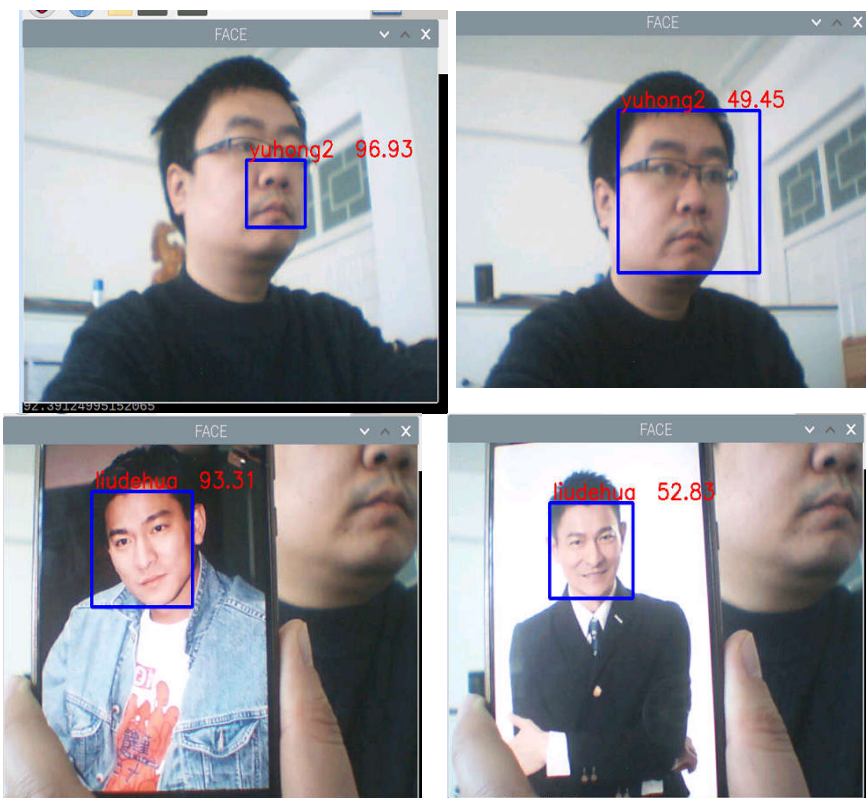


# 智能系统与控制

## 树莓派：OpenCV 人脸识别



于泓

鲁东大学

信息与电气工程学院

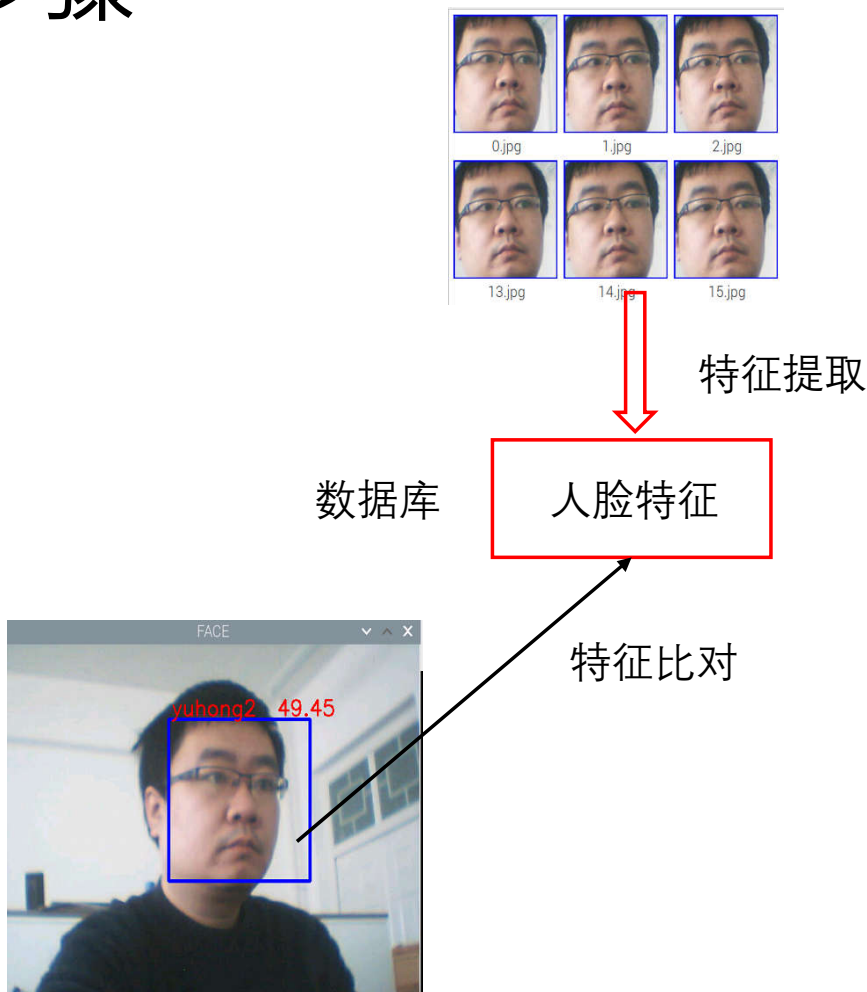
2022.2.15

# 人脸识别

人脸识别指识别并理解一张脸。人脸识别是一项热门的计算机技术研究领域，它属于生物特征识别技术，是对生物体（一般特指人）本身的生物特征来区分生物体个体。