**基于树莓派的地面饮料瓶识别**

**1 简介**

本项目采用树莓派驱动摄像头采集实时图像，使用OpenCV视觉库处理图像，进而完成饮料瓶的识别。

**2 硬件平台**

树莓派

DC5V 2A电源

Micro USB数据线

micro SD卡（建议8G及以上，Class10）

micro SD卡读卡器（如果树莓派已经安装好系统，可不用）

500万树莓派摄像头

HDMI线

显示器（带HDMI口，没有需要转接线）

USB鼠标、USB键盘

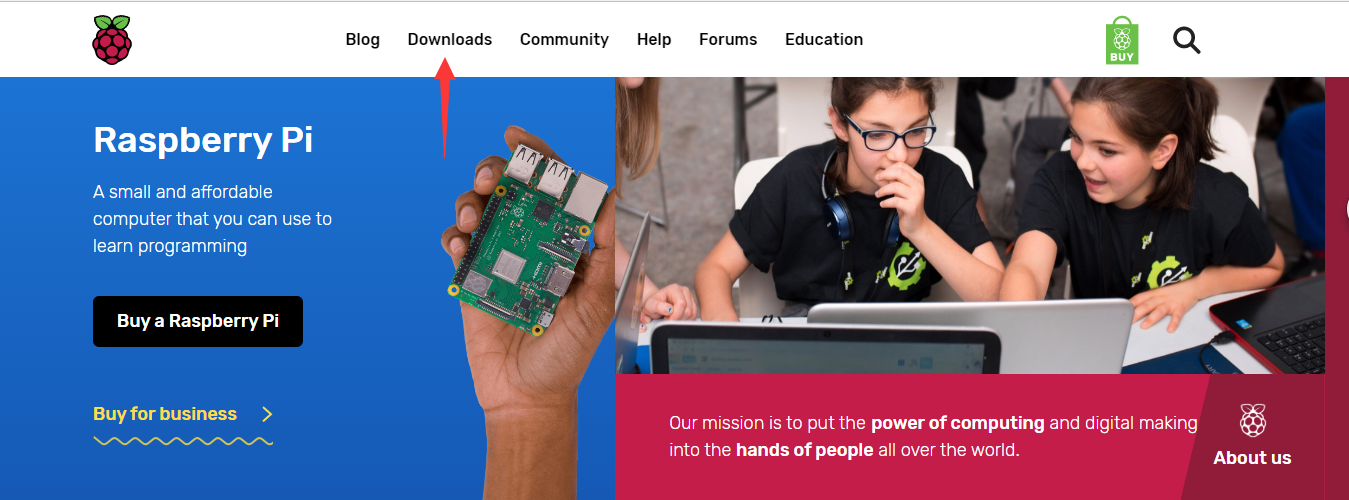
无线网卡（树莓派3B自带无线网卡，可以用，但是不好用）

**3 安装树莓派系统**

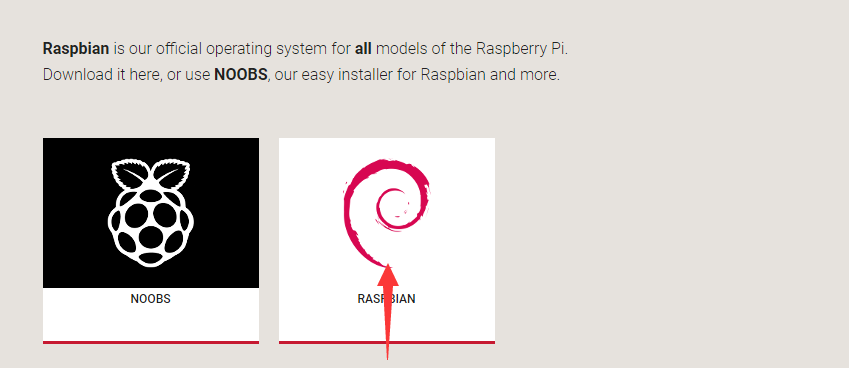
首先，登陆树莓派的官网

<https://www.raspberrypi.org/>

点击DOWNLOADS进行系统镜像的下载



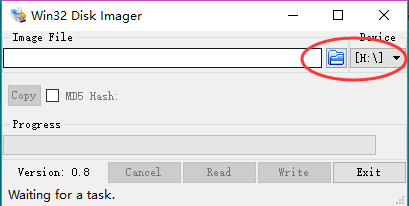
选择RASPBIAN



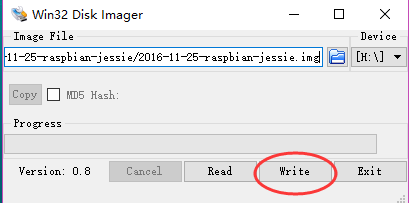
选择Download ZIP，并解压得到“.img"文件



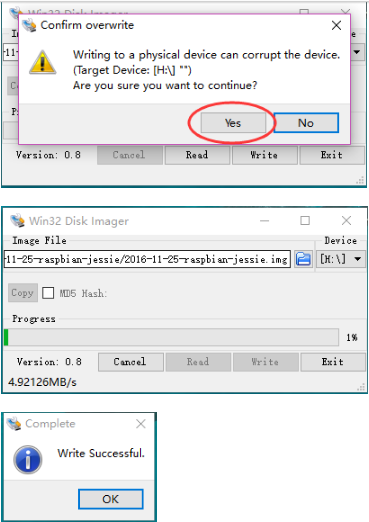
下载一个工具Win32 Disk Imager专门用来写“.img"文件到SD卡。使用读卡器将SD卡插入电脑，并在Win32 Disk Imager软件选择该盘。



点击蓝色“文件夹”图标选择镜像，完成后，点击Write，进行写卡



提示框直接点YES，格式化SD卡



SD卡变成了,boot:卷标，并且容量变小。弹出SD插入树莓派即可。

**4 安装OpenCV**

树莓派系统安装好之后，将鼠标、键盘、显示器、无线网卡插入树莓派，接上供电线，自动开机，开机之后连接好WIFI保证能够正常上网，可打开浏览器测试。

**4.1 更新软件源**

若已更新可跳过，若未更新软件源可打开终端执行以下命令：

sudo nano /etc/apt/sources.list

删除原文件所有内容，替换为

deb http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ jessie main contrib non-free

