

基于树莓派的人工智能小 车教程 v1.1

Original

mumu 林

目录

第一章 颜色和物体追踪.....	1
1.1 OpenCV 简介	1
1.1.1 OpenCV 简介	1
1.1.2 OpenCV 例程演示	1
1.2 树莓派安装 OpenCV3.4.0	2
1.2.1 windows 平台安装 python-OpenCV （目前只支持 Python2）	2
1.2.2 树莓派安装 python-opencv	4
1.2.3 源码编译方式安装 OpenCV	4
1.3 快速上手 OpenCV	4
1.3.1 图像的读入、显示和保存.....	4
1.3.2 视频的读入、显示和保存.....	5
1.3.3 绘制直线、矩形、圆.....	6
1.3.4 openCV 创建滑动条	7
1.4 OpenCV 的颜色空间	9
1.4.1 RGB 和 HSV 颜色空间	9
1.4.2 颜色空间转换和颜色识别.....	9
1.4.3 物体追踪.....	11
1.5 Opencv 实现颜色识别和物体追踪.....	13
1.5.1 颜色识别.....	13
1.5.2 跟屁虫小车（跟随小车）	17
第二章 二维码导航及巡线.....	19
2.1 AI 车的运动控制	19
2.1.1 电机驱动原理讲解.....	19
2.1.2 综合运动控制.....	20
2.2 二维码识别.....	21
2.2.1 二维码简介	21
2.2.2 二维码识别技术.....	22
2.2.3 基于二维码信息识别的导航小车.....	26
2.3 基于 openCV 的巡线小车	27
2.3.1 单线巡线.....	27
2.3.2 双线巡线.....	28
第三章 语音识别及语音合成.....	30
3.1 语音识别功能介绍.....	30
3.1.1 语音识别和语音合成简介	30
3.1.2 树莓派相关环境配置	30
3.2 百度云语音识别 Python-SDK 使用.....	35
3.2.1 python-SDK 的安装.....	35
3.2.2 Python-SDK 实现语音识别和语音合成.....	37
3.2.3 语音识别结果及错误分析	41
3.3 离线语音唤醒.....	42
3.3.1 离线唤醒引擎 SnowBoy.....	42
3.3.2 离线唤醒控制 LED 灯	43

3.3.3 修改个性唤醒词.....	45
3.4 语音唤醒及语音识别实现语音交互.....	46
3.4.1 语音控制及语音机器人.....	46
3.4.2 控制 AI 车的运动	49
第四章 基于人工神经网络的自动驾驶小车.....	50
4.1 有趣的神经网络.....	50
4.1.1 浅谈神经网络.....	50
4.1.2 构建自己的神经网络.....	53
4.1.3 识别手写数字.....	55
4.2 神经网络构建自动驾驶小车.....	56
4.2.1 自动驾驶小车的工作原理.....	56
4.2.2 自动驾驶小车的实现.....	56
4.2.3 测试.....	60

Original

第一章 颜色和物体追踪

本教程说明：该教程适用于刚接触计算机视觉开发或者对计算机视觉开发感兴趣的读者，如果是大神级别的任务，绝对不适合。本教程来自于笔者开发经验，希望可以帮助大家在学习的道路上少走弯路。程序开发环境基于 `python2.7`, `opencv3.4.0` 版本，硬件平台为树莓派 `3Bplus` 版本，用 `python3` 的可能兼容性有问题。本教程的从简单的图像处理起步，到实现自动驾驶终结，中间穿插语音合成和语音识别的功能，由简单到复杂。

1.1 OpenCV 简介

1.1.1 OpenCV 简介

本小节主要介绍 AI 小车图像处理部分使用的开源计算机视觉环境 `OpenCV`。`OpenCv` 是 `Open Source Computer Vision Library` 的缩写，是一个基于开源发行的跨平台计算机视觉库，它实现了图像处理和计算机视觉方面的很多通用算法，已经成为计算机视觉领域最有力的研究工具之一。`OpenCV` 的底层由 C 和 C++ 编写，轻量且高效，可以运行在多个操作系统上(`Linux`、`Windows`、`Mac`、`Andorid`、`iOS` 等)，同时提供了多种编程语言的 API 接口，本教程是基于 `Python` 的接口进行一些简单的计算机视觉处理。

`OpenCV` 的应用领域：机器人视觉、模式识别、机器学习、工厂自动化生产线产品检测、医学影像、摄像机标定、遥感图像等

`OpenCV` 可以解决的问题：人机交互、机器人视觉、运动跟踪、图像分类、人脸识别、物体识别、特征检测、视频分析、深度图像等

`OpenCV` 的实时性能也非常出色，可实时应用，为实时场景的视觉环境开发提供了很好的解决方案，总之 `OpenCV` 是计算机视觉开发方面优秀的开源工具，如果对计算机视觉开发感兴趣可以多了解这方面的知识：

`OpenCV` 官方主页：<https://www.opencv.org>

`OpenCV` 中文论坛：<http://www.opencv.org.cn>

`OpenCV` CSDN 论坛：<https://bbs.csdn.net/forums/OpenCV>

1.1.2 OpenCV 例程演示

本小节演示几个好玩的 `OpenCV` 官方例程

光流： `opt_flow.py`

程序位于 `opencv-xxx\samples\python` 目录下，光流算法可用于运动图像的分析。

目标追踪： `Camshift.py`

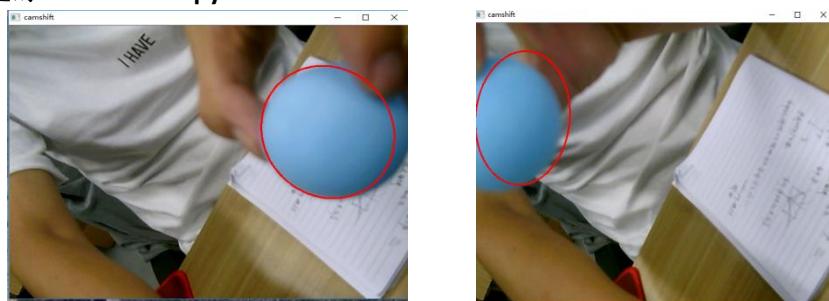


图 1.1 目标跟踪

人脸识别：facedetect.py

人脸识别是图像处理重要的应用之一，接下来演示一下官方给出的例程，识别一下鲁迅先生。

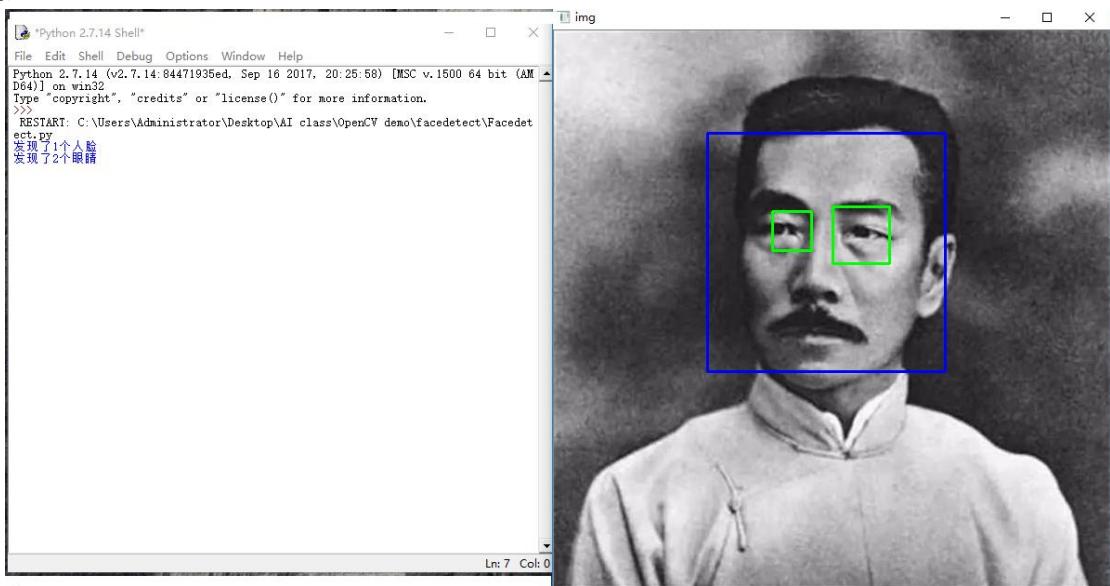


图 1.2 人脸识别

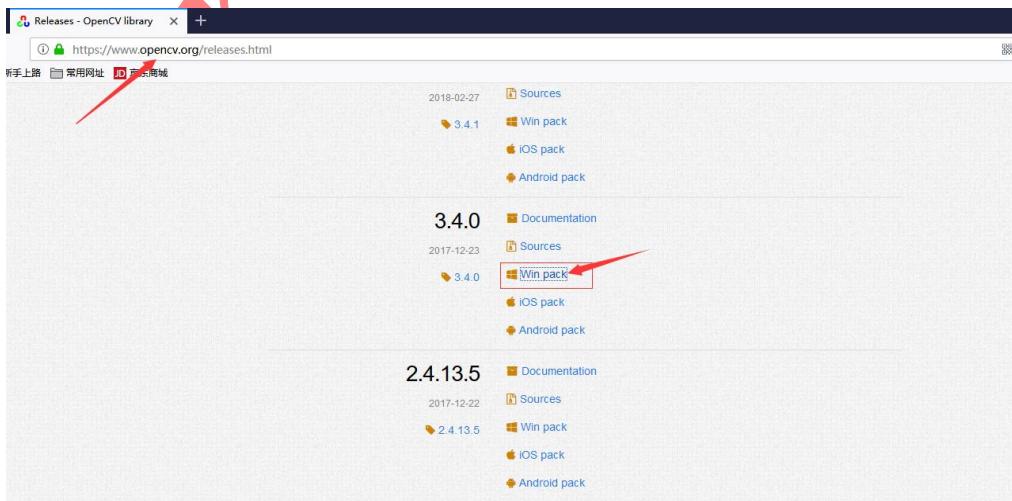
简单的介绍了 OpenCV，大家对 OpenCV 有了初步的了解，下面让我们正式进入学习 OpenCV 吧

1.2 树莓派安装 OpenCV3.4.0

本小节将正式进入 OpenCV 的学习，首先配置 OpenCV 的开发环境，本教程的终极目标是做一个智能小车，所以选择树莓派 3B+为开发平台，Python 为开发环境，此外适当的修改程序，代码都可以在安装过 OpenCV 的 windows 平台运行。下面分别介绍如何安装 OpenCV：

1.2.1 windows 平台安装 python-OpenCV（目前只支持 Python2）

1、进入 OpenCV 官网下载 windows 安装包



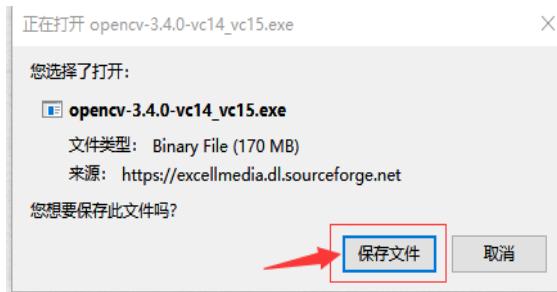
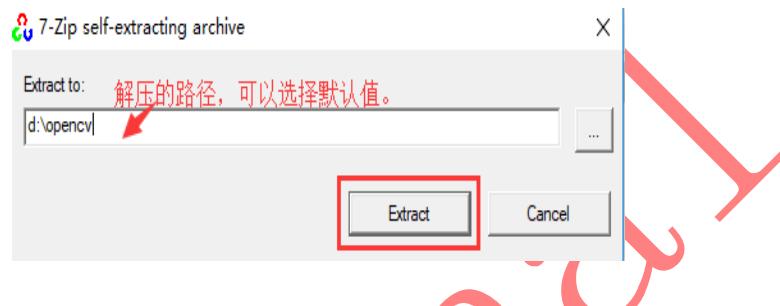


