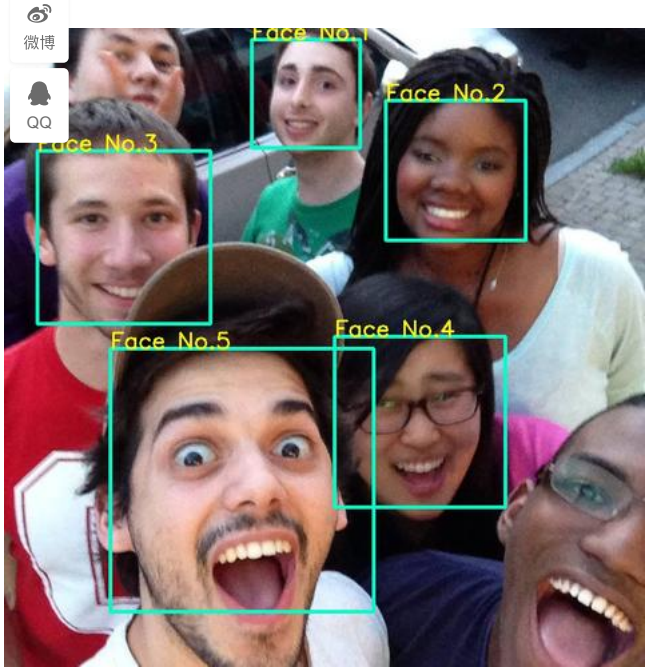
使用树莓派进行简易人脸识别

发 2016年08月10日 07:04:00

使用树莓派2和OpenCV制作一个简易的人脸识别和追踪系统。



所需硬件

需要：

- **树莓派2**
- **Pi Camera**

非必须（如果需要追踪人脸运动，需要一个有两个马达的小云台）：

- **云台**

安装OpenCV

```
1 sudo apt-get update
2 sudo apt-get upgrade
3 sudo apt-get install python-opencv
```

安装PiCamera

由于我没有使用USB摄像头，而是用了特殊的Pi Camera，样子如下图， 所以需要安装PiCamera来控制摄像头。

安装PiCamera:

```
sudo apt-get install python-pip
sudo apt-get install python-dev
sudo pip install picamera
```

至此人脸识别所需要的准备工作已经完成，可以使用下面的演示代码进行测试。

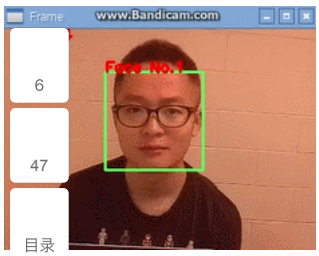
示例代码

Demo.1

第一个演示只使用单核，由于树莓派的性能有限，在只使用一个CPU核心的情况下视频的帧数非常之低，只有5帧左右，效果不太理想， 另外代码中 控制云台的电机，来实现追踪人脸的功能，不过考虑到这个功能不是必须，所以不在此进行介绍。

```
### Imports ###########################################################

from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import os