deb-src http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ jessie main contrib non-free

Ctrl+O保存，Ctrl+X退出

**4.2 安装OpenCV依赖库**

打开终端，然后依次执行下列命令：（注意所有与apt-get相关的下载指令，在下载过程中不可断网，也不能终止进程，需要等待下载完成，不然会出现一些麻烦问题）

// 软件源更新

sudo apt-get update

// 升级本地所有安装包，最新系统可以不升级，版本过高反而需要降级才能安装

sudo apt-get upgrade

安装构建OpenCV的相关工具：

// 安装build-essential、cmake、git和pkg-config

sudo apt-get install build-essential cmake git pkg-config

// 安装jpeg格式图像工具包

sudo apt-get install libjpeg8-dev

// 安装tif格式图像工具包

sudo apt-get install libtiff5-dev

// 安装JPEG-2000图像工具包

sudo apt-get install libjasper-dev

// 安装png图像工具包

sudo apt-get install libpng12-dev

再安装视频I/O包（注意最后一个包的数字“4”后面是“L”）：

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

下面安装gtk2.0（树莓派很可能下载错误，更换中科大或者清华源即可，ubuntu有可能出现包版本过高的情况，需要将依赖包降级安装）：

sudo apt-get install libgtk2.0-dev

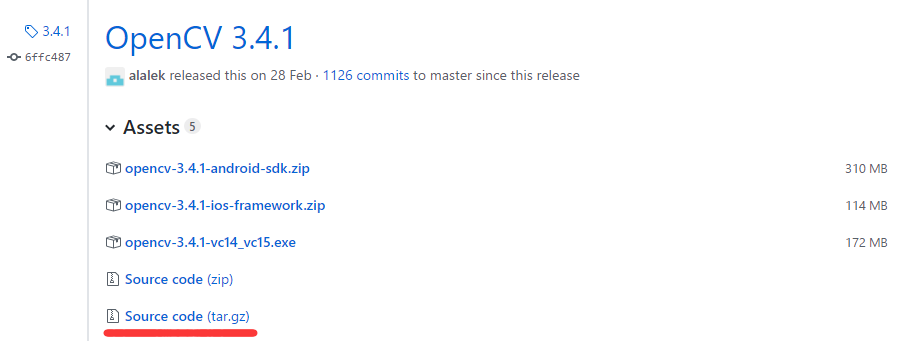
优化函数包：

sudo apt-get install libatlas-base-dev gfortran

**4.3 下载OpenCV源码**

网上都是用指令： wget -O opencv-3.4.1.zip <https://github.com/Itseez/opencv/archive/3.4.1.zip>下载OpenCV源码，但是会出现编译问题！因此需要在个人电脑上访问