图 1.3 安装包下载

2、下载完成进行解压，路径可以自己修改。建议选择 C:\opencv



解压过程

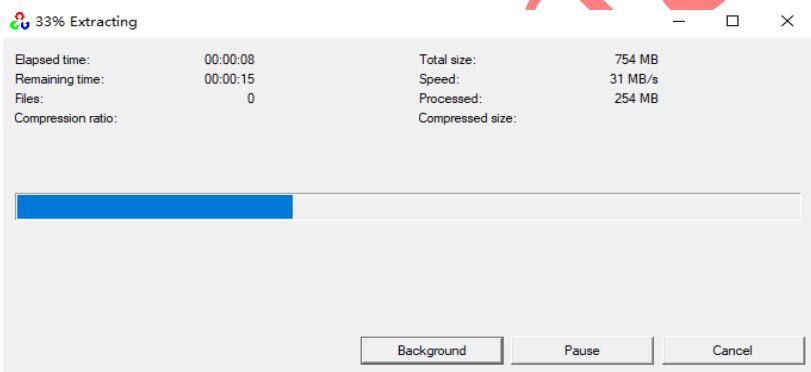


图 1.4 安装包解压

3、复制 cv2.pyd 到 python 的安装位置下的

找到刚刚解压的路径，在\build\python\2.7\x64 目录下有一个名为 cv2.pyd 的文件
如果你的电脑是 32 位的选择 x86 目录，将这个文件复制到 python2.7 安装目录下
\Lib\site-packages 文件夹下。我都是安装在 D 盘的，操作如下：

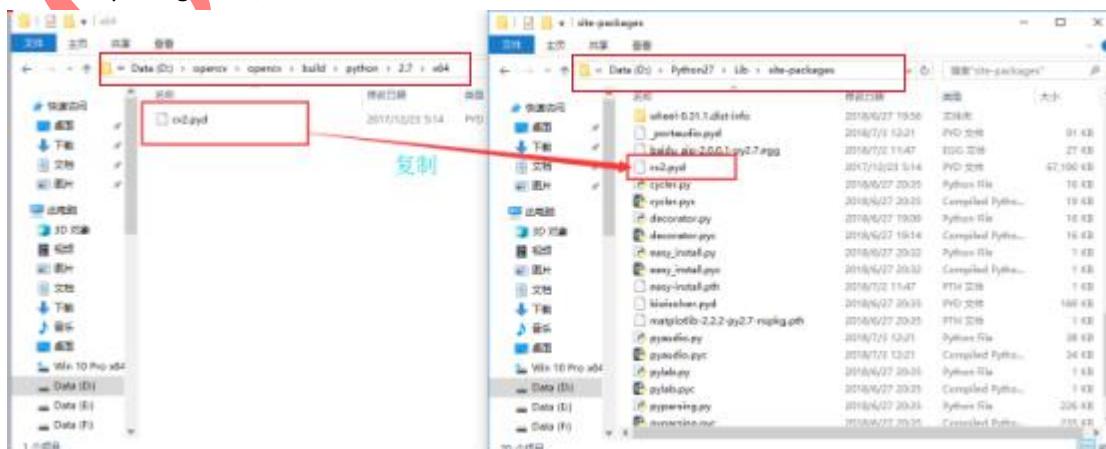


图 1.5 安装 python-opencv

1.2.2 树莓派安装 python-opencv

树莓派自带 python2 和 python3，不需要在安装 python，直接安装 OpenCV 即可，采取简单的方案安装 OpenCV，避免复杂的源码编译。

1、更新树莓派系统

```
sudo apt-get update  
sudo apt-get upgrade
```

2、安装 python—OpenCV

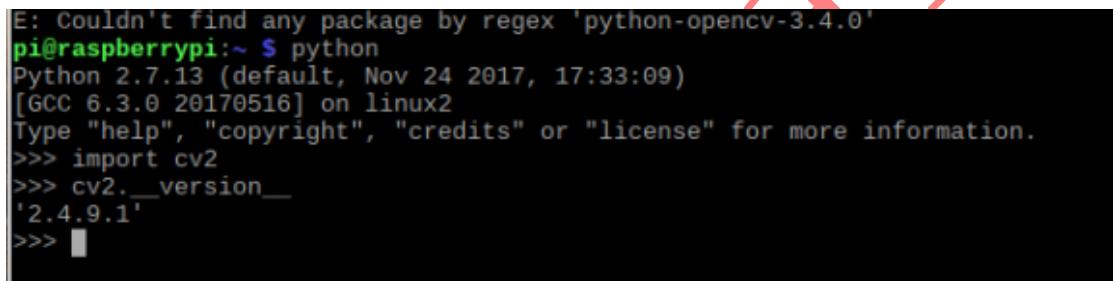
```
sudo apt-get install libopencv-dev  
sudo apt-get install python-opencv
```

该种安装方式不知道安装那个版本的 OpenCV

3、测试 opencv

```
import cv2
```

终端命令行打印 OpenCV 的版本号，本教程资料会增加一个使用 OpenCV 获取 usb 摄像头实时视频流的程序。该种 OpenCV 的安装方式，比较简单和快速，适合刚上手树莓派 OpenCV 的读者。



```
E: Couldn't find any package by regex 'python-opencv-3.4.0'  
pi@raspberrypi:~ $ python  
Python 2.7.13 (default, Nov 24 2017, 17:33:09)  
[GCC 6.3.0 20170516] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import cv2  
>>> cv2.__version__  
'2.4.9.1'  
>>> █
```

图 1.6 python-openCV 安装测试

1.2.3 源码编译方式安装 OpenCV

源码编译安装 OpenCV 比较繁琐，安装之后的 OpenCV 可以采用多种语言进行开发，是一种全面的安装，安装过程费时费力，后续会有专门一节讲述。

1.3 快速上手 OpenCV

本节讲解一些 OpenCV 的基础操作函数，涉及图像的读取、显示、存储、视频的读取和保存、OpenCV 和鼠标的交互、OpenCV 中的画图函数。

1.3.1 图像的读入、显示和保存

1、图像的读入：`img = cv2.imread('tankCar.jpg', 0)`

第一个参数是图片的路径，第二个参数是如何读取这幅图片

`cv2.IMREAD_COLOR`: 读入一副彩色图像。图像的透明度会被忽略，这是默认参数。

`cv2.IMREAD_GRAYSCALE`: 以灰度模式读入图像

2、图像的显示：`cv2.imshow(‘image’, img)`

第一个参数是窗口的名字，窗口会自动调整为图像的大小，第二个参数是显示图像的句柄

但是在程序执行的过程中窗口会一闪而过，需要添加以下语句：

`cv2.waitKey(0)`: 键盘绑定程序，等待键盘输入。

`cv2.destroyAllWindows()`: 删除我们建立的任何窗口

`cv2.destroyWindow(‘image’)`: 删除指定的窗口名

3、图像的保存：`cv2.imwrite(‘car.jpg’, img)`

第一个参数是保存的文件名，第二个参数是待保存的图像

4、图像的读取显示保存综合：

以下程序会加载一幅彩图并显示图片，键盘按下‘s’键保存图像并退出，或者按下‘ESC’键退出但不保存。源程序如下：

```
#-*-coding:UTF-8 -*-
# @author: zd1
'''

图像读取、显示和保存综合实验。
按下 ESC 键退出程序，按下 s 键保存图片
'''

import cv2

img = cv2.imread('tankCar.jpg',cv2.IMREAD_COLOR) #读取图像

cv2.imshow('image',img) #显示图像

k = cv2.waitKey(0) #获取键盘按下的键值
if k == 27:
    cv2.destroyAllWindows()
elif k == ord('s'):
    cv2.imwrite('car.jpg', img)
    print '保存图片成功'
    cv2.destroyAllWindows()
```

1.3.2 视频的读入、显示和保存

本节学习视频的读取，显示和保存。函数为 `cv2.VideoCapture()` 和 `cv2.VideoWriter()`

从摄像头获取视频流。

打开摄像头：`cap = cv2.VideoCapture(0)`

参数 0 表示设备的默认摄像头，当设备有多个摄像头时可以改变参数选择

读取摄像头的视频流：`ret, frame = cap.read()`

无参数，但需放在循环中不断读取形成视频或者定时读取视频

释放摄像头资源：`cap.release()`

无参数，程序关闭之前务必关闭摄像头，释放资源。

读取视频文件

`cap = cv2.VideoCapture('xxfilename')`

打开文件名为 xxfilename 的视频

保存视频：

创建一个 `VideoWriter` 对象，并指定输出文件名，指定视频编码格式

指定编码格式：`fourcc = cv2.VideoWriter_fourcc(*'XVID')`

指定输出文件：

out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))
output.avi 是保存的图像名，最后一个参数为视频的分辨率
视频读取显示保存综合：获取摄像头的视频流并保存在当前文件夹下

```
#-*-coding:UTF-8 -*-
# @author: zd1
'''

视频读取、显示和保存综合实验。
按下 ESC 键退出程序
'''

import cv2
cap = cv2.VideoCapture(0)
#设置视频输出
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))

while (cap.isOpened()):    #摄像头是否正常开启
    ret, frame = cap.read()
    out.write(frame)      # 写视频
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
# 释放资源
cap.release()
out.release()
print '视频保存成功'
cv2.destroyAllWindows()  #关闭窗口
```

1.3.3 绘制直线、矩形、圆

学习使用 openCV 绘制不同的几何图形，可用于后期程序的调试。

相关函数为 cv2.line(), cv2.circle(), cv2.rectangle(), cv2.putText() 等

绘制直线：

cv2.line(img, startPoint, endPoint, color, thickness)

第一个参数为待绘制的图像，startPoint 为直线起点，endPoint 为直线中点，color 为直线的颜色，thickness 为直线的宽度。

绘制圆：

cv2.circle(img, centerPoint, radius, color, thickness)

第一个参数为待绘制的图像，centerPoint 为圆心，radius 为圆的半径，color 为圆的颜色，thickness 为圆弧线的宽度，为负数时表示圆被填充。

绘制矩形：

cv2.rectangle(img, point1, point2, color, thickness)

第一个参数为待绘制的图像，point1 为左上顶点，point2 为 point1 对角线上的另一个顶点，color 为矩形的颜色，thickness 为矩形线的宽度，为负数时表示矩形被填充。

书写文本：

cv2.putText(img, text, point, font, size, color, thickness)

`text` 为要写的文本, `point` 为第一个字符的左下坐标, `font` 为字体类型, `size` 为字体大小, 该函数对中文的支持不够好。

综合: 在一幅图片上画线, 画圆, 画矩形和文字, 程序运行效果如下:

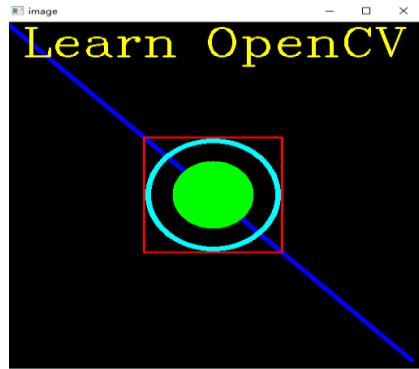


图 1.7 绘制直线圆文字

```
#-*-coding:UTF-8 -*-
```

```
# @author: zd1
```

```
'''
```

使用 OpenCV 函数绘制常用图形

在一幅图片上画线, 画圆, 画矩形和文字。

```
'''
```

```
import cv2
```

```
import numpy as np
```

```
img = np.zeros((512,512,3), dtype=np.uint8) # 创建一幅图像
```

```
cv2.line(img,(0,0),(500,500),(255,0,0),5) # 绘制直线
```

```
cv2.circle(img,(255,255),50,(0,255,0),-1) # 绘制填充圆
```

```
cv2.circle(img,(255,255),80,(255,255,0),5) # 绘制非填充圆
```

```
cv2.rectangle(img,(170,170),(340,340),(0,0,255),2) # 绘制矩形
```

```
# 绘制文本
```

```
cv2.putText(img,'Learn  
OpenCV',(20,50),cv2.FONT_HERSHEY_COMPLEX,2,(0,255,255),2)
```

```
cv2.imshow('image', img) # 显示图像
```

```
# 等待释放窗口
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

1.3.4 openCV 创建滑动条

学习将滑动条绑定到窗口, 实现动态调整某些参数。

学习 `cv2.getTrackbarPos()` 和 `cv2.createTrackbar()` 两个函数。

创建滑动条: `cv2.createTrackbar(name,window,min,max,callback)`

第一个参数是滑动条的名字, 第二个参数是滑动条存在的窗口, 第三和第四个参数是滑动条的取值范围, 第五个是回调函数。

回调函数：回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

获取滑动条的值：cv2.getTrackbarPos(name, window)

第一个参数是滑动条的名字，第二个参数是滑动条存在的窗口

Demo：改变 RGB 值，实现一个调色板。

运行效果：

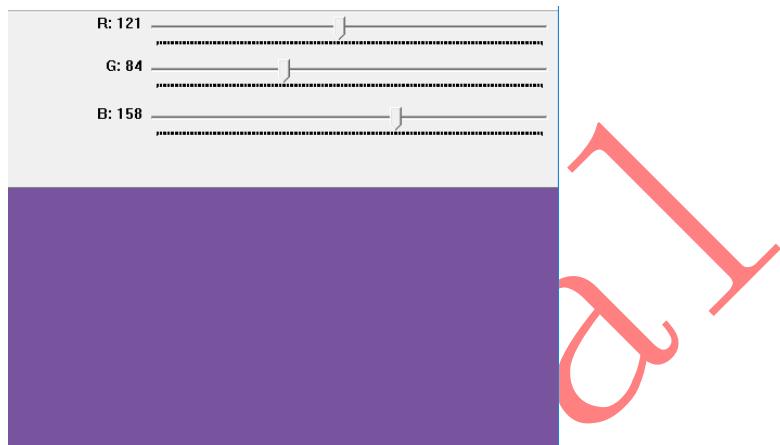


图 1.8 鼠标实现调色板

代码：