```

```
# Center coordinates
cx = 160
cy = 120

os.system( "echo 0=150 > /dev/servoblaster" )
os.system( "echo 1=150 > /dev/servoblaster" )

xdeg = 150
ydeg = 150

# Setup the camera
camera = PiCamera()
camera.resolution = ( 320, 240 )
camera.framerate = 60
rawCapture = PiRGBArray( camera, size=( 320, 240 ) )

# Load a cascade file for detecting faces
face_cascade = cv2.CascadeClassifier( '/home/pi/opencv-2.4.9/data/lbpcascades/lbpcascade_frontalface.xml' )

t_start = time.time()
fps = 0


### Main #########################################################################

# Capture frames from the camera
for frame in camera.capture_continuous( rawCapture, format="bgr", use_video_port=True ):

    image = frame.array

    # Use the cascade file we loaded to detect faces
    gray = cv2.cvtColor( image, cv2.COLOR_BGR2GRAY )
    faces = face_cascade.detectMultiScale( gray )

    print "Found " + str( len( faces ) ) + " face(s)"

    # Draw a rectangle around every face and move the motor towards the face
    for ( x, y, w, h ) in faces:

        cv2.rectangle( image, ( x, y ), ( x + w, y + h ), ( 100, 255, 100 ), 2 )
        cv2.putText( image, "Face No." + str( len( faces ) ), ( x, y ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 )

        tx = x + w/2
        ty = y + h/2

        if   ( cx - tx >  10 and xdeg <= 190 ):
            xdeg += 3
            os.system( "echo 0=" + str( xdeg ) + " > /dev/servoblaster" )
        elif ( cx - tx < -10 and xdeg >= 110 ):
            xdeg -= 3
            os.system( "echo 0=" + str( xdeg ) + " > /dev/servoblaster" )

        if   ( cy - ty >  10 and ydeg >= 110 ):
            ydeg -= 3
            os.system( "echo 1=" + str( ydeg ) + " > /dev/servoblaster" )
        elif ( cy - ty < -10 and ydeg <= 190 ):
            ydeg += 3
            os.system( "echo 1=" + str( ydeg ) + " > /dev/servoblaster" )

    # Calculate and show the FPS
    fps = fps + 1
    sfps = fps / ( time.time() - t_start )
    cv2.putText( image, "FPS : " + str( int( sfps ) ), ( 10, 10 ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 ), 2 )

    # Show the frame
    cv2.imshow( "Frame", image )
    cv2.waitKey( 1 )

    # Clear the stream in preparation for the next frame
```



注意由于我使用HaarCascade来进行人脸检测， 需要使用到识别人脸的XML，这些人脸识别的XML文件是随着OpenCV一起安装的，不需要自己树莓派上运行时，请注意调整XML文件的路径， 就是调整这一行：

```
# Load a cascade file for detecting faces
face_cascade = cv2.CascadeClassifier( '你的XML文件路径' )
```

Demo2

时使用不同的XML文件，可以实现同时识别不同物体的功能，比如下面这段代码可以同时识别人脸和黑色手机，识别手机所需要的XML文件是和Thiago Hersan制作的， 来源在这里。 更进一步的，我们可以根据自己的需要训练自己的Cascade文件，Naotoshi Seo在此处 给出了详细的教程Thorsten Ball的香蕉识别教程。

```
### Imports ###################################################################

from picamera.array import PiRGBArray
from picamera import PiCamera

import time
import cv2
import os
import pygame


### Setup #####################################################################

os.putenv('SDL_FBDEV', '/dev/fb1')

# Setup the camera
camera = PiCamera()
camera.resolution = ( 320, 240 )
camera.framerate = 40
rawCapture = PiRGBArray( camera, size=( 320, 240 ) )

# Load the cascade files for detecting faces and phones
face_cascade = cv2.CascadeClassifier( '/home/pi/opencv-2.4.9/data/lbpcascades/lbpcascade_frontalface.xml' )
phone_cascade = cv2.CascadeClassifier( 'cascade.xml' )

t_start = time.time()
fps = 0


### Main ######################################################################

# Capture frames from the camera
for frame in camera.capture_continuous( rawCapture, format="bgr", use_video_port=True ):

    image = frame.array

    # Look for faces and phones in the image using the loaded cascade file
    gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    faces  = face_cascade.detectMultiScale(gray)
    phones = phone_cascade.detectMultiScale(gray)

    # Draw a rectangle around every face
    for (x,y,w,h) in faces:
        cv2.rectangle( image, ( x, y ), ( x + w, y + h ), ( 255, 255, 0 ), 2 )
```

```
    for (x,y,w,h) in phones:
        cv2.rectangle( image, ( x, y ), ( x + w, y + h ), ( 255, 0, 0 ), 2 )
        cv2.putText( image, "iPhone", ( x, y ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 255, 255 ), 2 )

    # Calculate and show the FPS
    fps = fps + 1
    sfps = fps / ( time.time() - t_start )
    cv2.putText( image, "FPS : " + str( int( sfps ) ), ( 10, 10 ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 ), 2 )

    cv2.imshow( "Frame", image )
    cv2.waitKey( 1 )

    # Clear the stream in preparation for the next frame
    rawCapture.truncate( 0 )
```

