

## 实验 1 基于 OpenCV 的颜色特征识别

### 实验目的：

通过对图像捕捉和颜色特征提取，了解机器视觉的一般工作流程，掌握 OpenCV 的使用及基本图像处理算法。

### 实验内容：

1. 图像传感器驱动应用
2. 图像直方图生成
3. 颜色特征设定及目标识别
4. 基于颜色特征的应用扩展

### 实验设备：以下平台二选一

平台一：树莓派, Linux OS, OpenCV 开发库

平台二：个人计算机, Linux OS, USB 摄像头, OpenCV 开发库

### 预备知识：

1. 数字图像处理基础
2. C++、python 编程基础

### 程序文件：

```
pi@raspberrypi ~/test $ ls
common.py  common.pyc  lan.jpg  test.py  video.py  video.pyc
pi@raspberrypi ~/test $
```

其中 video.py 和 common.py 是 opencv 自带的 simple 文件，在 test.py 中会调用这两个文件，lan.jpg 是参考图像文件用于学习目标的颜色特征，.pyc 文件是程序中间生成文件。

### 实验步骤：

#### 1. 图像传感器驱动应用

将提供的 USB CMOS 型图像传感器连接至树莓派主板上任一 USB 端口，并安装图像传感器驱动程序，能够实时捕获图像帧并显示。

- (1) 如果选择树莓派为实验平台，连接图像传感器至树莓派主板，实验采用图像传感器型号为 JD-202，如图 1 所示；



默认的树莓派主板型号是树莓派第二代，操作系统为 RASPBIAN (2015-05-05)，具体安装方式及相关配置请参考 <http://www.cnblogs.com/abel/p/3441175.html>；

如果选择个人计算机，该步骤可跳过；

- (2) 在摄像头插上以后便可直接使用，无需驱动，可以使用 video.py 测试摄像头。整体连线见下图效果。

测试方式：在当前文件夹路径下执行 ./video.py，如设备正常则生成一个图像窗口并实时显示摄像头画面。



图 1 设备连接

## 2. 图像直方图生成

了解直方图的概念，理解像素灰度值的概率分布函数，根据提供的 OpenCV 框架，获得直方图结果，并显示当前图像直方图。

实验文件夹下有样本图像 lan.jpg，从该图像上获取颜色概率用于图像颜色目标搜索，并绘制该图像的颜色直方图。



图 2(a) 样本图像

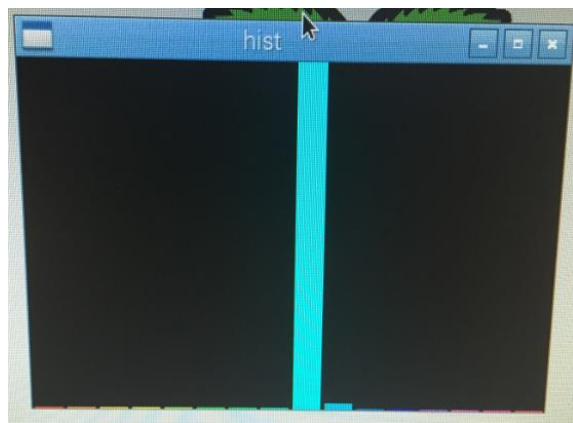


图 2(b) 样本图像直方图

- (1) 为了打开样本图像，首先创建一个窗口对象，在 OpenCV 中通过对窗口命名的方式来创建一个对象，如：

```
cv2.namedWindow('camshift')
if color == 0:
    self.roi = cv2.imread('hong.jpg')
    self.flag = "Hong"
else :
    self.flag = "Lan"
    self.roi = cv2.imread('Lan.jpg')
```

这里，创建了一个窗口，并命名为“camshift”，为后续窗口访问提供资源，程序中 self.flag 为一全局 string 型变量，根据目标颜色取相应数值，如目标颜色为红色时则置 self.flag 为“hong”并读入 hong.jpg，将得到的数据传递给变量 self.roi。

- (2) 计算并显示颜色概率直方图

图像直方图为图像中不同颜色和亮度的像素点的统计分布，横轴为颜色和亮度纵轴为该颜色和亮度下像素点的个数，有一维直方图(针对灰度图像)和二维直方图(针

对彩色图像)。

由于该样本图像已经目标图像都为彩色的，故需要二维直方图，计算直方图的函数为 `cv2.calcHist(images; channels; mask; histSize; ranges; hist; accumulate[])`，其中