```
#-*-coding:UTF-8 -*-
# @author: zd1
'''

创建滑动条实验
获取三个RGB三个滑动条的值，实现调色板功能
按键按下ESC退出程序
'''

import cv2
import numpy as np
# create empty callback
def nothing(x):
    pass # pass 为 python 中的空语句，占位使用无实际意义
# 创建一幅黑色图片
img = np.zeros((320,512,3), dtype=np.uint8)
# 创建一个窗口
cv2.namedWindow('image')
# 创建滑动条
cv2.createTrackbar('R', 'image', 0, 255, nothing)
cv2.createTrackbar('G', 'image', 0, 255, nothing)
cv2.createTrackbar('B', 'image', 0, 255, nothing)

while True:
```

```

cv2.imshow('image', img)

# 获取滑动条的值
r = cv2.getTrackbarPos('R', 'image')
g = cv2.getTrackbarPos('G', 'image')
b = cv2.getTrackbarPos('B', 'image')
# 将 BGR 的值赋给图像矩阵
img[:, :] = [b, g, r]
if cv2.waitKey(1) & 0xFF == 27:
    break
cv2.destroyAllWindows()

```

1.4 OpenCV 的颜色空间

1.4.1 RGB 和 HSV 颜色空间

RGB 颜色空间是大家最熟悉的颜色空间，即三基色空间，任何一种颜色都可以由该三种颜色混合而成。然而一般对颜色空间的图像进行有效处理都是在 HSV 空间进行的，HSV(色调 Hue,饱和度 Saturation,亮度 Value)是根据颜色的直观特性创建的一种颜色空间，也称六角锥体模型。参考：https://blog.csdn.net/taily_duan/article/details/51506776

为什么会选择 HSV 空间而不是 RGB 空间？对于图像而言，识别相应的颜色在 RGB 空间、HSV 空间或者其它颜色空间都是可行的。在 RGB 空间中识别某种具体的颜色需要获取 R、G、B 三个颜色通道的值，并且 RGB 值受周围的环境影响很大。之所以选择 HSV，结合图 1.9 我们可以看出，在 HSV 颜色空间中，锥体的尖端部分代表黑色，其他常见的纯色都分布在锥体的底端圆上面，可以看出通过判断 H 值的范围大小就基本上可以确定某种颜色，再结合饱和度和亮度信息判断大于某一个阈值，就可以很准确的识别颜色。而 RGB 由三个分量构成，需要判断每种分量的贡献比例。即 HSV 空间的识别的范围更广，更方便。

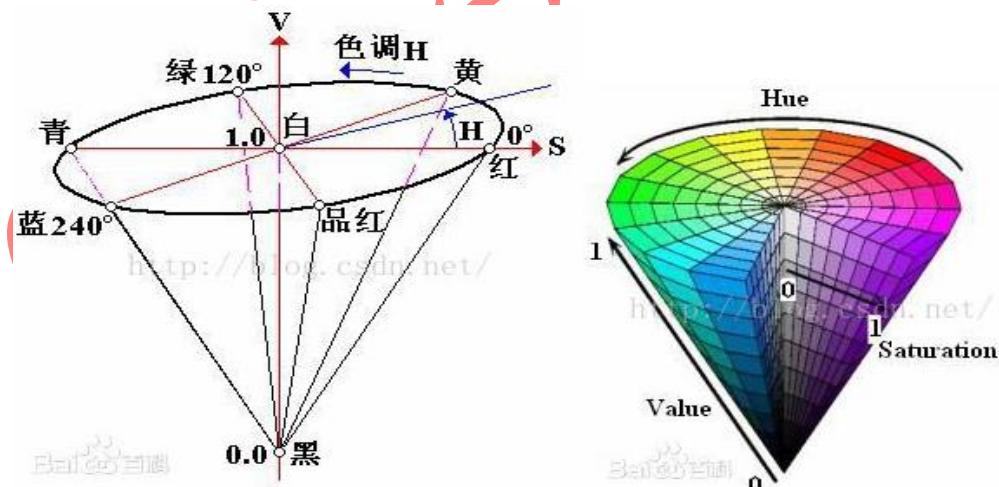


图 1.9 HSV 颜色空间

1.4.2 颜色空间转换和颜色识别

在 openCV 中有超过 150 种的颜色空间转换的方法，但是我们经常会用到的也就只有两种，即 BGR->Gray 和 BGR->HSV。**注意 Gray 和 HSV 不可以互相转换。**

颜色空间转换：`cv2.cvtColor(input_image, flag)`

BGR->Gray：flag 就是 `cv2.COLOR_BGR2GRAY`

BGR->HSV: flag 就是 cv2.COLOR_BGR2HSV

OpenCV 中 HSV 颜色空间的取值范围:

H [0, 179] S [0, 255] V [0, 255]

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34	77	99	124	155
smin	0	0	0	43	43	43	43	43	43	43	43
smax	255	43	30	255	255	255	255	255	255	255	255
vmin	0	46	221	46	46	46	46	46	46	46	46
vmax	46	220	255	255	255	255	255	255	255	255	255

图 1.10 部分颜色 HSV 值表

Demo: 颜色空间转换识别图片中的黄色和红色部分

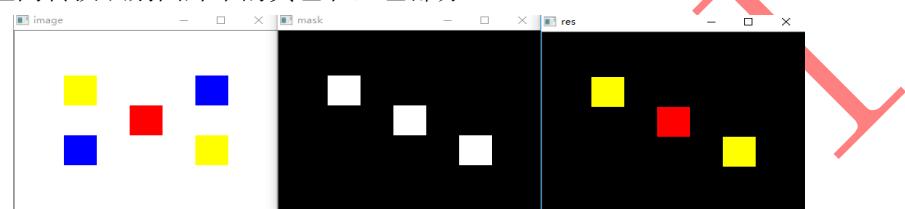


图 1.1 HSV 颜色识别

```
#-*-coding:UTF-8 -*-
# @author: zd1
...
颜色空间变换实验
BGR 颜色空间转换为 HSV 颜色空间，构建掩膜实现颜色识别
实现红色和黄色块的识别
HSV 值的范围 H[0,179] S[0,255] V[0,255]
...
import numpy as np
import cv2
# 创建图片和颜色块
img = np.ones((240,320,3),dtype = np.uint8)*255
img[100:140, 140:180] = [0,0,255]
img[60:100,60:100] = [0,255,255]
img[60:100,220:260] = [255,0,0]
img[140:180,60:100] = [255,0,0]
img[140:180,220:260] = [0,255,255]

# 设定红色的 HSV 值
yellow_lower = np.array([26,43,46])
yellow_upper = np.array([34,255,255])
red_lower = np.array([0,43,46])
red_upper = np.array([10,255,255])
```

```

# 颜色空间变换 BGR—>HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# 构建掩膜
mask_yellow = cv2.inRange(hsv, yellow_lower, yellow_upper)
mask_red = cv2.inRange(hsv, red_lower, red_upper)

# 利用掩膜位运算
mask = cv2.bitwise_or(mask_yellow, mask_red) # 或运算
res = cv2.bitwise_and(img, img, mask=mask)

cv2.imshow('image', img)
cv2.imshow('mask', mask)
cv2.imshow('res', res)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

可自行尝试实现多种颜色的识别。

1.4.3 物体追踪

从上一小节我们知道了如何在 HSV 颜色空间识别特定的颜色了，在 HSV 颜色空间更容易提取和表示一个颜色，利用这一特点，我们可以提取特定颜色的物体，并在摄像头的视野范围内追踪物体，实时打印物体的中心坐标。具体实现的步骤如下：

- 获取视频流
- 颜色空间转换 RGB—>HSV
- 设置 HSV 的阈值
- 识别并追踪物体

Demo 代码如下：

```

#-*-coding:UTF-8 -*-
# @author: zd1
...
颜色追踪实验
BGR 颜色空间转换为 hsv 颜色空间，构建掩膜实现颜色识别
寻找掩膜运算过的图片的轮廓，轮廓面积最大的区域就是要追踪的物体
HSV 值的范围 H[0,179] S[0,255] V[0,255]
按键按下 ESC 退出程序
...
# 导入包
import numpy as np
import cv2

# 设定追踪物体颜色阈值 蓝色
blue_lower = np.array([100,50,50])
blue_upper = np.array([124,255,255])

```

```

cap=cv2.VideoCapture(0)

# 修改摄像头的分辨率 320*240
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

while(cap.isOpened()):
    ret, frame = cap.read()

    frame = cv2.GaussianBlur(frame,(5,5),0) # 高斯滤波
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) # 颜色空间变换
    mask = cv2.inRange(blue_lower,blue_upper) # 根据阈值构建掩模

    # 图像形态学操作，膨胀腐蚀
    mask = cv2.erode(mask,None,iterations=2)
    mask = cv2.dilate(mask,None,iterations=2)
    mask = cv2.GaussianBlur(mask,(3,3),0)

    res = cv2.bitwise_and(frame,frame,mask=mask) # 对源图像进行位操作
    # 寻找轮廓，并绘制轮廓
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                           cv2.CHAIN_APPROX_SIMPLE)[-2]
    if len(cnts) > 0:
        # 寻找面积最大的轮廓并画出其最小外接圆
        cnt = max(cnts, key=cv2.contourArea )
        (x,y), radius = cv2.minEnclosingCircle(cnt)
        cv2.circle(frame,(int(x),int(y)),int(radius),(255,0,255),2)
        print(int(x),int(y)) # 打印追踪物体的中心坐标
    else:
        pass

    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    if cv2.waitKey(5)&0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()

```

执行效果：图像存在噪声，需要添加相应的滤波器进行滤波，程序很准确的圈出了摄像头视野范围内的蓝色小球。

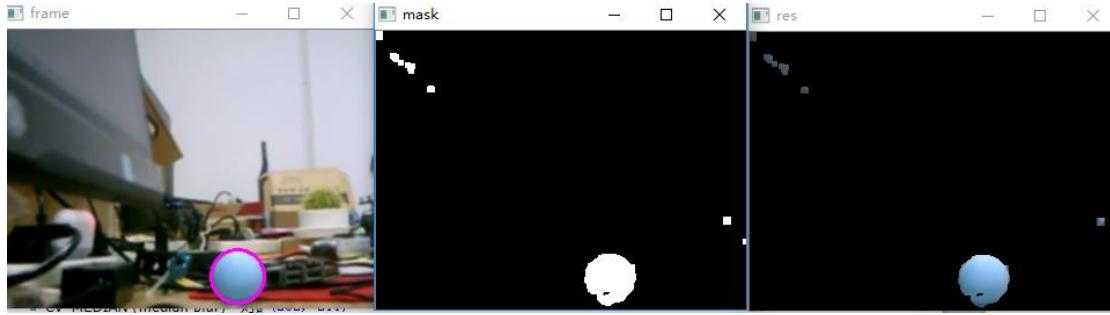


图 1.12 color tracking

1.5 OpenCV 实现颜色识别和物体追踪

1.5.1 颜色识别

上一节中，我们使用颜色空间转换的方法，实现对特定颜色的识别和跟踪，但这种识别和追踪是建立在我们已经知道要追踪物体的颜色基础上。然而当我们不知道所要追踪物体的颜色时，这种方法就不适用了。所以本小节我们讲解采用 openCV 识别物体颜色并采用语音合成技术对识别物体的颜色进行语音输出。