人脸识别技术在安防监控、身份认证等众多领域都有着重要的作用。

# 人脸识别的一般步骤

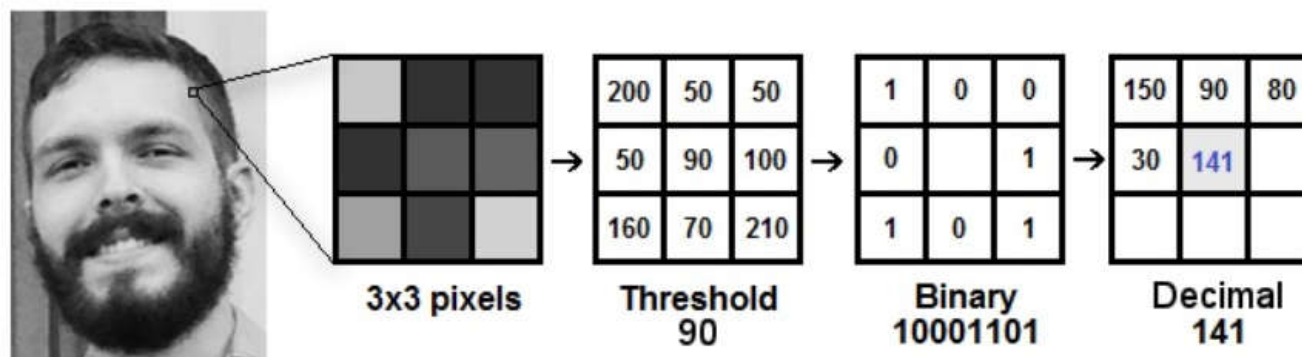
- (1) 人脸注册 (收集)
- (2) 对注册人脸进行特征提取  
存入人脸特征库
- (3) 人脸检测 (识别过程)
- (4) 对检测人脸提取特征  
然后与数据库内人脸  
进行比对
- (5) 根据阈值做出判断



