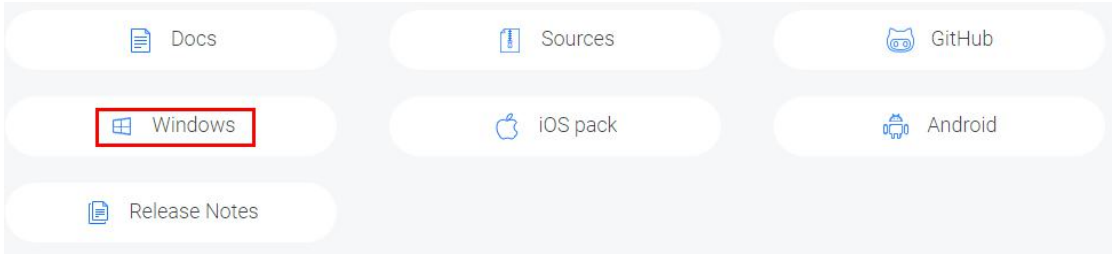
Haar 特征是一种用于实现实时人脸跟踪的特征。每一个 Haar 特征都描述了相邻图像区域的对比模式。例如，边、顶点和细线都能生成具有判别性的特征。

获取 Haar 级联数据

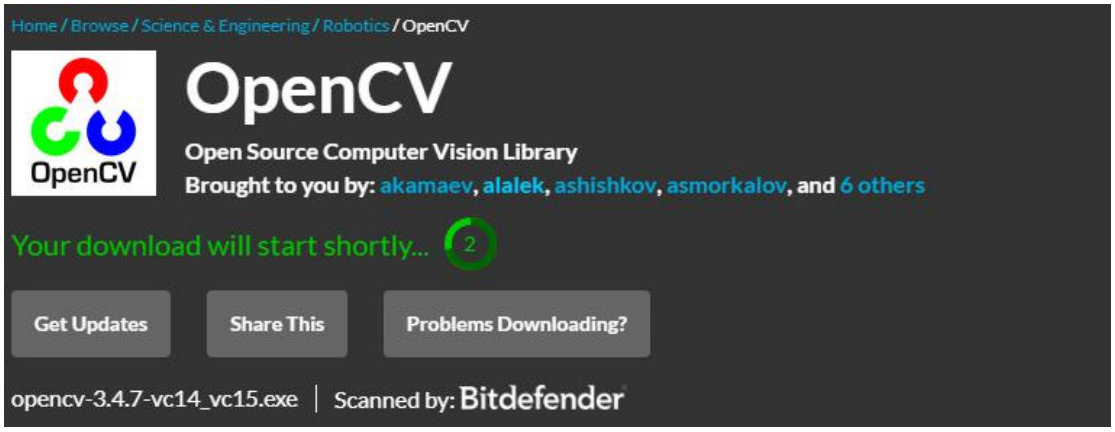
首先我们要进入 OpenCV 官网：https://opencv.org 下载你需要的版本。点击 RELEASES（发布）。如下图所示：



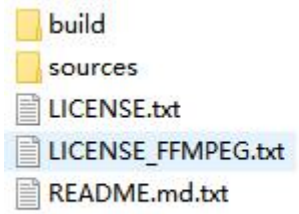
由于 OpenCV 支持好多平台，比如 Windows, Android, Maemo, FreeBSD, OpenBSD, iOS, Linux 和 Mac OS，一般初学者都是用 windows，点击 Windows。