如何采用 OpenCV 识别颜色呢？常用的方案有两个：

- 1、采用颜色直方图，统计 HSV 颜色空间 HS 值的范围
- 2、在 HSV 颜色空间中，对常见颜色的 HSV 值进行遍历统计轮廓。综合对比两个方案，方案一更加简单且高效，所以采用颜色直方图的方法进行颜色识别。

颜色直方图是什么，为什么采用它可以识别颜色呢，下面我来一一解答。直方图（Histogram）是一种对数据的分布进行统计的一种数学方法，是一种二维统计图表，其对应的坐标统计样本和该样本对应的一种属性。例如我们统计一个学校学生人数的情况，统计的样本为每个年级，对应的样本属性就是每个年级的人数。那么将直方图应用到数字图像领域，统计的样本就是图像的像素值，对应的样本属性就是该副图像具有相同像素值的像素点数。所以可以总结一下直方图的作用如下：

- 统计图像中每一个像素值所具有的像素个数
- 直方图是图像中像素强度分布的图像表达方式
- 直方图不包含图像的位置信息

下面介绍绘制直方图的函数 `cv2.calcHist()`

```
cv2.calcHist(images; channels; mask; histSize; ranges[; hist[; accumulate]])
```

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。（后边有例子）
4. `histSize:BIN` 的数目。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]`

下面我们进行直方图统计的实践，绘制灰度图像的一维直方图，demo 代码如下：

```

#-*- encoding:utf-8 -*-
# @author: zd1
# 绘制图像直方图

import cv2
import numpy as np
from scipy.misc import imresize
from matplotlib import pyplot as plt

img = cv2.imread('tankCar.jpg', 0)
img = imresize(img, (240, 320))

hist = cv2.calcHist([img], [0], None, [256], [0, 256]) # 绘制直方图
hist_max = np.where(hist == np.max(hist)) # 获取直方图最大的值及其索引
print hist_max[0]

cv2.imshow('image', img)

# pyplot 绘图
plt.plot(hist)
plt.xlim([0, 256])
plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()

```

绘制的结果如下图，该图表明像素值为 139 的像素点最多，同时也可以说直方图只表达了统计信息，不包含图像像素点的位置信息。

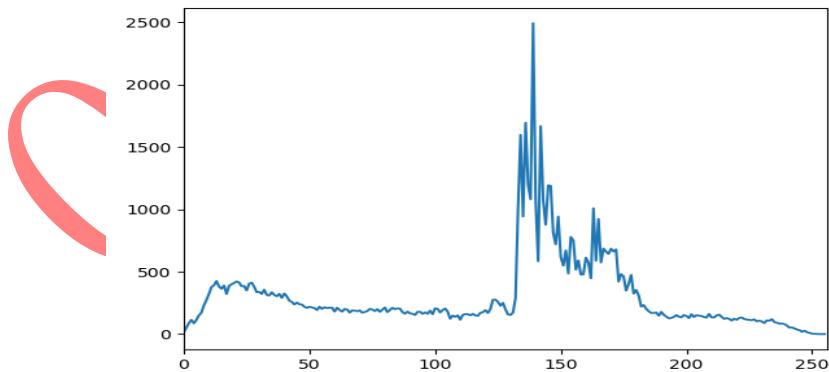


图 1.13 绘制的直方图

由以上的结论，同学们会有疑惑，直方图为什么可以识别颜色？在颜色空间转换一节中，六角锥体模型不知道还有没有印象，在 HSV 颜色空间中可以采用 H(色调)来表示常见的颜色，是不是豁然开朗。我们统计图像中 H(色调)的直方图，结合常见颜色 H 的范围就可以识别颜色了。前面所提的直方图都是灰度直方图（一维直方图），都是统计图片的灰度信息的。OpenCV 中也存在 2D（二维）直方图，即颜色直方图（H-S，色调-饱和度），统计该直方图可以更加准确的识别颜色，下面是 2D 直方图的 demo：

```

#-*- encoding:utf-8 -*-
# @author: zd1
# 绘制 HSV 颜色空间 2D 直方图

import cv2
import numpy as np
from scipy.misc import imresize
from matplotlib import pyplot as plt

img = cv2.imread('tankCar.jpg',cv2.IMREAD_COLOR)
img = imresize(img, (240,320))
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# 生成 2D 直方图 [0,1] 分别代表 H,S 通道
hist = cv2.calcHist([hsv],[0, 1],None,[180, 256],[0, 180, 0, 256])

cv2.imshow('image', img)

# pyplot 绘图
plt.imshow(hist, interpolation = 'nearest')
plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()

```



生成的 2D 直方图

图 1.14 颜色二维直方图

Demo 运行的结果如下：x 轴为 s 值，y 轴为 h 值，在 2D 直方图中可以看出 $H=105, S=230$ 比较亮，表示有较高的值，表示图片中蓝色的区域比较多。通过对 H 和 S 值的判断可以在背景比较单一的场景下识别颜色。

由以上两节的知识讲解，现在来实现物体颜色的识别是不是就很简单了。然而单纯的颜色识别太单调了，给它加点料，加个 TTS (text to speech) 语音合成。语音合成的内容会在后面讲解，这里暂时用一下。这里总结一下颜色识别的原理，不要被一维、二维直方图整迷糊了，由于二维颜色直方图的计算量比较大，且需要 H 和 S 组成联合判断条件比较繁琐，我们知道在 HSV 颜色空间中颜色可以直接用 H 的值直接表示，所以我们采用统计二维颜色直方图中的一维 (H 的值) 直方图来实现颜色识别。下面是对颜色识别的 Demo：

```

#-*- encoding:utf-8 -*-
# @author: zd1
# 统计 HSV 颜色空间 H 的直方图，实现颜色识别

```

```

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

# 生成颜色直方图
def color_hist(img):
    # 构建掩膜
    mask = np.zeros(img.shape[:2], np.uint8)
    mask[70:170, 100:220] = 255
    # 生成 HSV 颜色空间 H 的直方图
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hist_mask = cv2.calcHist([hsv], [0], mask, [180], [0, 180])
    # 统计直方图识别颜色
    object_H = np.where(hist_mask == np.max(hist_mask)) # 获取直方图最大的值及其索引
    print object_H[0]
    return object_H[0]
    # 绘制直方图
    #plt.plot(hist_mask)
    #plt.xlim([0,180])
    #plt.imshow(hist, interpolation = 'nearest')
    #plt.show()

    # 判断直方图 H 的值，实现颜色识别
    # yellow (22,36) red(156,180) blue(100,124), green(35,77)
    # cyan-blue(78,99) orange(6,20)
    # try except 捕获 object_H 存在多个值的异常
    def color_distinguish(object_H):
        try:
            if object_H > 26 and object_H < 34:
                color = 'yellow'
            elif object_H > 156 and object_H < 180:
                color = 'red'
            elif object_H > 100 and object_H < 124:
                color = 'blue'
            elif object_H > 35 and object_H < 77:
                color = 'green'
            elif object_H > 78 and object_H < 99:
                color = 'cyan-blue'
            elif object_H > 6 and object_H < 15:
                color = 'orange'
            else:

```

```

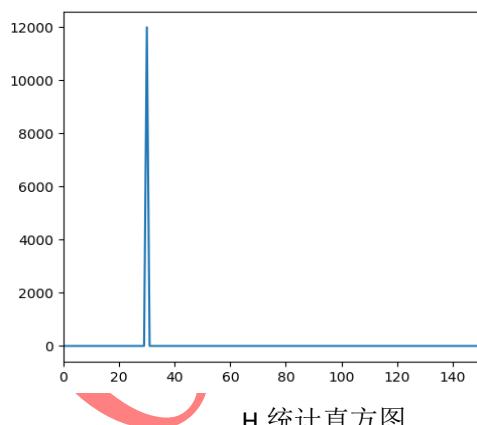
        color = 'None'
print color
return color
except:
    pass

# main 函数入口
if __name__ == '__main__':
    # 构建图片
    img = np.ones((240, 320, 3), dtype = np.uint8) * 128
    img[60:180, 80:240] = [0, 255, 255]
    #颜色识别
    object_H = color_hist(img)
    color_distinguish(object_H)

    cv2.imshow('image', img)
    cv2.waitKey(0)

```

程序执行的结果如下所示：采用 ROI 图像操作，实现对黄色块的识别，将本 Demo 程序简单修改，可以识别实时视频流摄像头前方物体的颜色，结合 TTS，实现颜色识别及语音播报功能。



H统计直方图



识别结果

图 1.15 识别结果

1.5.2 跟随小车

每一位创客应该都会有属于自己的一台小车，即使没有车也会买几个车轮子买两个电机买块亚克力板或者木板自己做个属于自己小车。小车可以玩的东西有很多，例如 APP 遥控、WiFi 视频图传、巡线、跟踪等。自然搭载好的处理器可以进阶高级玩家，玩一下图像处理，自动驾驶，图像分类等。基于本章节的知识学习，我们来学习制作一款属于自己的跟屁虫小车。首先说明的是，该小车是基于摄像头跟随的，无超声波传感器，无红外传感器。具体实现的步骤如下：

- 设定追踪的目标物体（带颜色物体，与周围环境颜色不同）
- 图像处理识别目标物体

- 获取目标物体相对小车的位置信息
- PID 算法调节控制小车运动

在本章节中，我们通过采用颜色空间变换，实现了在摄像头的视野范围内追踪物体，并实时打印物体的中心坐标。跟随小车也是基于这个原理来实现的，获取中心坐标后控制小车的运动实现对物体的跟随。颜色物体追踪的过程在这里我就不再赘述，下面主要讲解如何控制小车跟随的。看下面的图，直观一点。

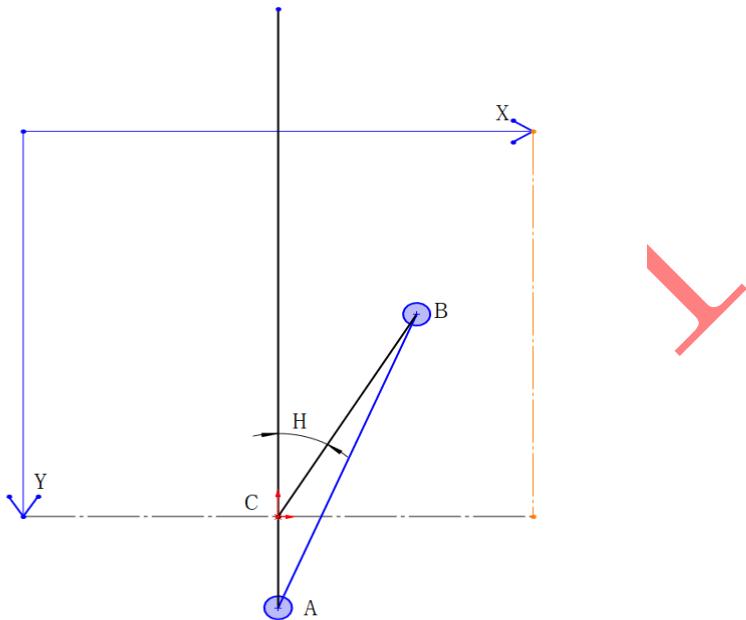


图 1.16 控制方法示意图

上图中 A 点是摄像头的位置，B 点是识别到的物体的位置，其坐标结合物体追踪的结果已知。C 点是图片的最下面一行的中点，即离摄像头最近的那一行的中点位置，其坐标可以表示为 $(Width/2, Height)$ ($Width$ 和 $Height$ 分别是摄像头捕捉图像的宽度和高度)。C 点的坐标是可求的，假设为 (x, y) 。则可以计算 H 的弧度值如下：

$$H = \text{atan}\left(\frac{X - Width/2}{Height - y}\right)$$

既然是追踪目标物体，所以我们要确保目标物体要位于摄像头的正前方，同时可以判断目标物体在图像中的大小来大致判断小车离目标物体的远近。结合弧度 H 的值和目标物体的轮廓大小实现物体追踪，由此我们可以总结出跟随小车的控制策略：