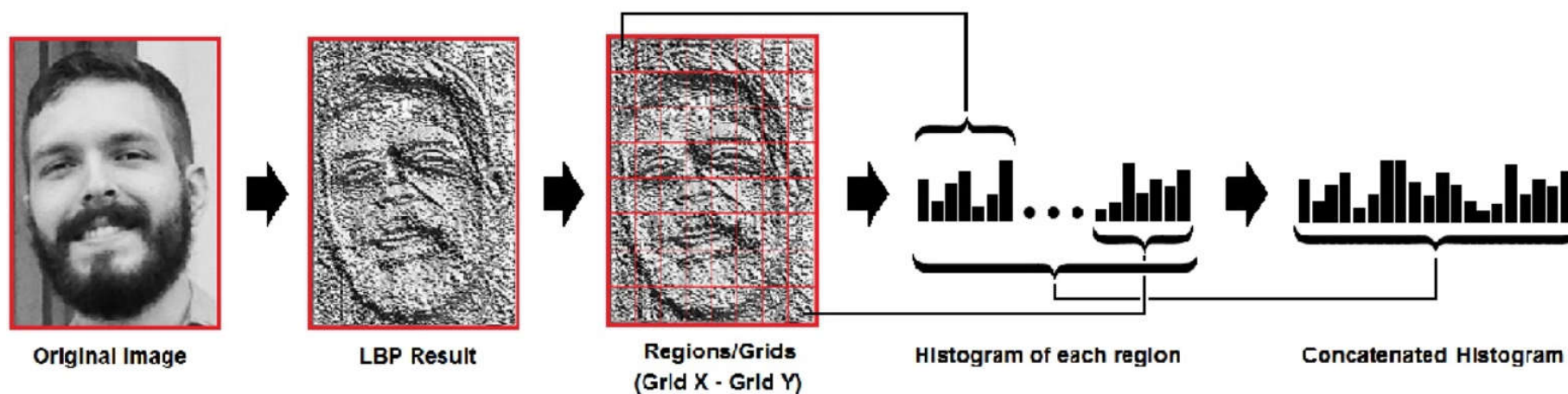
## 人脸特征：LBP

LBP 是一种在纹理识别以及人脸识别中经常使用的特征，其提取过程如图所示。

首先将输入图像灰度化之后，对于每一个像素点，在  $3 \times 3$  的窗口内，比较其与周围 8 个像素点的大小。根据大小比较情况对周围的 8 个像素点进行二值化编码，比中心点小的编码为 0，比中心点大的编码为 1。这样中心点的像素就可以用一个 8 位 2 进制数来进行描述。



对图像中的每一个点都进行上述操作后，就可以将一张灰度图重新编码为一张 LBP 特征图。



将这个特征图分成小块（通常是  $8 \times 8$  的小块），计算每个小块中特征点的统计直方图，然后将所有小块的直方图拼接起来，就可以形成一个能够描述输入人脸图像的特征向量。

特征维度：  $8 \times 8 \times 256 = 16384$

## 代码实现 (1) 人脸收集

```
import cv2
import os

if __name__ == "__main__":
    str_face_id = ""
    index_photo=0

    # 加载训练好的人脸检测器
    faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

    # 打开摄像头
    cap = cv2.VideoCapture(0)

    while True:

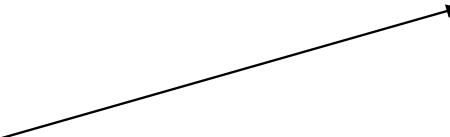
        # 判断人脸id 是否为空，空的话创建face_id
        if str_face_id.strip()=="":
            str_face_id = input('Enter your face ID:')
            index_photo=0

            if not os.path.exists(str_face_id):
                os.makedirs(str_face_id)

        # 读取一帧图像
        success, img = cap.read()

        if not success:
            continue
```

首次运行  
输入人脸id



```
# 转换为灰度
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 进行人脸检测
faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(50, 50), flags=cv2.CASCADE_SCALE_IMAGE)

# 画框
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3)

# 显示检测结果
cv2.imshow("FACE", img)

# 读取按键键值
key = cv2.waitKey(1) & 0xFF

# 按键"c" 进行人脸采集
if key == ord('c'):

    # 保存人脸
    for (x, y, w, h) in faces:
        roi = img[y:y+h, x:x+w]
        cv2.imwrite("%s/%d.jpg"%(str_face_id, index_photo), roi)
        index_photo = index_photo+1
    key = 0
# 按键"x" 切换 人脸_id
elif key == ord('x'):
    str_face_id = ""
    key = 0
# 按键 "q" 退出
elif key == ord('q'):
```

人脸检测

按“c”键实现人脸收集保存

按x 进行人脸切换，输入人脸id 进行新一轮人脸收集



## (2) 构建人脸特征库

该模块  
属于opencv-contrib

**pip3 install opencv-contrib-python**

```
import cv2
import os
import numpy as np

# 获取所有文件 (人脸id)
def get_face_list(path):
    for root,dirs,files in os.walk(path):
        if root == path:
            return dirs

if __name__ == "__main__":

    # 创建人脸识别器
    recognizer = cv2.face.LBPHFaceRecognizer_create()

    # 用来存放人脸id的字典
    # 构建人脸编号 和 人脸id 的关系
    dic_face = {}

    # 人脸存储路径
    base_path = "../face_collect/"

    # 获取人脸id
    face_ids = get_face_list(base_path)
    print(face_ids)
    # 用来存放人脸数据与id号的列表
    faceSamples=[]
    ids = []
```

创建人脸识别器

遍历存放人脸的文件夹



```
# 遍历人脸id命名的文件夹
for i, face_id in enumerate(face_ids):

    # 人脸字典更新
    dic_face[i] = face_id

    # 获取人脸图片存放路径
    path_img_face = os.path.join(base_path, face_id)

    for face_img in os.listdir(path_img_face):
        # 读取以.jpg为后缀的文件
        if face_img.endswith(".jpg"):
            file_face_img = os.path.join(path_img_face, face_img)

            # 读取图像并转换为灰度图
            img = cv2.imread(file_face_img)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            # 保存图像和人脸ID
            faceSamples.append(img)
            ids.append(i)

print(dic_face)

# 进行模型训练
recognizer.train(faceSamples, np.array(ids))

# 模型保存
recognizer.save('trainer.yml')

# 进行字典保存
with open("face_list.txt", 'w') as f:
    for face_id in dic_face:
        f.write("%d %s\n"%(face_id, dic_face[face_id]))
```