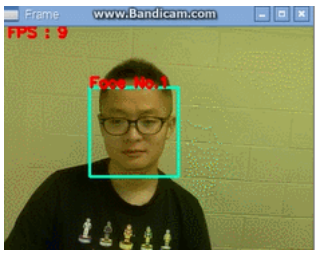

由于使用了更多的XML文件进行识别，帧数降低到了2~3帧。

Demo.3

为了解决帧数较低的问题，有一个比较简单的方法就是跳帧，可以不对每一帧图像都进行识别，而是隔几帧识别一次（因为最初因为懒不想将程序为了提高帧数，所以有了这个蛋疼的方法…）。

```
### Imports ###################################################################

from picamera.array import PiRGBArray
from picamera import PiCamera

import time
import cv2
import os
import pygame


### Setup #####################################################################

os.putenv( 'SDL_FBDEV', '/dev/fb1' )

# Setup the camera
camera = PiCamera()
camera.resolution = ( 320, 240 )
camera.framerate = 30
rawCapture = PiRGBArray( camera, size=( 320, 240 ) )

fcounter = 0
facefind = 0

# Load a cascade file for detecting faces
face_cascade = cv2.CascadeClassifier( '/home/pi/opencv-2.4.9/data/lbpcascades/lbpcascade_frontalface.xml' )

t_start = time.time()
fps = 0

### Main ######################################################################

# Capture frames from the camera
for frame in camera.capture_continuous( rawCapture, format="bgr", use_video_port=True ):

```

```
    if fcounter == 3:

        fcounter = 0

        # Look for faces in the image using the loaded cascade file
        gray = cv2.cvtColor( image, cv2.COLOR_BGR2GRAY )
        faces = face_cascade.detectMultiScale( gray )

        print "Found " + str( len( faces ) ) + " face(s)"

        if str( len( faces ) ) != 0:
            facefind = 1
            facess = faces
        else:
            facefind = 0

        # Draw a rectangle around every face
        for ( x, y, w, h ) in faces:
            cv2.rectangle( image, ( x, y ), ( x + w, y + h ), ( 200, 255, 0 ), 2 )
            cv2.putText( image, "Face No." + str( len( facess ) ), ( x, y ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0,
            facess = faces

    else:
        if facefind == 1 and str( len( facess ) ) != 0:

            # Continue to draw the rectangle around every face
            for ( x, y, w, h ) in facess:
                cv2.rectangle( image, ( x, y ), ( x + w, y + h ), ( 200, 255, 0 ), 2 )
                cv2.putText( image, "Face No." + str( len( facess ) ), ( x, y ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0,

    fcounter += 1


    # Calculate and show the FPS
    fps = fps + 1
    sfps = fps / ( time.time() - t_start )
    cv2.putText( image, "FPS : " + str( int( sfps ) ), ( 10, 10 ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 ), 2 )

    cv2.imshow( "Frame", image )
    cv2.waitKey( 1 )

    # Clear the stream in preparation for the next frame
    rawCapture.truncate( 0 )
```