- 转弯：将求解的角度 H 作为控制变量，采用 PID 差速调节小车的转向
- 前进后退：目标物体在图像中的大小判断目标物体与小车的距离关系

我们的小车只有两个电机没有舵机系统，所以无法通过计算角度控制舵机实现准确的转向，这里采用 PID 调节两个电机的转速差实现转向调整。差速转向的原理如下：

我们先选取好一定的参照物，例如选取摄像头的正前方为参考点，当小车检测到偏离方向时，控制电机向偏离的反方向加速，如果不引入 PID 算法进行校正，因为电机的差异或者机械结构的差异小车走直线会有偏差，引入 PID 差速调节会改善这种情况。

PID 算法在另一个文档叙述，采用 python 实现 PID 算法。代码见源程序。

第二章 二维码导航及巡线

2.1 AI 车的运动控制

本节讲解 AI 车的运动控制方案，本小车的电机采用 H 桥控制方案驱动，采用大功率驱动芯片 AM2857 驱动电机，采用 74HC126D 进行隔离。

2.1.1 电机驱动原理讲解

H 桥的电机驱动原理比较简单，现在市场上存在的电机驱动芯片大部分是基于这个原理。下图为 H 桥简化的模型，H 桥存在四个输入，通过控制四个开关组合，从而实现电机的正反转功能。

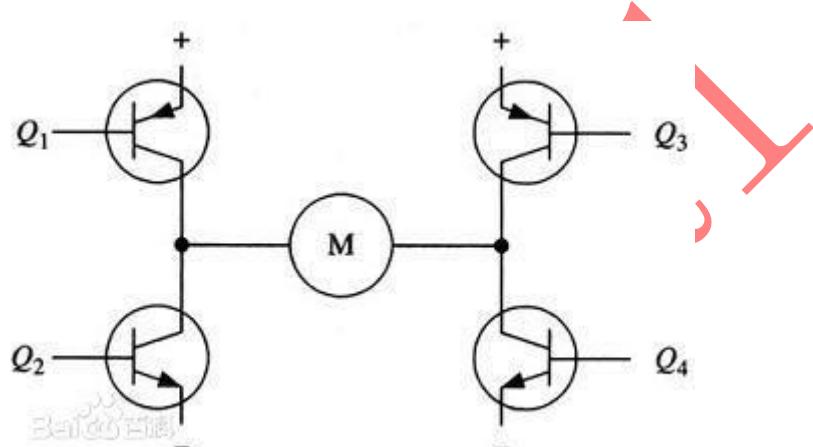


图 2.1 H 桥

当 Q1 和 Q4 导通，Q2 和 Q3 截至时，电机实现顺时针转动。当 Q2 和 Q3 导通，Q1 和 Q4 截至时，电机实现逆时针转动。下图为电机电流的方向：

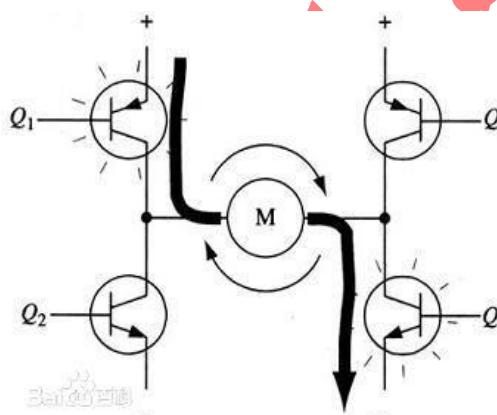


图 2.2 H 桥导通模式一

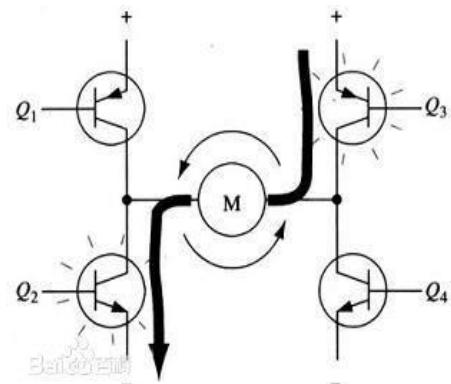


图 2.3 H 桥导通模式二

以上就是 H 桥的工作原理，有条件的同学可以尝试采用两个 NPN，两个 PNP 三极管搭建一个 H 桥电路，驱动电机。

本小车采用集成的电机驱动芯片 AM2857，电流大，功率强。下图是 AM2857 的内部结构图：

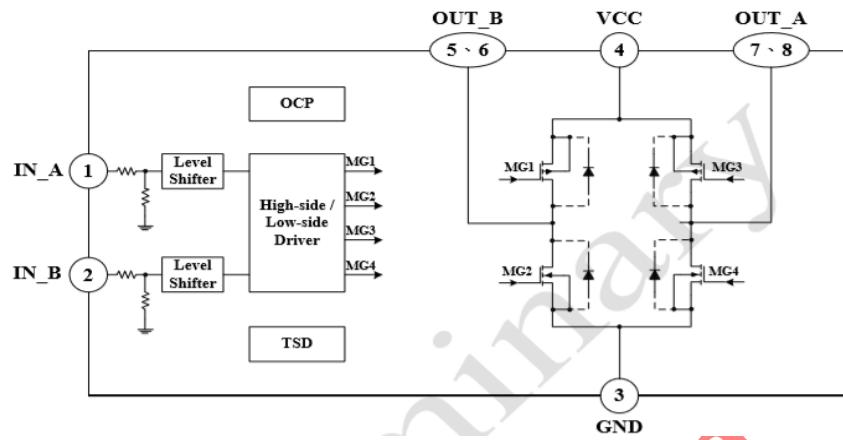


图 2.4 AM2857 结构图

电机控制真值表如下：

IN_A	IN_B	OUT_A	OUT_B	Mode
L	L	Hi-Z	Hi-Z	Stop
L	H	L	H	Reverse
H	L	H	L	Forward
H	H	L	L	Brake

图 2.5 AM2857 控制真值表

2.1.2 综合运动控制

由上一小节学习的电机驱动的知识，我们将理论转化为实践。采用树莓派 3B+和 AM2857 和 python 编程实现 AI 小车的运动控制，实现小车的前后左右旋转运动。下图为 AM2857 驱动电机的原理图，AM2857 的输入分别对应如下：AIN1—IO13 AIN2—IO6 BIN1—IO26 BIN2—IO19。树莓派的引脚编号方式遵循 Python 语言 GPIO 库 RPi.GPIO 约定的编号方式。

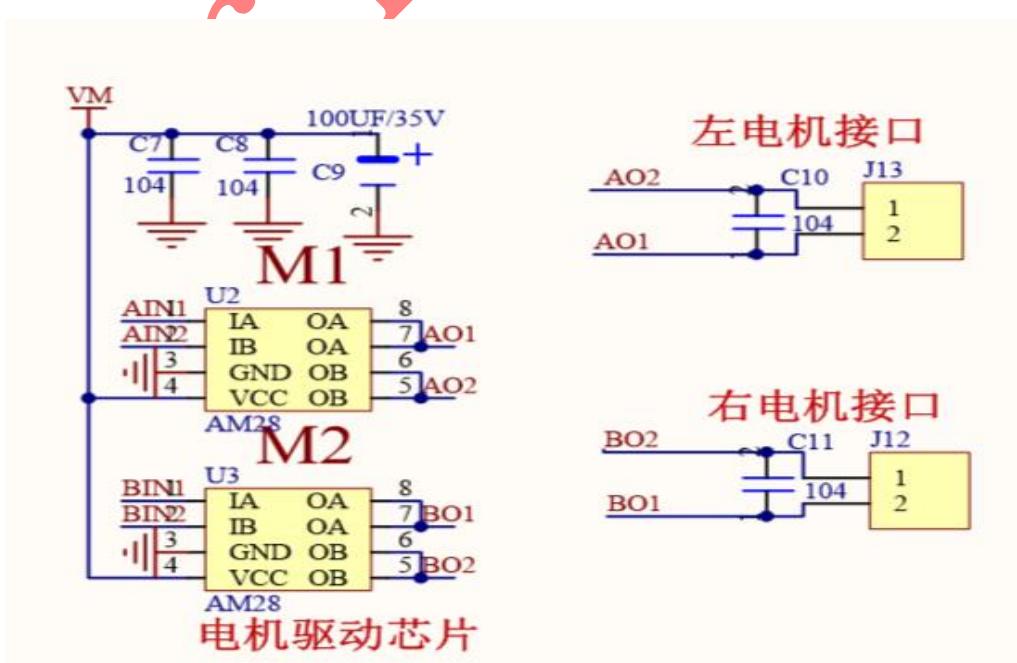


图 2.6 电机驱动 IC AM2857

wiringPi 编码		BCM 编码	功能名	物理引脚 BOARD编码	功能名	BCM 编码	wiringPi 编码
			3.3V	1	2	5V	
8	2	SDA.1		3	4	5V	
9	3	SCL.1		5	6	GND	
7	4	GPIO.7		7	8	TXD	14
		GND		9	10	RXD	15
0	17	GPIO.0		11	12	GPIO.1	18
2	27	GPIO.2		13	14	GND	1
3	22	GPIO.3	3.3V	15	16	GPIO.4	23
12	10	MOSI		17	18	GPIO.5	4
13	9	MISO		19	20	GND	24
14	11	SCLK		21	22	GPIO.6	5
		GND		23	24	CE0	25
				25	26	CE1	6
30	0	SDA.0		27	28	SCL.0	7
21	5	GPIO.21		29	30	GND	10
22	6	GPIO.22		31	32	GPIO.26	24
23	13	GPIO.23		33	34	GND	23
24	19	GPIO.24		35	36	GPIO.27	22
25	26	GPIO.25		37	38	GPIO.28	19
		GND		39	40	GPIO.29	21

图 2.7 树莓派 GPIO 编码对照表

大家请思考，如何通过控制 GPIO 输入来控制电机呢。答案当然是控制 GPIO 的 PWM(Pulse Width Modulation)。采用 python 的 GPIO API 库来实现 PWM 的产生和改变。本车的电机控制逻辑表如下：

AIN1	AIN2	BIN1	BIN2	AI 车
1	0	1	0	前进
0	1	0	1	后退
1	0	0	0	右转
0	0	1	0	左转
0	1	1	0	左旋转
1	0	0	1	右旋转
1	1	1	1	停止
0	0	0	0	停止

表 2-1 电机控制逻辑

程序代码见程序源文件。程序源码讲解。

2.2 二维码识别

本节讲解 AI 车的二维码识别技术，采用 openCV+ZBAR 实现二维码的识别，并提取二维码携带的信息，控制 AI 车的运动实现导航，同时也可以扩展其他功能。

2.2.1 二维码简介

二维码 (2-dimensional bar code) 是用某种特定的几何图形按一定规律在平面 (二维方向上) 分布的黑白相间的图形记录数据符号信息的；在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图象输入设备或光电扫描设备自动识读以实现信息自动处理：它具有条码

技术的一些共性：每种码制有其特定的字符集；每个字符占有一定的宽度；具有一定的校验功能等。同时还具有对不同行的信息自动识别功能、及处理图形旋转变化点。本小车的二维码识别技术主要还是针对二维 QR Code(Quick Response Code)，对 QR 码的信息进行提取和打印。

二维码的制作可以由第三方软件生成，如草料二维码等。

2.2.2 二维码识别技术

现在二维码活跃在我们身边的大街小巷，基本上每天都会和二维码打交道，共享单车啊，移动支付等。那么如何实现二维码的识别呢，其实我们仔细观察二维码的特征的话，可能会看出一些特征，每一个 QR 码，都有三个定位块。识别二维码时会先找到定位块，然后开始读取二维码的编码信息，然后遵循一定的规则对二维码的编码信息进行解码（翻译），解码出的内容就是我们想要的内容了，可以跳转网站，查看文字信息等。二维码识别的原理基于以上的过程，当然现在也有很多开源的二维码识别工具。作为一名创客，我们应该学会利用现有的工具去 make，有人会说这不是五柳先生的不求甚解吗，其实不然，当今时代的知识更新很快，我们很难系统的学习，作为一个 maker，自然享受的是创作作品和实现作品的过程，我们利用现有的软件开发包加快自己的开发，快速的把自己的想法实现，岂不是更爽。下面我们来介绍两个用于二维码识别的开源软件，zbar 和 zxing。