<https://github.com/opencv/opencv/releases> 下载opencv-3.4.1版本源码



下载后解压得到opencv-3.4.1文件夹，使用U盘拷贝该文件夹。

在树莓派终端新建文件夹opencv

mkdir opencv

打开

cd opencv

将U盘插入树莓派并将opencv-3.4.1文件夹复制到opencv文件夹下

cp –r /media/pi/\*\*\*/opencv-3.4.1 /home/pi/opencv

\*\*\*为U盘的名字，也可以直接像Windos那样直接界面操作复制。

完成后在终端输入 ls 查看opencv-3.4.1是否存在。

// 当前路径下，下载OpenCV\_contrib库：

wget -O opencv\_contrib-3.4.1.zip https://github.com/Itseez/opencv\_contrib/archive/3.4.1.zip

// 解压OpenCV\_contrib库：

unzip opencv\_contrib-3.4.1.zip

**4.4 编译OpenCV源码**

// 打开源码文件夹

cd opencv-3.4.1

之后我们新建一个名为release的文件夹用来存放cmake编译时产生的临时文件：

// 新建release文件夹

mkdir release

// 进入release文件夹

cd release

设置cmake编译参数，安装目录默认为/usr/local ，注意参数名、等号和参数值之间不能有空格，但每行末尾“\”之前有空格，参数值最后是两个英文的点：

sudo cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv/opencv\_contrib-3.4.1/modules \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

-D BUILD\_EXAMPLES=ON ..

出现

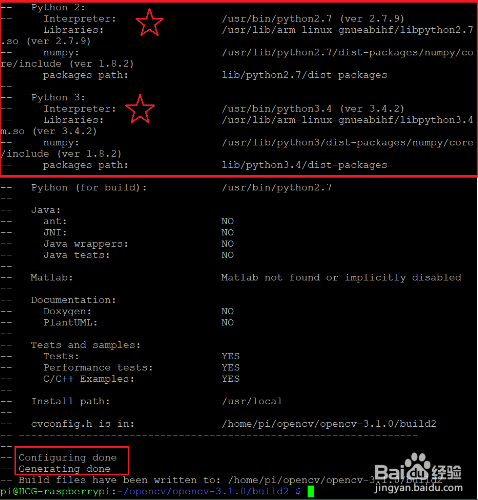
Configuring done

Generating done

编译完成。如图所示，确保图中显示numpy的路径，如果没有（一般都有）需要安装。

sudo apt-get install python3-numpy python-numpy

然后重新cmake



修改树莓派的 swap 的大小，不然编译过程中会挂起

sudo nano /etc/dphys-swapfile

将 CONF\_SWAPSIZE 的值修改为1024（默认为100）

然后，重新启动 dphys-swapfile 文件服务：

sudo /etc/init.d/dphys-swapfile restart

// 编译，以管理员身份，否则容易出错

编译时间会很长，若使用

sudo make

网上表明需要4小时左右，如果树莓派安装了散热片，有小风扇对板子降温可使用四线程，时间在1小时左右

sudo make -j4

// 安装

sudo make install

// 更新动态链接库

sudo ldconfig

**4.5 测试OpenCV**

到这里，OpenCV的编译完成，已经可以正常使用了，下面提供一个Python语言的测试程序，用来测试OpenCV是否正常：

回到根目录

cd ~

新建python文件

nano opencvTest.py

输入代码

# -\*- coding：utf-8 -\*-

import cv2

import numpy as np

cv2.namedWindow("gray")

img = np.zeros((512,512),np.uint8)

cv2.line(img,(0,0),(511,511),255,5)

cv2.imshow("gray",img)

while True:

key=cv2.waitKey(1)

if key==ord("q"):

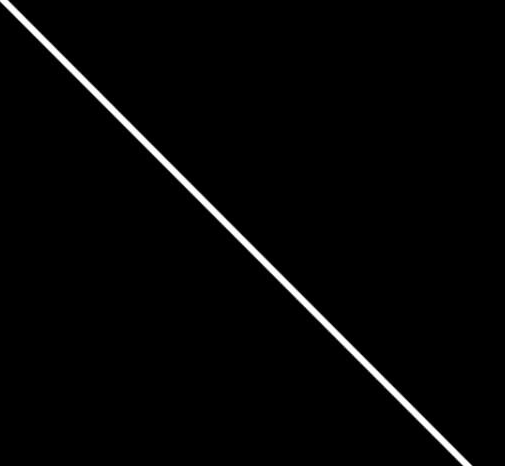
break

cv2.destoryWindow("gray")