这样子帧数会提高到10帧左右，已经不像原来那么卡顿，但是当你移动速度很快的时候，识别框会出现滞后。

Demo.4

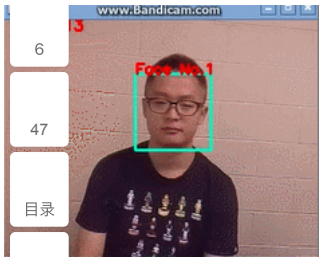
毕竟跳帧只是权宜之计，这个版本使用了全部的CPU核心，帧数稳定在了15帧左右。

```
### Imports ##################################################################

from picamera.array import PiRGBArray
from picamera import PiCamera
from functools import partial

import multiprocessing as mp
```

```
### Setup ################################################################

os.putenv( 'SDL_FBDEV', '/dev/fb0' )

resX = 320
resY = 240

cx = resX / 2
cy = resY / 2

os.system( "echo 0=150 > /dev/servoblaster" )
os.system( "echo 1=150 > /dev/servoblaster" )

xdeg = 150
ydeg = 150


# Setup the camera
camera = PiCamera()
camera.resolution = ( resX, resY )
camera.framerate = 60

# Use this as our output
rawCapture = PiRGBArray( camera, size=( resX, resY ) )

# The face cascade file to be used
face_cascade = cv2.CascadeClassifier('/home/pi/opencv-2.4.9/data/lbpcascades/lbpcascade_frontalface.xml')

t_start = time.time()
fps = 0


### Helper Functions #########################################################

def get_faces( img ):

    gray = cv2.cvtColor( img, cv2.COLOR_BGR2GRAY )
    faces = face_cascade.detectMultiScale( gray )

    return faces, img

def draw_frame( img, faces ):

    global xdeg
    global ydeg
    global fps
    global time_t

    # Draw a rectangle around every face
    for ( x, y, w, h ) in faces:

        cv2.rectangle( img, ( x, y ),( x + w, y + h ), ( 200, 255, 0 ), 2 )
        cv2.putText(img, "Face No." + str( len( faces ) ), ( x, y ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 ), 2

        tx = x + w/2
        ty = y + h/2

        if   ( cx - tx > 15 and xdeg <= 190 ):
            xdeg += 1
            os.system( "echo 0=" + str( xdeg ) + " > /dev/servoblaster" )
        elif ( cx - tx < -15 and xdeg >= 110 ):
            xdeg -= 1
            os.system( "echo 0=" + str( xdeg ) + " > /dev/servoblaster" )

        if   ( cy - ty > 15 and ydeg >= 110 ):
            ydeg -= 1
```

```
            os.system( "echo 1=" + str( ydeg ) + " > /dev/servoblaster" )

    # Calculate and show the FPS
    fps = fps + 1
    sfps = fps / (time.time() - t_start)
    cv2.putText(img, "FPS : " + str( int( sfps ) ), ( 10, 10 ), cv2.FONT_HERSHEY_SIMPLEX, 0.5, ( 0, 0, 255 ), 2 )

    cv2.imshow( "Frame", img )
    cv2.waitKey( 1 )


### Main ########################################################################

if __name__ == '__main__':

    pool = mp.Pool( processes=4 )
    fcount = 0

    camera.capture( rawCapture, format="bgr" )

    r1 = pool.apply_async( get_faces, [ rawCapture.array ] )
    r2 = pool.apply_async( get_faces, [ rawCapture.array ] )
    r3 = pool.apply_async( get_faces, [ rawCapture.array ] )
    r4 = pool.apply_async( get_faces, [ rawCapture.array ] )

    f1, i1 = r1.get()
    f2, i2 = r2.get()
    f3, i3 = r3.get()
    f4, i4 = r4.get()

    rawCapture.truncate( 0 )

    for frame in camera.capture_continuous( rawCapture, format="bgr", use_video_port=True ):
        image = frame.array

        if   fcount == 1:
            r1 = pool.apply_async( get_faces, [ image ] )
            f2, i2 = r2.get()
            draw_frame( i2, f2 )

        elif fcount == 2:
            r2 = pool.apply_async( get_faces, [ image ] )
            f3, i3 = r3.get()
            draw_frame( i3, f3 )

        elif fcount == 3:
            r3 = pool.apply_async( get_faces, [ image ] )
            f4, i4 = r4.get()
            draw_frame( i4, f4 )

        elif fcount == 4:
            r4 = pool.apply_async( get_faces, [ image ] )
            f1, i1 = r1.get()
            draw_frame( i1, f1 )

            fcount = 0

        fcount += 1

        rawCapture.truncate( 0 )
```