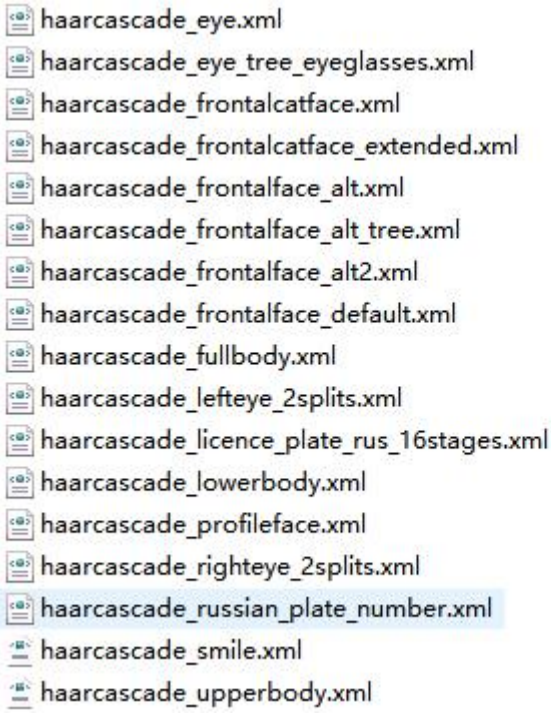
点击 Windows 后跳出下面界面，等待 5s 自动下载。



然后双击下载的文件，进行安装，实质就是解压一下，解压完出来一个文件夹，其他什么也没发生。安装完后的目录结构如下。其中 build 是 OpenCV 使用时要用到的一些库文件，而 sources 中则是 OpenCV 官方为我们提供的一些 demo 示例源码。



在 sources 的一个文件夹 data/haarcascades。该文件夹包含了所有 OpenCV 的人脸检测的 XML 文件，这些可用于检测静止图像、视频和摄像头所得到图像中的人脸。



人脸检测器（默认）：haarcascade_frontalface_default.xml

人脸检测器（快速 Harr）：haarcascade_frontalface_alt2.xml

人脸检测器（侧视）：haarcascade_profileface.xml

眼部检测器（左眼）：haarcascade_lefteye_2splits.xml

眼部检测器（右眼）：haarcascade_righteye_2splits.xml

嘴部检测器：haarcascade_mcs_mouth.xml

鼻子检测器：haarcascade_mcs_nose.xml

身体检测器：haarcascade_fullbody.xml

人脸检测器（快速 LBP）：lbpcascade_frontalface.xml

使用 OpenCV 进行人脸检测

静态图像中人脸检测

人脸检测首先是加载图像并检测人脸，这也是最基本的一步。为了使所得到的结果有意义，可在原始图像的人脸周围绘制矩形框。

【示例】识别图片中的人脸

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

```
def face_detect_demo():
    gray=cv.cvtColor(src,cv.COLOR_BGR2GRAY)

face_detector=cv.CascadeClassifier('E:\\soft\\opencv\\opencv\\sources\\data\\haarcascades\\haarcascade_frontalface_alt_tree.xml')
    faces=face_detector.detectMultiScale(gray,1.02,5)
    for x,y,w,h in faces:
        cv.rectangle(src,(x,y),(x+w,y+h),color=(0,0,255))
    cv.imshow('result',src)

src = cv.imread('lena.jpg')
cv.imshow('result',src)
face_detect_demo()
cv.waitKey(0)
cv.destroyAllWindows()
```

【示例】识别图片中多张人脸

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
def face_detect_demo():
    gray=cv.cvtColor(src,cv.COLOR_BGR2GRAY)

face_detector=cv.CascadeClassifier('E:\\soft\\opencv\\opencv\\sources\\data\\haarcascades\\haarcascade_frontalface_alt_tree.xml')
    faces = face_detector.detectMultiScale(src)
    #修改检测参数 scaleFactor minNeighbors
    faces=face_detector.detectMultiScale(src,scaleFactor=1.01,minNeighbors=3)
    for x,y,w,h in faces:
        cv.rectangle(src,(x,y),(x+w,y+h),color=(0,0,255),thickness=2)
        cv.circle(src,center=(x+w//2,y+h//2),radius=w//2,color=(0,255,0),thickness=2)
    cv.imshow('result',src)

src = cv.imread('face2.jpg')
```

```
cv.imshow('result',src)
face_detect_demo()
cv.waitKey(0)
cv.destroyAllWindows()
```

视频中的人脸检测

视频是一张一张图片组成的，在视频的帧上重复这个过程就能完成视频中的人脸检测。

【示例】识别视频中人脸