保存退出之后，在终端输入

python opencvTest.py

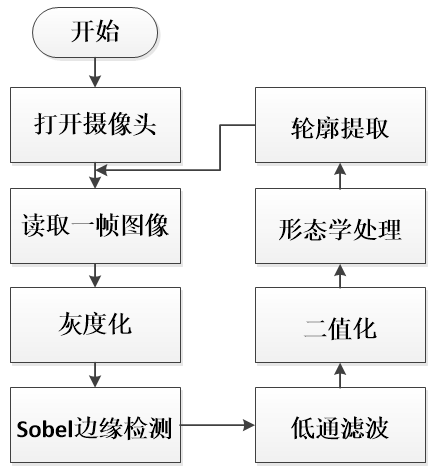
显示结果



**5 软件部分**

**5.1 算法流程**

项目采用摄像头读取当前环境图像，使用OpenCV视觉库对图像进行处理，包括灰度化、Sobel边缘检测、低通滤波、二值化、形态学操作、轮廓提取。算法流程图如下图所示。



**5.2 程序讲解**

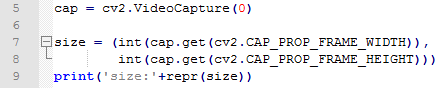
**5.2.1 objectldentificat.py文件**



导入OpenCV库、numpy库和目标检测函数sobclde

NumPy是Python语言的一个扩充程序库，支持高级大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库，运算效率极好。

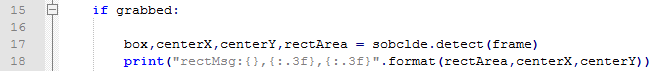
sobclde是自行编写的检测函数，在sobclde.py中讲解



第5行打开摄像头，7-9读取当前图像尺寸并输出。测试用。

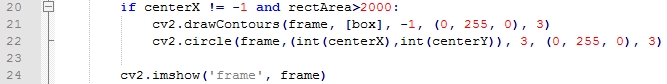


当摄像头打开时，读取一帧图像，其中grabbed的值为True 或者False，代表有没有读取到图片。第二个参数frame表示截取到一帧的图片。



当grabbed为True时，17行调用sobclde中识别函数detect，传入的参数为当前读到的一帧图片，返回值box代表找到的物体的最小外接矩形的4个顶点，centerX,centerY为矩形的中心坐标，rectArea为矩形的面积。

18行输出最小外接矩形的面积和中心，中心保留三位小数。

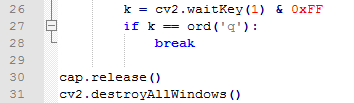


如果检测到当前的图像不存在物体，返回的centerX值为-1，当该值不为-1时，即检测到物体存在。为了避免一些不必要的误判，通过限制物体最小外接矩形的面积来避免，即当rectArea值大于2000时才认为是自己想要的物体目标。该值受图像远近的影响，相同的物体距离摄像头不同的距离，rectArea值也会不同。可以在输出界面观察18行输出的该值实际大小，根据实际需求调节2000这个值。

21行在图像上画出该矩形，第一个参数为需要操作的图像，第二个参数为轮廓，即矩形四个顶点，第三个参数-1表示画出所有存在的轮廓，第四个参数为颜色，第五个为粗细。

22行画出矩形的中心，第一个参数为需要操作的图像，2,3参数为中心，第四个为半径，第五个为颜色，第六个为粗细。

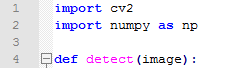
24行显示图片，第一个参数表示窗体名称，第二个是要显示的图片。



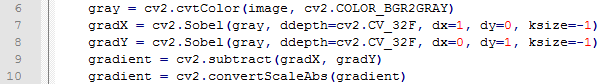
26行表示，每1ms判断是否按下键盘的按键，27、28行如果按下“q”则退出。

30、31行表示退出时，释放摄像头，关闭所有窗口。

**5.2.1 sobclde.py文件**



导入OpenCV和NumPy库并定义一个名为detect的函数，带1个参数，就是需要检测的图像



6行对输入图像进行灰度化处理，第一个参数为需要处理的图像，第二个参数为转化类型。

7、8行对输入图像求x、y方向的梯度，第一个参数为需要处理的图像，第二个参数为深度，由于Sobel函数求完导数后会有负值，还有会大于255的值。而原图像是uint8，即8位无符号数，所以Sobel建立的图像位数不够，会有截断。因此使用32位有符号的数据类型。dx和dy表示的是求导的阶数，0表示这个方向上没有求导，ksize是Sobel算子的大小。

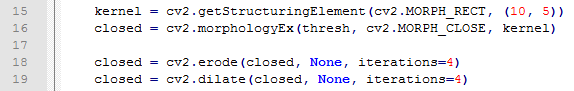
9行在x方向上减去y方向上的梯度，通过这个减法，留下具有高水平梯度和低垂直梯度的图像区域。