帧数上升到了13左右，而且识别框没有延迟。



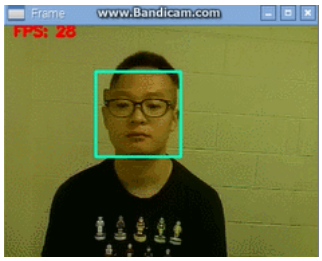
0.5

针对了低帧数问题，我又试了试多核加跳帧...帧数可到28帧左右。

```
### Imports ###################################################################

from picamera.array import PiRGBArray
from picamera import PiCamera
from functools import partial

import multiprocessing as mp
import cv2
import os


### Setup #####################################################################

os.putenv( 'SDL_FBDEV', '/dev/fb0' )

resX = 320
resY = 240

# Setup the camera
camera = PiCamera()
camera.resolution = ( resX, resY )
camera.framerate = 90

t_start = time.time()
fps = 0

# Use this as our output
rawCapture = PiRGBArray( camera, size=( resX, resY ) )

# The face cascade file to be used
face_cascade = cv2.CascadeClassifier( '/home/pi/opencv-2.4.9/data/lbpcascades/lbpcascade_frontalface.xml' )


### Helper Functions ##########################################################

def get_faces( img ):

    gray = cv2.cvtColor( img, cv2.COLOR_BGR2GRAY )
    return face_cascade.detectMultiScale( gray ), img

def draw_frame( img, faces ):

    global fps
    global time_t

    # Draw a rectangle around every face
    for ( x, y, w, h ) in faces:
        cv2.rectangle( img, ( x, y ),( x + w, y + h ), ( 200, 255, 0 ), 2 )

    # Calculate and show the FPS
    fps = fps + 1
```

```
    cv2.imshow( "Frame", img )
    cv2.waitKey( 1 )


### Main ########################################################################

if __name__ == '__main__':

    pool = mp.Pool( processes=4 )

    i = 0
    rList = [None] * 17
    fList = [None] * 17
    iList = [None] * 17

    camera.capture( rawCapture, format="bgr" )

    for x in range ( 17 ):
        rList[x] = pool.apply_async( get_faces, [ rawCapture.array ] )
        fList[x], iList[x] = rList[x].get()
        fList[x] = []

    rawCapture.truncate( 0 )

    for frame in camera.capture_continuous( rawCapture, format="bgr", use_video_port=True ):
        image = frame.array

        if   i == 1:
            rList[1] = pool.apply_async( get_faces, [ image ] )
            draw_frame( iList[2], fList[1] )

        elif i == 2:
            iList[2] = image
            draw_frame( iList[3], fList[1] )

        elif i == 3:
            iList[3] = image
            draw_frame( iList[4], fList[1] )

        elif i == 4:
            iList[4] = image
            fList[5], iList[5] = rList[5].get()
            draw_frame( iList[5], fList[5] )

        elif i == 5:
            rList[5] = pool.apply_async( get_faces, [ image ] )
            draw_frame( iList[6], fList[5] )

        elif i == 6:
            iList[6] = image
            draw_frame( iList[7], fList[5] )

        elif i == 7:
            iList[7] = image
            draw_frame( iList[8], fList[5] )

        elif i == 8:
            iList[8] = image
            fList[9], iList[9] = rList[9].get()
            draw_frame( iList[9], fList[9] )

        elif i == 9:
            rList[9] = pool.apply_async( get_faces, [ image ] )
            draw_frame( iList[10], fList[9] )

        elif i == 10:
            iList[10] = image
            draw_frame( iList[11], fList[9] )
```

```
126             draw_frame( iList[12], fList[9] )
127
128         elif i == 12:
129             iList[12] = image
130             fList[13], iList[13] = rList[13].get()
131             draw_frame( iList[13], fList[13] )
132
133         elif i == 13:
134             rList[13] = pool.apply_async( get_faces, [ image ] )
135             draw_frame( iList[14], fList[13] )
136
137         elif i == 14:
138             iList[14] = image
139             draw_frame( iList[15], fList[13] )
140
141         elif i == 15:
142             iList[15] = image
143             draw_frame( iList[16], fList[13] )
144
145         elif i == 16:
146             iList[16] = image
147             fList[1], iList[1] = rList[1].get()
148             draw_frame( iList[1], fList[1] )
149
150             i = 0
151
152         i += 1
153
154         rawCapture.truncate( 0 )
```



跳帧加多核，强行30帧哈哈，不过还是建议最终使用Demo4。

09/23/2016 Update:
我使用的云台是这个，马达型号是sg90。
ServoBlaster的下载地址是：这个链接需要翻墙。

> 这篇博客节选翻译自我自己的课程报告， 同样的内容也出现于我自己的英文博客， 最后出镜的是我的搭档Andre Heil。

文章标签： 脸部识别 树莓派 opencv raspberry pi
个人分类： 树莓派

applehelle 2018-05-18 13:41:07 #25楼

问：在pi中使用multiprocessing处理视频 在如果进程（2个）中使用imshow，会蹦出来2个窗口分别显示两个进程的结果，显示跟处理会卡，但是pi的CPU占用约
t处理结果到主函数再在主函数中显示的话 虽然会在一个窗口显示视频，但是CPU占用20%多，处理速度上也和单程无差（在pi上甚至还慢了） 请问是什么原因呢
把两个处理结果弄到一个窗口上面显示呢？ 感谢！！！