```
import cv2 as cv
def face_detect_demo(img):
    #将图片灰度
    gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    #加载特征数据
    face_detector = cv.CascadeClassifier(
        'E:/soft/opencv/opencv/sources/data/haarcascades/haarcascade_frontalface_default.xml')
    faces = face_detector.detectMultiScale(gray)
    for x,y,w,h in faces:
        cv.rectangle(img,(x,y),(x+w,y+h),color=(0,0,255),thickness=2)
        cv.circle(img,center=(x+w//2,y+h//2),radius=(w//2),color=(0,255,0),thickness=2)
    cv.imshow('result',img)
#读取视频
cap=cv.VideoCapture('video.mp4')
while True:
    flag,frame=cap.read()
    print('flag:',flag,'frame.shape:',frame.shape)
    if not flag:
        break
    face_detect_demo(frame)
    if ord('q') == cv.waitKey(10):
        break
cv.destroyAllWindows()
cap.release()
```

人脸识别

人脸检测是 OpenCV 的一个很不错的功能，它是人脸识别的基础。什么是人脸识别呢？

其实就是一个程序能识别给定图像或视频中的人脸。实现这一目标的方法之一是用一系列分好类的图像来“训练”程序，并基于这些图像来进行识别。

这就是 OpenCV 及其人脸识别模块进行人脸识别的过程。

人脸识别模块的另外一个重要特征是：每个识别都具有转置信（confidence）评分，因此可在实际应用中通过对其设置阈值来进行筛选。

人脸识别所需要的人脸可以通过两种方式来得到：自己获得图像或从人脸数据库免费获得可用的人脸图像。互联网上有许多人脸数据库：

https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

为了对这些样本进行人脸识别，必须要在包含人脸的样本图像上进行人脸识别。这是一个学习的过程，但并不像自己提供的图像那样令人满意。

训练数据

有了数据，需要将这些样本图像加载到人脸识别算法中。所有的人脸识别算法在它们的 train()函数中都有两个参数：图像数组和标签数组。这些标签表示进行识别时候某人人脸的 ID，因此根据 ID 可以知道被识别的人是谁。要做到这一点，将在「trainer/」目录中保存为.yml 文件。

在使用 Python 3 &OpenCV 3.0.0 进行人脸识别训练时发现异常：

AttributeError: ‘module’ object has no attribute ‘LBPHFaceRecognizer_create’OpenCV 需要安装 opencv-contrib-python 模块，直接使用 pip 就可以进行安装，命令如下：

```
pip install opencv-contrib-python
```

【示例】训练数据

```
import os
import cv2
import numpy as np
import sys
from PIL import Image
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')
```

```
            id = int(os.path.split(imagePath)[-1].split(".")[0])
            faces = detector.detectMultiScale(img_numpy)
            for (x,y,w,h) in faces:
                faceSamples.append(img_numpy[y:y+h,x:x+w])
                ids.append(id)
    return faceSamples,ids

if __name__ == '__main__':
    path='./data/jm3/'
    faces, ids = getImagesAndLabels(path)
    recognizer.train(faces, np.array(ids))
    # Save the model into trainer/trainer.yml
    recognizer.write('trainer/trainer.yml')
```

基于 LBPH 的人脸识别

LBPH（Local Binary Pattern Histogram）将检测到的人脸分为小单元，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。

调整后的区域中调用 predict()函数，该函数返回两个元素的数组：第一个元素是所识别个体的标签，第二个是置信度评分。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。可能有时不想保留所有的识别结果，则需要进一步处理，因此可用自己的算法来估算识别的置信度评分。LBPH 一个好的识别参考值要低于 50 ，任何高于 80 的参考值都会被认为是低的置信度评分。

【示例】基于 LBPH 的人脸识别

```
import cv2
import numpy as np
import os
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
img=cv2.imread('9.pgm') #识别的图片
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(gray,scaleFactor = 1.2,minNeighbors = 5)
```

```
for(x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
    print(id,confidence)
cv2.imshow('camera',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```