10行转回uint8形式，否则将无法显示图像。



12行使用9\*9的内核对梯度图进行平均模糊，去除高频噪声。第一个参数为操作图像，第二个为内核。

13行二值化，梯度图像中不大于130的任何像素都设置为0（黑色）。 否则，像素设置为255（白色）。第一个参数为操作图像，2、3为阈值，第4个为阈值类型。

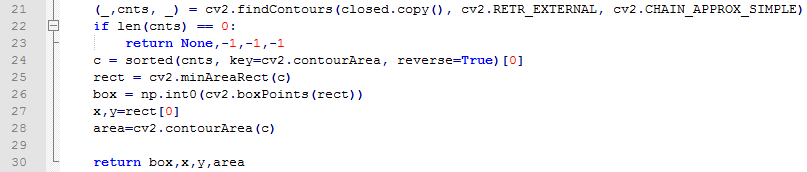


15行使用cv2.getStructuringElement构造一个长方形内核，10\*5。

16行形态学闭运算消除一些小型的黑洞，即图中黑色区域，使得物体形状更突显。第一个参数为操作图像，第二个为运算类型，第三个为内核。

18行为形态学腐蚀操作，腐蚀图像中白色像素，以此来消除一些小的斑点。第三个参数为腐蚀次数。

19行为形态学膨胀操作，使剩余的白色像素扩张并重新增长回去。



21行找出物体的轮廓，cv2.findContours()第一个参数为操作图像，第二个参数表示轮廓的检索模式，第三个参数为轮廓的近似方法。返回三个参数，这里只关心第二个参数，轮廓cnts。

22、23行判断cnts的长度，即是否有轮廓存在，如果没有返回对应的外界矩形顶点为None，中心、面积为-1。

24行对轮廓进行排序（轮廓按面积大小降序排列）。假设面积最大的轮廓就是目标物体的位置。

25行计算该轮廓的最小外界矩形包络。

26行得到矩形四个顶点。

27行得到矩形的中心点。

28行计算矩形面积。

30行返回矩形顶点、中心、面积。

**6 功能测试**

**6.1 打开摄像头**

将树莓派关机，断电（要断开供电的线）。把摄像头连接到树莓派板子上，注意接口的方向！对于树莓派3B，摄像头有蓝色条的一边和USB在一边，如图，黑的卡扣可以向上拉开。接好之后，上电，开机。注意，不要带电插拔摄像头！



在终端输入

ls /dev/

查看是否有video0，若没有命令行输入

sudo nano /etc/modules

// 进入编辑界面后，在末尾添加输入

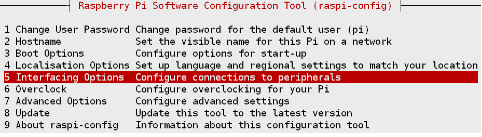
bcm2835-v4l2

保存退出

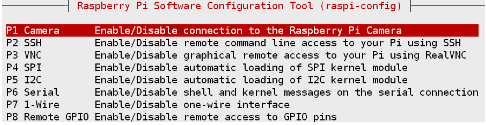
修改Raspberry的启动配置使能项

sudo raspi-config

得到如下的配置界面：



选择Interfacing Option，选中Select然后Enter进入，如下图所示：



接下来问你是否同意使能Pi camera，选择是然后会让你重启，重启就好了：



选择 “是”

重启完之后，

ls /dev/

查看是否有video0，若还没有，检查硬件连接是否有问题，或者摄像头本身有问题。

若没问题，终端输入

nano cameraTest.py

输入测试代码

#!/usr/bin/python

import cv2

import time

cv2.namedWindow("camera", 1)

capture = cv2.VideoCapture(0)

while True:

ret,prev = capture.read()

if ret==True:

cv2.imshow("camera", prev)

else:

break

key = cv2.waitKey(1) & 0xFF

if key == ord('q'):

break

cv2.destroyAllWindows()

保存退出之后，在终端输入

python cameraTest.py

测试成功后可以看到界面出现摄像头的视频，点击键盘的“q”退出，摄像头测试完成。

**6.2 识别算法测试**

使用U盘将附件中的objectldentificat.py和sobclde.py两个文件拷贝到树莓派中，任意目录均可，保证两个文件在一个目录下。

打开终端，输入如下指令

python objectldentificat.py

显示器有图像输出，图中绿色的框代表识别到的物体。



命令窗口也有持续数据输出，第一个参数是识别出的图像的面积，2、3为中心坐标。