plehelle 2018-05-16 17:54:24 #24楼

好 仿照您的代码写了一个多进程（4个） 会出现四个窗口显示的情况 请问是为什么？

72691 2018-03-27 08:18:40 #23楼

encv在你运行apt-get install python-opencv的当前目录下安装，目录名不是opencv，如果实在找不到XML文件可以使用find / -name *face.xml搜索找到，跟博主文

st-begain 2017-09-22 22:12:26 #22楼

查看 47 条热评

派3B+ 人脸识别(OpenCV)

树莓派3B+ 人脸识别(OpenCV) 相信大家都看了前面的OpenCV安装和人脸检测教程，有了基础后，现在我们正式进入重头戏——人脸识别 的教程。 注意：该教程面向p

kxwinxp 2017-11-13 17:22:53 阅读数：6112

树莓派用Python+OpenCV做人脸识别

硬件：树莓派 B+ 先安装python-opencv： sudo apt-get install python-opencv Python 2.7 OpenCV3.0 # -*...

dexinzheng 2016-03-04 13:48:49 阅读数：18414

从前端到后台，开发一个完整功能的小程序

微信开发 | wanghui_777

使用树莓派进行简易人脸识别 - CSDN博客

使用树莓派2和OpenCV制作一个简易的人脸识别和追踪系统。 所需硬件需要:...为了解决帧数较低的问题,有一个比较简单的方法就是跳帧,可以不对每一帧图像都...

2018-6-22

树莓派3B使用tensorflow的classify_image进行物体识别 - CSDN博客

看到一个有趣的项目,在树莓派上利用TensorFlow搭建图像识别系统(参考这里),手头正好有个树莓派在吃灰,反正也要入坑TensorFlow,不如就此开头吧。硬件版本是pi 3B..

2018-6-29

树莓派+普通usb摄像头 做人脸识别

http://v.eepw.com.cn/video/play/id/14049 我是根据这个来做的，一开始真是无从下手，妈蛋也不说源代码就在上传到github了，就在那里调几条代码讲讲讲，要不是看..

sinat_20276189 2016-02-13 18:34:49 阅读数：11442

树莓派3B+ 人脸识别(OpenCV) - CSDN博客

树莓派3B+ 人脸识别(OpenCV) 相信大家都看了前面的OpenCV安装和人脸检测教程,有了基础后,现在我们正式进入重头戏——人脸识别 的教程。 注意:该教程面向pytho

2018-6-29

树莓派利用python、opencv识别人脸

树莓派利用python、opencv识别人脸,替换后可识别其它物体。... TensorFlow 图像识别功能在树莓派上的应用 树莓派利用python、opencv识别人脸 立即下载 上传者: lda

2018-6-26

在树莓派上实现face++人脸识别

概述 目前市场上的人脸识别技术已经相对成熟，并且已经广泛的应用于门禁，考勤，美颜软件等方面。但是对于开发者来说自己开发高效准确的算法并不是一件容易的
提供了良好的API接口来让开发...

oBuZuoPiQi 2017-01-14 19:58:40 阅读数：11730

提前申明一下，我搭建的环境版本是2.4.9（很老的版了，但是基本功能都有）源码安装。1.安装依赖文件sudo apt-get updatesudo apt-get install build-esse...

bwm 2018-02-28 12:48:14 阅读数：219

树莓派搭建图像识别基站 - CSDN博客

搭建图像识别基站2017年09月11日 22:14:50 阅读数:977 博主最近参加一个比赛,需要识别白色纸片上的黑色胶带,由于单片机跑不了opencv等一些高级算法,于是就

13

tensorflow实现图像识别 - CSDN博客

一周的作业---用tensorflow框架训练深层网络实现图像识别,代码写完后问题...TensorFlow 图像识别功能在树莓派上的应用 上周TensorFlow 1.0 的发布使之成...