ZBar 是一个开源软件套件，用于从各种来源读取条形码，例如视频流，图像文件和原始强度传感器。它支持许多流行的符号（条形码类型），包括 EAN-13 / UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2 of 5 和 QR Code。灵活的分层实现方便了任何应用程序的条形码扫描和解码，可轻松地将条形码扫描小部件集成到您的 Qt, GTK + 或 PyGTK GUI 应用程序中，具备 python,C++ 等多种语言 API 接口。广泛应用于零售、自动文档处理、库存跟踪、移动应用等领域。ZBar 的特性如下：

- 跨平台 - Linux / Unix, Windows, iPhone®, 嵌入式
- 高速 - 从视频流中实时扫描
- 内存占用少
- 小代码大小 - 核心扫描器和 EAN 解码器代表 1K 行的 C 代码
- 没有浮点运算
- 适用于使用廉价处理器/硬件的嵌入式应用
- 模块化组件可以一起使用或单独使用

本小车采用的开源二维码识别软件是 ZBar，下面将详细介绍 ZBar 如何安装与使用。这里先提一下，ZBar 支持中文识别很奇怪，有的能识别有的不可以识别，笔者扒了好几天文档也没有找到完全之策，待改善。下面介绍 ZBar 的安装：

开发环境树莓派 3Bplus+python2.7.13

安装方法如下：

方案一：

命令行安装：sudo apt-get install python-zbar

直接安装，识别英文一点问题都没有，识别中文不支持。

```
pi@raspberrypi:~ $ sudo apt-get install python-zbar
Reading package lists... Done
```

测试安装成功，进入 python 环境：import zbar

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import zbar
>>> 
```

方案二：源码安装，可识别中文

1、wget 命令用来从指定的 URL 下载 zbar 源码，命令如下：

```
wget http://downloads.sourceforge.net/project/zbar/zbar/0.10/zbar-0.10.tar.gz
```

下载不成功的话，先用电脑下载 zbar-0.10.tar.gz，上传到树莓派的 pi 目录下

2、解压 zbar 的源文件，解压至 pi 目录下的 zbar-0.10 文件夹

```
tar -zvxf zbar-0.10.tar.gz
```

3、zbar 是基于 C 语言开发的，安装编译时需要编译器，安装 python-gtk 和 libqt4-dev

```
sudo apt-get install python-gtk2-dev
```

4、安装 libqt4-dev

```
sudo apt-get install libqt4-dev
```

5、进入 zbar-0.10 文件夹，运行 configure 命令如下，此步运行成功生成 makefile

```
./configure --without-imagemagick -disable-video -without-qt
-without-gtk -without-x
```

```
pi@raspberrypi:~ $ cd zbar-0.10
pi@raspberrypi:~/zbar-0.10 $ ./configure --without-imagemagick -disable-video -w
ithout-qt -without-gtk -without-x
```

6、编译 makefile : make

```
pi@raspberrypi:~/zbar-0.10 $ make
```

7、make 安装： sudo make install

```
pi@raspberrypi:~/zbar-0.10 $ sudo make install
```

8、测试进入 python 环境：import zbar

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import zbar
>>> 
```

注意：第 5 步安装时要进入 /home/pi/zbar-0.10 目录，第 6、7 步安装时可能会出现错误，但是并不影响使用。

ZBar 是日本人写的，若要识别带有中文文字的二维码，需要在安装的时候加个小插曲，需要修改一下 ZBar 源文件里面的函数：修改源文件下 zbar-0.10/zbar/qrcode/qrdectxt.c

终端使用 nano 或者 vim 编辑器修改 zbar-0.10/zbar/qrcode/qrdectxt.c 函数，以下该步骤需要在下载完源码后进行，修改完成后再进行编译和安装。

1、打开 qrdectxt.c

```
pi@raspberrypi:~ $ cd zbar-0.10
pi@raspberrypi:~/zbar-0.10 $ cd zbar
pi@raspberrypi:~/zbar-0.10/zbar $ cd qrcode
pi@raspberrypi:~/zbar-0.10/zbar/qrcode $ nano qrdectxt.c
```

2、修改编码，将 ISO8859-1，换成 GB18030 将 SJIS，换成 GB2312

```
/*This is the encoding the standard says is the default.*/
latin1_cd=iconv_open("UTF-8","GB18030");
/*But this one is often used, as well.*/
sjis_cd=iconv_open("UTF-8","GB2312");
/*This is a trivial conversion just to check validity without extra code.*/
utf8_cd=iconv_open("UTF-8","UTF-8");
```

3、修改解码编码顺序

```
eci_t,
enc_list[0]=latin1_cd;
enc_list[1]=sjis_cd;
enc_list[2]=utf8_cd;
eci_cd=(iconv_t)-1;
```

4、编译源码，安装 ZBar

按照以上的步骤安装基本不会出现问题，如果出现问题仔细检查是否缺少相应的依赖安装即可。下面我们使用 ZBar 实现二维码的识别，ZBar 可以直接对输入的图像提取二维码、扫码和解码，但是为提高扫码和解码的效率，我们先使用图像处理算法实现对二维码的定位并提取二维码的图像，将该部分的图像送入 ZBar 的扫码器中。如何实现对二维码的定位呢，方法如下：

- 1、基于梯度算子，水平垂直交叉定位
- 2、角点检测，仿射变换法，可校正倾斜的二维码
- 3、轮廓法，寻找最小外接矩形。
- 4、轮廓层析结构法，寻找二维码的三个定位点，可校正二维码。

我们采用第三种方法实现静态二维码的提取，实现二维码的识别。具体实现的步骤如下：

- 1、输入图像进行灰度化，滤波
- 2、图像进行 Otsu 自适应二值化。
- 3、二值化的图像进行图像形态学操作，实现细小连通区域的闭合。
- 4、图像位运算进行反色，这时二维码区域为白色。
- 5、寻找反色图像中的轮廓，二维码区域为轮廓中面积最大的那个，实现对二维码区域的定位。
- 6、将定位的二维码图像取出，输入至 zbar 进行扫码和解码。

源代码如下：

```
# -*- coding: UTF-8 -*-
# Simple QR code detection
# @author: zd1
# ZBar+python2.7.13

import cv2
import numpy as np
import zbar

# 基于轮廓定位 QR 位置
def find_code(img):
    # 高斯滤波
    blur = cv2.GaussianBlur(img,(5,5),0)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```

# 二值化
ret,th =
cv2.threshold(gray,0,255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# 形态学操作
mask=cv2.erode(th,None,iterations=4)
mask=cv2.dilate(mask,None,iterations=4)

# 图像反色
cv2.bitwise_not(mask, mask)

# 寻找轮廓
_,contours,hierarchy = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

# 寻找 QR 码的位置
if len(contours)>0:
    cnt = max(contours, key=cv2.contourArea)
    x,y,w,h = cv2.boundingRect(cnt)
    img =
cv2.rectangle(img,(x-15,y-15),(x+w+15,y+h+15),(0,255,0),3)

    img_ROI = img[y-15:y+h+15, x-15:x+w+15]

else:
    img_ROI = img

return img_ROI

# main 函数
if __name__ == '__main__':
    # 读入图片
    img = cv2.imread('introduce.jpg', 1)
    # 定位 QR 码的位置
    img_ROI = find_code(img)

    # 初始化 scanner
    scanner = zbar.ImageScanner()
    scanner.parse_config('enable')

    font = cv2.FONT_HERSHEY_SIMPLEX # openCV 字体

    img_ROI_gray = cv2.cvtColor(img_ROI, cv2.COLOR_BGR2GRAY)
    # 扫码
    width, height = img_ROI_gray.shape # 获取图片大小
    raw = img_ROI_gray.tobytes()          # 图像矩阵数据转为字节数据
    zarimage = zbar.Image(width, height, 'Y800', raw) # 设置参数

```

```

scanner.scan(zarimage)    # 扫码
# 解码
for symbol in zarimage:
    # do something useful with results
    if not symbol.count:
        print 'decoded', symbol.type, 'symbol', '"%s"' % symbol.data
    else:
        print 'no'
#cv2.putText(img,symbol.data,(20,100),font,1,(0,255,0),4) # 打印字符在图片上
# DisPlay
cv2.imshow('img_ROI', img_ROI_gray)
cv2.imshow('image', img)
cv2.waitKey(0)

```

运行效果：

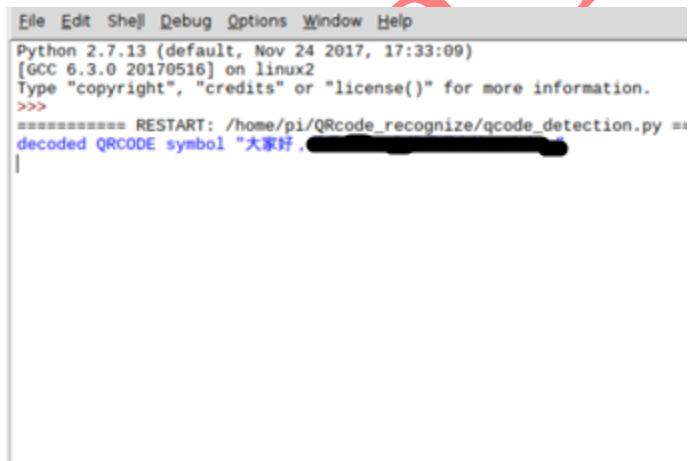


图 2.9 二维码识别结果

基于本节的学习，我们已经实现了二维码的识别，那么结合语音合成方面的知识，是不是可以做点好玩的事情呢。结合 AI 车的运动控制，是不是可以做二维码导航小车呢？

本节讲述的是对照片的二维码识别，下一节讲解实时二维码检测，笔者在实践的过程中发现以上算法比较耗时，所以下面采用了 python 的 pil 图像处理库来实现一个“快速”的二维码识别并控制 AI 车的运动。

2.2.3 基于二维码信息识别的导航小车

传统的智能小车采用远程遥控或者红外传感器或者超声波实现导航或者避障，那我们来点不一样的玩法，使用二维码识别完成对小车的控制或者导航，make 一个简单的 agv 小车。基于二维码信息识别的导航小车主要由两部分构成，一部分是二维码命令识别，另一部分是二维码命令解析。上一节，我们已经实现二维码的识别，这里会简单修改一下，然后再写一个命令解析的代码即可。我们首先需要定义好控制小车的指令，这里简单的定义一下控制指令：

二维码信息	AI 车运动控制	备注
AI car Run	前进	识别 Run 即可
AI car Reverse	后退	识别 Reverse 即可
AI car Left	左转	识别 Left 即可
AI car Right	右转	识别 Right 即可
AI car Stop	停止	识别 Stop 即可

备注：可添加侧方停车、倒车入库等其他功能

表 2-2 控制指令表

程序见源码。

2.3 基于 openCV 的巡线小车

2.3.1 单线巡线

现在市场上大多数的循迹小车，都是基于巡线传感器—红外传感器来实现的，成本低廉、控制简单但准确性差限制了小车的速度，作为一个进阶的创客玩家，是时候入手一款摄像头了，make 一款图像处理小车。有了摄像头，有了性能还可以的处理器，当然要实现一下 auto_driving 了，让我们从简单的单线巡线开始吧。

单线巡线是如何实现的呢？下面我们来分析一下：我们将摄像头拍到的视频分为三部分，单线的左侧、单线、单线的右侧。一般来说，单线的两侧背景一致且与单线区别较大。有了这个背景之后，解决方案就出来了。下面将介绍几种处理方法：

1、单线拟合法，图像的特征很明显，采用图像二值化，将单线映射为黑色 0，单线的两侧为白色 1，遍历图像数组，提取黑色中线的坐标位置，然后采用最小二乘法拟合黑色中线的点集，根据拟合直线斜率的偏差控制小车的转向。这种方法难点在于黑色中线的提取。