[0,0,0,0,...1,1,1,1,]

/home/pi/source\_opencv/lesson\_face\_recognize/OpenCv\_re

Name	Size (KB)	Li
..		
trainer.yml	3 372	2022-
face_list.txt	1	2022-
train_model_LBP.py	1	2022-
face_recognize_LBP.py	2	2022-

OCR.py test\_OCR\_2.py

1	0	liudehua
2	1	yuhong
3		

存放LBP特征

```
%YAML: 1.0
---
opencv_lbpfaces:
  threshold: 1.7976931348623157e+308
  radius: 1
  neighbors: 8
  grid_x: 8
  grid_y: 8
  histograms:
    - !!opencv-matrix
      rows: 1
      cols: 16384
      dt: i
      data: [ 6.85871067e-03, 6.85871067e-03, 0., 0., 1.37174211e-03,
```

```
labels: !!opencv-matrix
  rows: 24
  cols: 1
  dt: i
  data: [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1 ]
labelsInfo:
  []
```

## 人脸识别

```
import cv2
import os
import numpy as np

def read_dic_face(file_list):
    data = np.loadtxt(file_list, dtype='str')
    dic_face = {}
    for i in range(len(data)):
        dic_face[int(data[i][0])] = data[i][1]

    return dic_face

if __name__ == "__main__":
    # 加载人脸字典
    dic_face = read_dic_face("face_list.txt")
    print(dic_face)

    # 加载Opencv人脸检测器
    faceCascade = cv2.CascadeClassifier('../face_collect/haarcascade_frontalface_alt.xml')

    # 加载训练好的人脸识别器
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read('trainer.yml')

    # 打开摄像头
    cap = cv2.VideoCapture(0)
```

加载字典



1	0	liudehua
2	1	yuhong
3		

```
while True:
```

```
    # 读取一帧图像
```

```
    success, img = cap.read()
```

```
    if not success:
```

```
        continue
```

```
    # 转换为灰度
```

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

```
    # 进行人脸检测
```

```
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(50, 50), flags=cv2.CASCADE_SCALE_IMAGE)
```

```
    # 遍历检测到的人脸
```

```
    for (x, y, w, h) in faces:
```

```
        # 画框
```

```
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3)
```

```
        # 进行人脸识别
```

```
        id_face, confidence = recognizer.predict(gray[y:y+h, x:x+w])
```

```
        print(confidence)
```

```
        # 检测可信度，这里是通过计算距离来计算可信度，confidence越小说明越近似
```

```
        if (confidence < 100):
```

```
            str_face = dic_face[id_face]
```

```
            str_confidence = " %.2f"%(confidence)
```

```
        else:
```

```
            str_face = "unknown"
```

```
            str_confidence = " %.2f"%(confidence)
```

判定阈值（距离），  
越小说明越相似

根据人脸id 获取人脸名称

显示识别的结果

# 检测结果文字输出

`cv2.putText(img, str_face+str_confidence, (x+5,y-5), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)`

# 显示检测结果

`cv2.imshow("FACE",img)`

# 按键 "q" 退出

`if cv2.waitKey(1) & 0xFF == ord('q'):`  
`break`

`cap.release()`

# 基于Dlib的人脸识别

Dlib中 用于人脸特征提取预训练模型为“dlib\_face\_recognition\_resnet\_model\_v1.dat”，可以在 Dlib 的网站上进行下载<sup>12</sup>。该模型是一个包含了 29 个卷积层的 ResNet 网络，其具体结构参照了经典的 ResNet-34 网络<sup>13</sup>并进行了一些缩减，减少了层数，并将每一层的滤波器数目减半。整个网路训练用了 7485 个人的大约 300 万张人脸。这些人脸采集自 face scrub 数据库<sup>14</sup>、VGG 数据库<sup>15</sup> 以及从网上采集的数据，经过了精细数据清理去除了错误标签以及容易判断错误的的数据。网络训练完成后，中间层的 128 维输出被用作人脸特征。