30

树莓派做人脸识别必要环境搭建

建 Ubuntu 16.04 LTS python 2.7 MySQL MySQLdb 前面已经介绍了系统的安装和配置 1.安装Python sudo apt-get install pyt...

geek 2017-10-13 22:29:25 阅读数：741

勃不起来？吃它，让你延长40分钟

美思源 · 顶新

树莓派魔镜总结(包括语音和图像识别) - CSDN博客

1、最终实现方案(树莓派、php+html、arecord、百度语音、face++图像识别) 1.1、硬件部分 因为加了一个开关来控制语音的录入,所以用到了树莓派的中断,因此硬件的..

2018-6-8

树莓派-WebCamera图像采集(OpenCV) - CSDN博客

opencv识别赛道黑线 qq_33894122 2017-09-11 22:14:50 阅读数:769 没有更多推荐了,返回首页 不良信息举报 举报内容: 树莓派-WebCamera图像采集(OpenCV) ...

2018-5-20

树莓派人脸识别python代码及xml模型

2017年01月31日 1.99MB 下载

树莓派上做人脸识别

点击打开链接

kevin198528 2017-11-17 14:21:08 阅读数：386

Google工程师:教你用树莓派+Arduino+TensorFlow搭建图像识别小车

于是我干脆用Arduino负责机械(马达+舵机),相当于身体;树莓派只负责图像识别,相当于大脑。 Arduino不是Linux...

2018-6-11

【opencv】树莓派picamera+opencv人脸识别

所需硬件 需要： 树莓派2树莓派摄像头 云台+舵机（非必须） 安装OpenCV-3.3.0（教程以及安装问题后续发布） sudo apt-get update sudo apt-get upg...

g1fdgfgdf2_ 2017-11-18 18:43:42 阅读数：1471

树莓派-人脸识别资料收集

使用树莓派进行简易人脸识别：http://blog.csdn.net/jireren/article/details/52167791#commentsImportsfrom picamera.arr...

qq_30968657 2016-11-30 21:42:18 阅读数：1763

树莓派人脸识别

网上很多人脸识别教程并没有haarcascades的xml文件，现在给出这个文件和人脸识别的python代码，可惜树莓派计算速度有点慢具体多慢自己试试就知道了别忘了在代径...

Lingdongtianxia 2017-01-31 15:22:44 阅读数：2223

安装完OpenCV是不是已经等不急要试试它的强大能力了？这里教大家使用自带的Haar级联分类器来进行人脸检测。 对于没有安装OpenCV的伙伴，可参考： 树莓派3B+（Ope...

nxp 2017-11-13 13:38:43 阅读数：2141

派利用python、opencv识别人脸

10月23日 4KB 下载

员不会英语怎么行?

教你一个数学公式秒懂天下英语

派上搭建opencv环境

上搭建opencv环境本文记录搭建opencv环境的具体过程。

ghz999 2017-09-19 10:45:01 阅读数：840

OpenCV——面部识别FaceDetector的简单使用

下面是简单的demo，从我的项目中抽取出来，这个方法可以简单检测出人脸，返回的rect可以进行下一步操作 参数文件lbpcascade_frontalface.xml是可以在文件夹内 ../

u012476249 2016-11-29 10:46:56 阅读数：2388

树莓派搭建图像识别基站

opencv识别赛道黑线

qq_33894122 2017-09-11 22:14:50 阅读数：1084

基于树莓派的门禁系统

这几天黑客松比赛， 在大神的带领下稳稳地落幕了， 虽然是第二次参加这个活动了， 但是这一次才是我尽全力的去做自己的东西， 这里也感谢xx大牛的指导 。 也不废

baidu_28138887 2015-05-26 18:22:13 阅读数：4640

树莓派人脸识别门禁系统第二课：Win IOT微软人脸识别API刷脸开门

项目中使用的硬件与软件树莓派3B×1面包板×112伏直流电动门×1双通道5V继电器模块×1直流电源适配器×112V 5A开关电源任何高于3A应该工作。必须是12V。×1网络摄证工作。×...

u012773758 2018-03-21 12:36:36 阅读数：192

人脸识别sdk

人脸识别开源SDK源码 网络应用

基于 树莓派＆opencv&face++开发考勤机（一）

基于 树莓派＆opencv&face++开发考勤机（一） 简述：这是我大创内容的一部分。年前用opencv+qt实现了一个人脸识别的考勤机，但效果不理想，不说你也明白为什么fa...