2、加权平均法，采用图像 ROI(感兴趣区域图像操作)，将摄像头采集的图像分成上中下三个部分，提取黑线的位置，然后加权平均出三个 ROI 的中点位置坐标，利用该坐标控制小车的转向。

3、检测直线法，采用霍夫直线检测，计算当前图像中黑线的特征，控制小车的转向。

本小车采用第二种方法，加权平均法来实现小车的巡线功能。识别算法原理如下图所示：

选取上中下三个 ROI 域，提取黑线的位置，并计算黑色块的中心的坐标，分配不同的权重拟合黑线的中心点坐标，权重分配遵循越靠近摄像头的部分权重优先，远离摄像头的部分权重分配弱。获取中心点坐标后我们就可以采取追踪小球的算法，如图所示。

如何获取黑色色块呢？这个有两种解决方案：第一种是对图像进行二值化，即除了黑线的部分都是白色，然后获取 ROI 域面积最大的轮廓，画出其最小外界矩形，获取黑色块的中心坐标。第二种方案是采用 HSV 颜色空间转换，识别特定的颜色，采用这种方案可以识别多种颜色的线，提取中心坐标的方法同前者。程序源码见源文件。

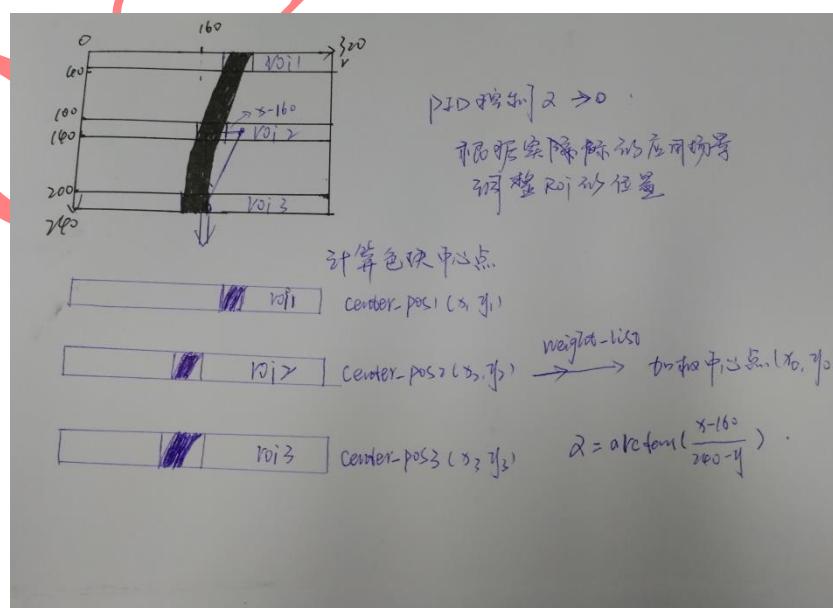


图 2.10 黑线提取方法

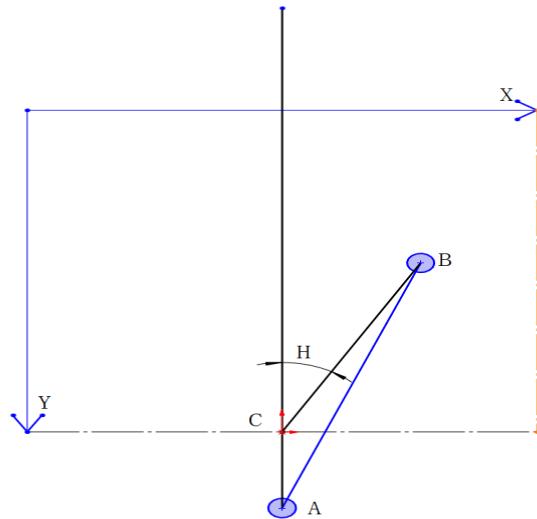


图 2.11 运动控制策略

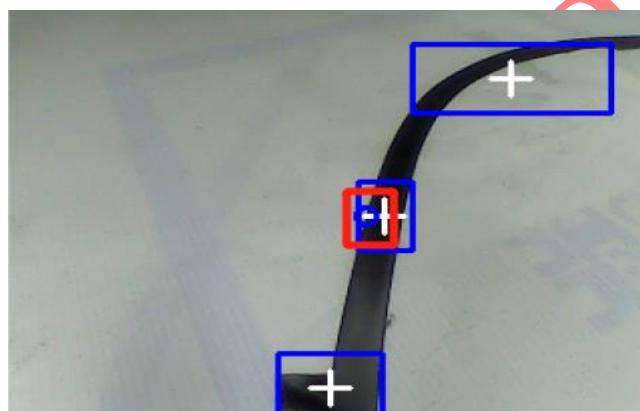


图 2.12 实际识别效果

2.3.2 双线巡线

上节课，我们完成了单线巡线小车的制作，那么这一节升级一下，实现双线巡线小车的制作，本节主要讲解双线巡线的原理。

双线巡线的终极目标是拟合出正确且精确的中线，利用中线的偏差控制小车舵机的转向。所以整个双线巡线的目标是围绕双线的提取来做的，首先双线的特征很明显，和周围的背景差异很大，一般采用图像二值化算法，提取黑白跳变点，一般为了提高程序的效率会从图像的中间向两边扫描或者从中 $1/4$ 、 $3/4$ 处扫描行或者列快速获取左右黑线的坐标，然后为了确保提取黑线的准确性需要对左右黑线数据进行滤波调整。获取了左右黑线数据后，基本上左右黑线的数据不相等，这个时候统一计算中线坐标会出错，所以我们需要计算黑线的截至行，选取左右黑线数据较小的一组数据的大小为黑线截止行，然后开始利用计算式：

$$\text{Mid_line} = (\text{Right_line} - \text{Left_line}) / 2 + \text{Left_line}$$

计算出中线数据后，计算中线的平均值。我们采用三点拟合直线的方法，快速拟合直线，采用拟合直线的偏差控制舵机的转向，利用直线的偏差对赛道类型进行分类，实现速度控制。总结如下：

- 1、左右黑线的提取
- 2、黑线截止行的选取，计算中线
- 3、选取中线的中点，起点和终点拟合中线

4、中线偏差控制舵机

5、中线偏差实现赛道分类，实现动态速度控制

基于 AI 小车的双线巡线方案，我们可以结合 openCV 的图像处理算法来实现，主要是对两边黑线的提取，我们可以采用类似单线循迹的方案，方法如下：

- 1、选取上中下三个 ROI 域
- 2、提取 ROI 域的左右两边的色块
- 3、根据色块的中点对色块进行左右分类
- 4、三个 ROI 域的中点坐标
- 5、三个 ROI 域中点坐标拟合直线
- 6、偏差控制差速转向

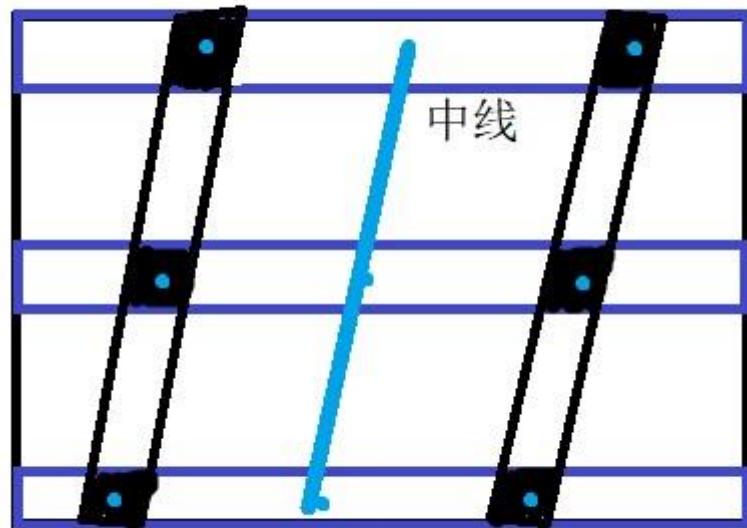


图 2.13 双线提取方法

OK

第三章 语音识别及语音合成

解决 usb 录音采样频率 16K 报错的问题

<https://blog.csdn.net/u013860985/article/details/79326379>

3.1 语音识别功能介绍

人工智能时代的带来，人们的生活方式也开始发生改变，各种智能设备接入千家万户，智能家居、物联网（IOT）。实现智能设备自然就少不了友好的人机交互方式，语音控制和语音合成是目前最友好的人机交互方式。AI 小车具备一定的图像处理能力，当然也少不了语音交互的功能。我们采用百度云语音识别和语音合成技术及 snowBoy 离线唤醒引擎，实现属于我们自己的语音控制小车。本章节需要的硬件工具为 usb 麦克风、3.5mm 音频输出，软件工具为语音识别 Python-SDK、mplayer、pyaudio、snowboy、格式工厂等

3.1.1 语音识别和语音合成简介

我觉得大家可能比较关心语音识别和语音合成能够做什么有趣好玩的功能。我在这里简单的介绍两个有趣且有学习意义的开源项目：

1、语音识别接入图灵机器人实现聊天机器人通过图灵机器人开放平台，用户可快速构建自己的专属聊天机器人并为其添加丰富的机器人云端技能

可参考项目：https://www.urlteam.org/2016/05/python_语音智能对话聊天机器人%ef%bc%8clinux 树莓派双平台兼容/

2、结合语音识别和语音合成技术搭建属于自己的智能音箱，可是实现智能唤醒，智能点歌，智能推送天气的功能，想想都很丰富。

可参考项目：叮当：一个开源的树莓派中文智能音箱项目

那么我们来切入本节的正题吧，讲讲语音合成和语音识别吧。想要了解历史的可以出门左转百度一下，这里就不再赘述了。**语音识别**，机器的听觉系统，机器（PC, MCU 等）通过识别和理解过程把语音信号转变为相应的文本或者命令。语音识别技术主要包括三个阶段，录音特征提取技术，模式匹配准则和模式训练技术。人工智能的到来，利用机器学习对足够的数据进行训练，可以快速搭建语音识别系统且提高语音识别的准确率。语音识别的硬件解决方案只能识别固定的字条或者定制词条，所以本方案采用云语音识别的方案。目前采用免费的百度云语音识别方案。**语音合成（TTS）技术**，是将文本或者控制指令进行合成并输出语音的技术，实现人机交互的重要方法，其包含语言处理，语言理解，声学处理等三部分内容，难点是自然语言处理，尤其对博大精深的中文语音更加难以理解。笔者认为，作为一个创客，我们应该学会利用开源的工具，借助免费语音识别和语音合成引擎搭建属于自己的语音交互系统。下面主要讲解如何利用百度云语音 SDK 进行语音识别和语音合成的开发。

3.1.2 树莓派相关环境配置

我们已经知道如何采用免费的第三方平台搭建语音识别和语音合成的软件平台了，下面我们需要搭建一个声音采集和输出的硬件平台。硬件设备包括一个 usb 麦克风和一个 3.5mm 音频输出，麦克风最好采用免驱和树莓派兼容的 usb 设备。音频输出直接采用树莓派的 3.5mm 音频输出，树莓派的 3.5mm 接口采用的是美国标准的输出，如何测试没有输出你需要看一下你的耳机或者音箱接口是不是一致的，不一致的可以在某宝上购买转接头。我们采用 pyaudio 录音，下面配置一下树莓派。

树莓派下安装 pyaudio 与使用

pyaudio 是 python 的模块，在树莓派下安装 pyaudio 首先需安装 portaudio.dev

- 1、安装 portaudio.dev : sudo apt-get install portaudio.dev
 - 2、安装 python-pyaudio: sudo apt-get install python-pyaudio
 - 3、安装 sox 快速检测麦克风配置是否正确: sudo apt-get install sox
 - 4、测试麦克风配置是否正确，树莓派终端输入以下命令:
- ```
rec temp.wav
```

```
pi@raspberrypi:~ $ rec temp.wav
Input File : 'default' (alsa)
Channels : 2
Sample Rate : 48000
Precision : 16-bit
Sample Encoding: 16-bit Signed Integer PCM
In:0.00% 00:00:00.51 [00:00:00.00] Out:20.5k [|] Clip:0
```