<https://github.com/davisking/dlib-models>

```
import dlib
import os
import numpy as np
import cv2

# 获取所有文件（人脸id）
def get_face_list(path):
    for root,dirs,files in os.walk(path):
        if root == path:
            return dirs

if __name__ == "__main__":
    # 加载人脸特征提取器
    facerec = dlib.face_recognition_model_v1("dlib_face_recognition_resnet_model_v1.dat")

    # 加载人脸标志点检测器
    sp = dlib.shape_predictor("shape_predictor_5_face_landmarks.dat")

    # 人脸识别训练图片存放路径
    base_path = "../face_collect/"

    # 获取人脸 id 列表
    face_list = get_face_list(base_path)

    # 用来存储人脸特征和人脸id的列表
    list_face_vector = []
    list_face_id = []
```

人脸定位辅助



```
for face_id in face_list:

    for f_img in os.listdir(os.path.join(base_path, face_id)):
        if f_img.endswith(".jpg"):
            file_img = os.path.join(base_path, face_id, f_img)

            print("Extract face vector for file %s"%(file_img))
            # 读取图像并转换为RGB
            img = cv2.imread(file_img)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # 在整图内检测标志点
            img = np.array(img)
            h,w,_ = np.shape(img)
            rect = dlib.rectangle(0,0,w,h)
            shape = sp(img, rect)

            # 获取128维人脸特征
            face_vector = facerec.compute_face_descriptor(img, shape)

            # 特征 和 id 保存
            list_face_vector.append(face_vector)
            list_face_id.append(face_id)

# 将最终结果进行保存
face_vectors = np.array(list_face_vector)
ids = np.array(list_face_id)

# 模型保存
np.savez('trainer', face_vectors=face_vectors, ids=ids)
```

输入RGB

辅助进行人脸剪裁

```
import cv2
import os
import dlib
import numpy as np

def face_recognize(face_vec, face_dataset, ids):
    N = face_dataset.shape[0]
    diffMat = np.tile(face_vec, (N, 1)) - face_dataset

    # 计算欧式距离
    distances = np.linalg.norm(diffMat, axis=1)

    # 找到最小距离
    idx = np.argmin(distances)

    # 返回id编号与距离
    return ids[idx], distances[idx]

def face_recognize_cos(face_vec, face_dataset, ids):
    N = face_dataset.shape[0]
    a = np.tile(face_vec, (N, 1))
    b = face_dataset

    # 计算cos 距离
    dis_cos = np.sum(a*b, axis=1) / (np.linalg.norm(a, axis=1) * np.linalg.norm(b, axis=1))

    # 找到最大距离
    idx = np.argmax(dis_cos)

    # 返回id编号与距离
    return ids[idx], dis_cos[idx]
```

## 识别部分

```
if __name__ == "__main__":  
    # 加载训练好的人脸模型  
    model = np.load('trainer.npz')  
  
    face_vectors = model['face_vectors']  
    face_ids = model['ids']  
  
    print(face_vectors)  
    print(face_ids)  
  
    # Dlib 人脸检测器  
    detector = dlib.get_frontal_face_detector()  
  
    # Dlib 标志点检测器  
    sp = dlib.shape_predictor("shape_predictor_5_face_landmarks.dat")  
  
    # Dlib 人脸特征提取器  
    facerec = dlib.face_recognition_model_v1("dlib_face_recognition_resnet_model_v1.dat")  
  
    # 打开摄像头  
    cap = cv2.VideoCapture(0)  
  
    while True:  
        # 读取一帧图像  
        success, img = cap.read()  
  
        if not success:  
            continue
```

```
# BGR 转 gray
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# BGR 转 RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# 进行人脸检测
dets = detector(img_gray, 1)

# 遍历检测的人脸
for k, d in enumerate(dets):

    # 画框
    cv2.rectangle(img, (d.left(), d.top()), (d.right(), d.bottom()), (255, 0, 0), 3)

    # 标志点检测
    shape = sp(img_rgb, d)

    # 获取人脸特征
    face_vector = facerec.compute_face_descriptor(img_rgb, shape)

    # # 进行识别返回ID与距离
    # face_id, dis = face_recognize(np.array(face_vector), face_vectors, face_ids)

    # if (dis < 0.45):
    #     str_out = "%s %.2f"%(face_id, dis)
    # else:
    #     str_out = "unknown %.2f"%(dis)
```

利用欧式距离  
识别，距离越小  
越相似

```
# 进行识别返回ID与距离
face_id, dis = face_recognize_cos(np.array(face_vector), face_vectors, face_ids)

if (dis > 0.95):
    str_out = "%s %.2f"%(face_id, dis)
else:
    str_out = "unknown %.2f"%(dis)

# 检测结果文字输出
cv2.putText(img, str_out, (d.left()+5, d.top()), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

# 显示检测结果
cv2.imshow("FACE", img)

# 按键 "q" 退出
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
```

距离越**大**  
越**相似**