scylhy 2017-03-12 16:24:00 阅读数：3552

树莓派3B+ 安装计算机视觉库（OpenCV_3+OpenCV_Contrib_3）

树莓派3B+ 安装计算机视觉库（OpenCV_3+OpenCV_Contrib_3） 计算机视觉是一项应用于计算机模式生物视觉的新技术，它使得计算机能代替人眼实现对目标的识别景...

kxwinxp 2017-11-03 12:15:04 阅读数：5629

人脸识别主要算法原理

人脸识别主要算法原理 主流的人脸识别技术基本上可以归结为三类，即：基于几何特征的方法、基于模板的方法和基于模型的方法。 1. 基于几何特征的方法是最早、最要和其他算法结合...

AMDS123 2017-05-25 19:35:18 阅读数：28490

上次博客简单讲了一下调用face++的api来检测人脸。当然，生产环境中要实现复杂的需求光靠这么简单的调用一下api肯定是不行的。这次先来讲一讲face++中实现人脸相对较为复杂...

duyu 2017-12-31 10:57:49 阅读数：1085

+人脸识别

是北京旷视科技有限公司旗下的人脸识别云服务平台，Face++平台通过提供云端API、离线SDK、以及面向用户的自主研发产品等形式，将人脸识别技术广泛应用于。Face+...

2449851 2014-06-07 12:11:13 阅读数：2429

员不会英语怎么行?

教你一个数学公式秒懂天下英语

派3B使用tensorflow的classify_image进行物体识别

树莓派3B 软件环境： 1.python3.4 2.tensorflow-1.1.0-cp34-cp34m-linux_armv7l.whl 安装呢，其实网上特别多教程，这里我是...

tenJzhe 2017-07-22 20:49:52 阅读数：6096

TensorFlow 图像识别功能在树莓派上的应用

上周 TensorFlow 1.0 的发布使之成为最有前景的深度学习框架，也在中国 AI 社区中掀起了学习 TensorFlow 的热潮，不过光跑例子怎能脱颖而出？本文是数据科学公司

heyc861221 2017-02-21 10:04:01 阅读数：1308

树莓派自带摄像头OpenCV图像识别-二维码识别

1、安装树莓派镜像，SSH,VNC等这里就不介绍了很简单大家自行百度。2、我没有用VNC 所以装了teamviewer用于树莓派远程控制安装教程参考：http://blog.csdn.net/

x115104 2017-12-23 10:37:57 阅读数：1294

树莓派-WebCamera图像采集（OpenCV）

（1）在树莓派USB接口中插入webcamera，使用如下命令检测是否检测到camera cd /dev ls | grep video 如果有个设备名字是videox（x是...

u011923796 2015-07-31 18:00:40 阅读数：3348

Google工程师：教你用树莓派+Arduino+TensorFlow搭建图像识别小车

雷锋网按：本文作者赵智沉，Google软件工程师。来自知乎专栏：赵智沉的作坊。雷锋网(公众号：雷锋网)获授权转载。 从买第一个Arduino套装开始，我接触机器人有近才...

x32sky 2017-04-07 13:42:15 阅读数：4059

如何安装树莓派摄像头模块 Linux

树莓派摄像头

树莓派魔镜总结（包括语音和图像识别）

我实在是太懒了，现在才来写这篇博文。 这里我将总结做这个项目所用的思路，以及中间出现的各种问题还有问题的解决办法。1、最终实现方案（树莓派、php+html、音、face++图...

cscshaha 2017-04-10 22:25:48 阅读数：3691

自动跟随机器人教程（五）软件部分 树莓派+电脑 摄像头图像回传

既然你熟悉了Socket编程，也可以另外再写一个类似的简易服务器客户端程序。树莓派仍然作为服务器，电脑仍然作为客户端。只不过现在是树莓派发送，电脑接收。这指令，而是树莓派USB摄像头...

shukebeta008 2018-02-18 19:50:12 阅读数：677

原创 4 | 粉丝 23 | 喜欢 1 | 评论 50

等级： 博客 2 访问： 3万+
积分： 235 排名： 33万+