1. `images`: 原图像（图像格式为 `uint8` 或 `float32`）。当传入函数时应该用中括号 `[]` 括起来，例如：`[img]`。
2. `channels`: 同样需要用中括号括起来，它会告诉函数我们要统计那幅图像的直方图。如果输入图像是灰度图，它的值就是 `[0]`；如果是彩色图像的话，传入的参数可以是 `[0]`, `[1]`, `[2]` 它们分别对应着通道 `B`, `G`, `R`。
3. `mask`: 掩模图像。要统计整幅图像的直方图就把它设为 `None`。但是如果你想统计图像某一部分的直方图的话，你就需要制作一个掩模图像，并使用它。
4. `histSize`: `BIN` 也就是像素某一灰度值范围内的像素点数目。也应该用中括号括起来，例如：`[256]`。
5. `ranges`: 像素值范围，通常为 `[0, 256]`；

注意参数最后 `accumulate[]` 表示 `channel`、`range` 和 `histSize` 可以通过中括号写入两个数值，以此来表示两个不同颜色通道。

实验中采用彩色图像，因此要考虑每个像素的颜色（Hue）和饱和度（Saturation），需要将图像的颜色空间从 `BGR` 转换到 `HSV`，根据 `S`、`V` 这两个要素绘制 2D 直方图，相应地，`calcHist()` 函数的参数也要有两个通道的表示，例如：

- `channels=[0, 1]` 因为我们需要同时处理 `H` 和 `S` 两个通道。
- `bins=[180, 256]` `H` 通道为 180，`S` 通道为 256。
- `range=[0, 180, 0, 256]` `H` 的取值范围在 0 到 180，`S` 的取值范围在 0 到 256。

示例代码为：`channels` 只有一个参数？像素颜色取值只到 180？

```
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask_roi = cv2.inRange(hsv_roi, np.array((0., 60., 32.)), np.array((180., 255., 255.)))

hist = cv2.calcHist([hsv_roi], [0], mask_roi, [16], [0, 180])
cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX);
self.hist = hist.reshape(-1)
self.show_hist()
```

代码中引用了掩膜来进行图像区域限定，变量为 `mask_roi`，`np.array(0,60,32.)` 表示，

统计好的直方图进行显示时，先定义每个颜色范围的像素点个数，然后通过矩形表示该像素点总数数值，最后生成一个显示窗口，显示矩形图像，`self.show_hist()` 代码如下：

```
def show_hist(self):
    bin_count = self.hist.shape[0]
    bin_w = 24
    img = np.zeros((256, bin_count*bin_w, 3), np.uint8)
    for i in xrange(bin_count):
        h = int(self.hist[i])
        cv2.rectangle(img, (i*bin_w+2, 255), ((i+1)*bin_w-2, 255-h), (int(180.0*i/bin_count), 255, 255), -1)
    img = cv2.cvtColor(img, cv2.COLOR_HSV2BGR)
    cv2.imshow('hist', img)
```

### 3. 颜色特征设定及目标识别

(1) 通过 `self.cam.read()` 读取视频帧

```
while True:
    for frame in self.cam.capture_continuous(self.rCa, format='bgr', use_video_port=True):
        ret, self.frame = self.cam.read()
        self.frame = frame.array
        vis = self.frame.copy()
        vis = copy.deepcopy(self.frame)
        hsv = cv2.cvtColor(self.frame, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, np.array((0., 60., 32.)), np.array((180., 255., 255.)))
```

(2) 通过参考图片颜色概率的反向投影函数在视频帧中找到目标颜色，通过 `camshift` 函数得到目标颜色在帧中的坐标信息并存入 `track_box`，持续跟踪该物体。

```
prob = cv2.calcBackProject([hsv], [0], self.hist, [0, 180], 1)
prob &= mask
term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1 )
track_box, self.track_window = cv2.CamShift(prob, self.track_window, term_crit)
```

(3) 通过 `cv2.ellipse` 画椭圆将图中的目标物体标识出来，参数中 `track_box` 为目标颜色坐标值，(0,0,255)选择 BGR 模式中的红色，2 为椭圆线圈像素宽度。

```
cv2.ellipse(vis, track_box, (0, 0, 255), 2)
```



(4) 当目标物体变得太远或移出画面时，`track_box` 值将无法计算，判断 `track_box` 数值小于 1 时，需要重置算法，重新搜索并跟踪。

```
if track_box[1][1] <= 1:
    self.tracking_state = 0
    self.start()
```

当 `track_box` 范围小于一个阈值时，便把跟踪状态置为 0，重新 `start`，即给 `track_box` 重新赋值为整个图像范围，从新的范围开始搜索。

#### 4. 基于颜色特征的应用扩展

在前 3 步的基础上，尝试修改样本图像，或者修改代码，进一步地进行机器视觉处理，如识别红色、绿色或橙色特征区域等。

### 实验报告

1. 用自己的话给出上述各步骤所调用函数 API 的原理理解，代码分析和实验结果；
  2. 回答问题：(1)什么是图像的直方图？ (2) HSV 空间通过哪几个维度表达颜色分布？
- 文档以 word 形式提交，以“**学号\_姓名\_Lab1**”格式命名，每人一份，提交至教辅系统。