4、测试 pyaudio 代码如下：录音 40s 并保存为 audio.wav 播放。

```
#_*_ coding=UTF-8 _*_
@author: zd1
测试 pyaudio 使用 pyaudio 录音，录音完毕播放录音内容
需要安装 pyaudio 安装过程在教程中讲解
pyaudio API 函数库参考:
http://people.csail.mit.edu/hubert/pyaudio/docs/#pyaudio.Stream.write

import wave
from pyaudio import PyAudio,paInt16

设置采样参数
NUM_SAMPLES = 2000
TIME = 2
chunk = 1024

read wav file from filename
def read_wave_file(filename):
 fp = wave.open(filename,'rb')
 nf = fp.getnframes() #获取采样点数量
 print('sampwidth:',fp.getsampwidth())
 print('framerate:',fp.getframerate())
 print('channels:',fp.getnchannels())
 f_len = nf*2
 audio_data = fp.readframes(nf)

save wav file to filename
def save_wave_file(filename,data):

 wf = wave.open(filename,'wb')
 wf.setnchannels(1) # set channels 1 or 2
```

```

wf.setsampwidth(2) # set sampwidth 1 or 2
wf.setframerate(16000) # set framerate 8K or 16K
wf.writeframes(b"".join(data)) # write data
wf.close()

#recode audio to audio.wav
def record():
 pa = PyAudio() # 实例化 pyaudio

 # 打开输入流并设置音频采样参数 1 channel 16K framerate
 stream = pa.open(format = paInt16,
 channels=1,
 rate=16000,
 input=True,
 frames_per_buffer=NUM_SAMPLES)

 audioBuffer = [] # 录音缓存数组
 count = 0

 # 录制 40s 语音
 while count<TIME*20:
 string_audio_data = stream.read(NUM_SAMPLES) #一次性录音采样字节的
 大小
 audioBuffer.append(string_audio_data)
 count +=1
 print('. ', #加逗号不换行输出

 # 保存录制的语音文件到 audio.wav 中并关闭流
 save_wave_file('audio.wav',audioBuffer)
 stream.close()

 # 播放后缀为 wav 的音频文件
 def play():

 wf = wave.open(r"audio.wav",'rb') # 打开 audio.wav
 p = PyAudio() # 实例化 pyaudio

 # 打开流
 stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
 channels=wf.getnchannels(),
 rate=wf.getframerate(),
 output=True)

 # 播放音频
 while True:

```



```

data = wf.readframes(chunk)
if data == "":break
stream.write(data)

释放 IO
stream.stop_stream()
stream.close()
p.terminate()

main 函数 录制 40s 音频并播放
if __name__ == '__main__':
 print('record ready...')
 record()
 print('record over!')
 play()
程序演示结果:
=====
RESTART: /home/pi/pyaudio_record.py =====
record ready...
.
.
.
record over!
play...
>>> |

```

使用不同的 usb 摄像头会出现采样率报错的问题，解决方案参考以下博客：

<https://blog.csdn.net/u013860985/article/details/79326379>

<https://blog.csdn.net/u013372900/article/details/80296125>

添加一个新的~/.asoundrc 到 pi 目录下: sudo nano ~/.asoundrc

输入以下内容:

```

pcm.!default {
 type hw
 card 1
}
ctl.!default {
 type hw
 card 1
}

```

或者:

```

pcm.!default {
 type asym
 playback.pcm {
 type plug
 slave.pcm "hw:0,0"
 }
 capture.pcm {
 type plug
 }
}

```

```

 slave.pcm "hw:1,0"
 }
}

ctl.!default {
 type hw
 card 1
}

```

```

GNU nano 2.7.4 File: /home/pi/.asoundrc

pcm.!default {
 type asym
 playback.pcm{
 type plug
 slave.pcm "hw:0,0"
 }
 capture.pcm {
 type plug
 slave.pcm "hw:1,0"
 }
}

```

图 3.1 录音结果

**思考：如何采用 pyaudio 制作声音的监控装置呢？监控噪声、人或者物体运动然后提示报警？**

树莓派音频输出设置：

- 1、选择树莓派 audio output 为 AUTO 或者 3.5mm sudo raspi-config 设置
- 2、如何调整输出音量，终端输入 alsamixer 命令然后上下键调整



图 3.2 树莓派音量调整

3、语音合成的音频为 MP3 文件，pyaudio 只能播放 wav 文件，所以需要安装 MP3 播放插件 mplayer。

安装 sudo apt-get install mplayer

测试 mplayer xx.mp3 （需先进入 xx.mp3 的文件夹下）

Python 程序播放 MP3: os.system('mplayer %s' % 'xx.mp3')

```

>>> import os
>>> os.system('mplayer %s' % 'forward.mp3')

```

## 3.2 百度云语音识别 Python-SDK 使用

### 3.2.1 python-SDK 的安装

工欲善其事必先利其器，下面先来武装一下自己，申请属于自己的百度 AI 开发者账号，安装 SDK（Software Development Kit，软件开发工具包）。

1、进入百度云登录平台，申请注册账号登录，已有账号直接登录即可，注册或者登录成功出现如下页面，点击[百度语音](#)链接。

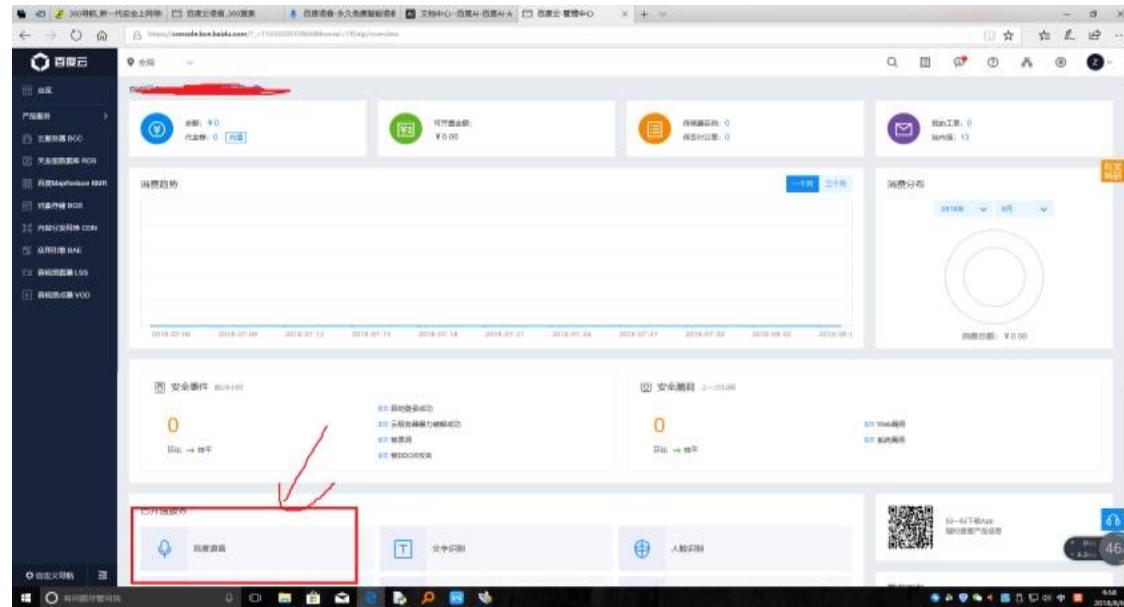


图 3.3 百度语音控制台

2、从上一个页面进入百度语音控制台后，点击[创建应用](#)：

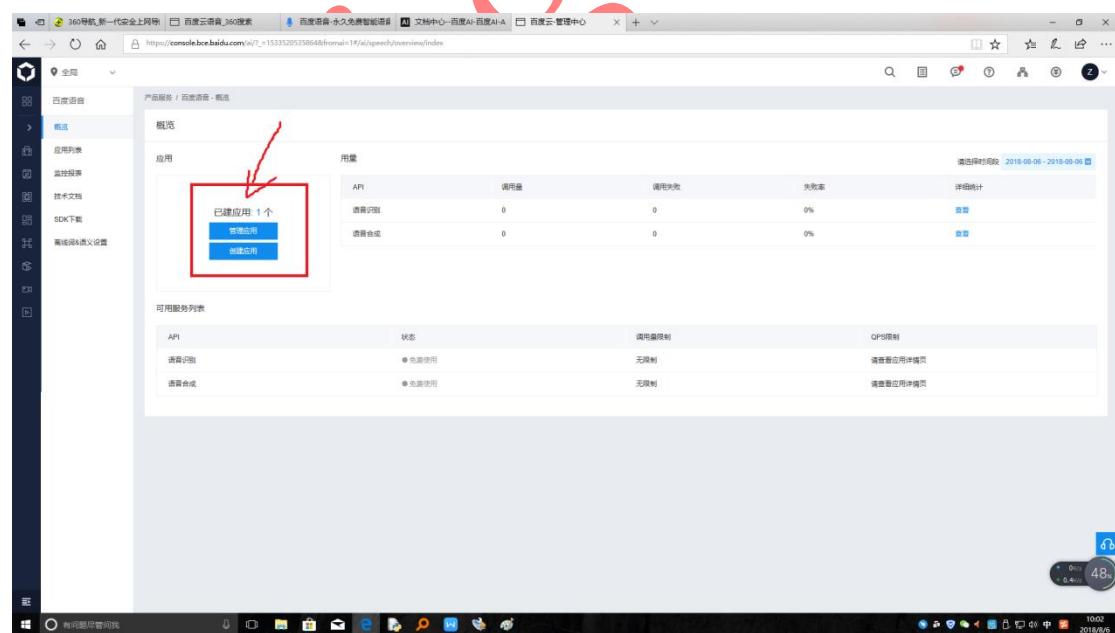


图 3.4 控制台创建应用

3、进入应用创建

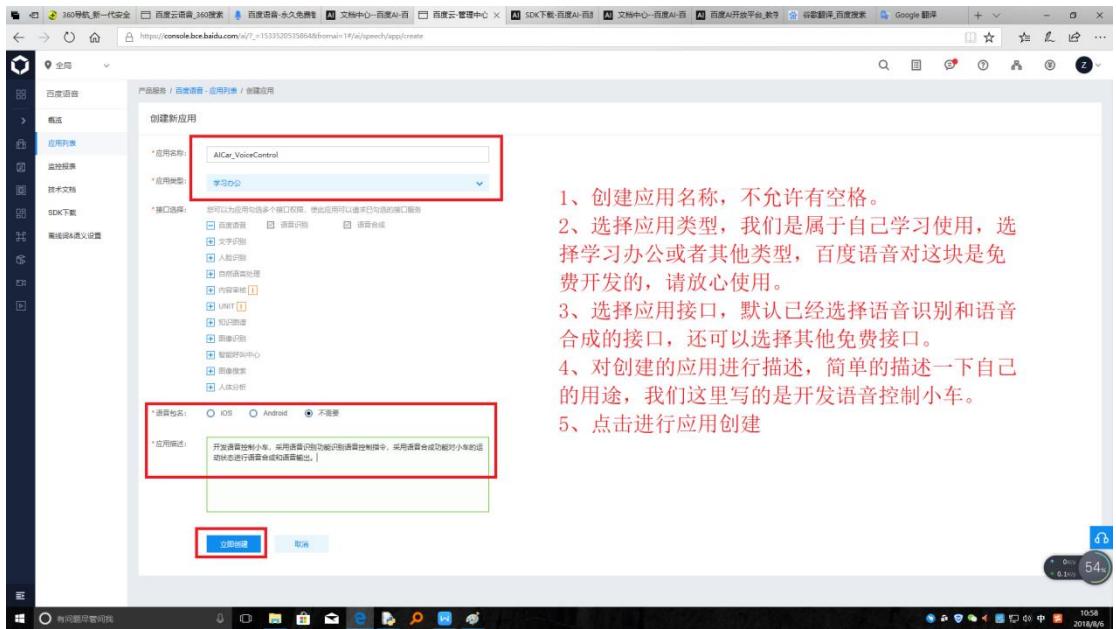


图 3.5 应用创建设置

4、应用创建完成后，下载相应的文档和软件开发包（SDK），我们用的是 python+树莓派进行开发，所以下载 Python-SDK 和相应的文档。

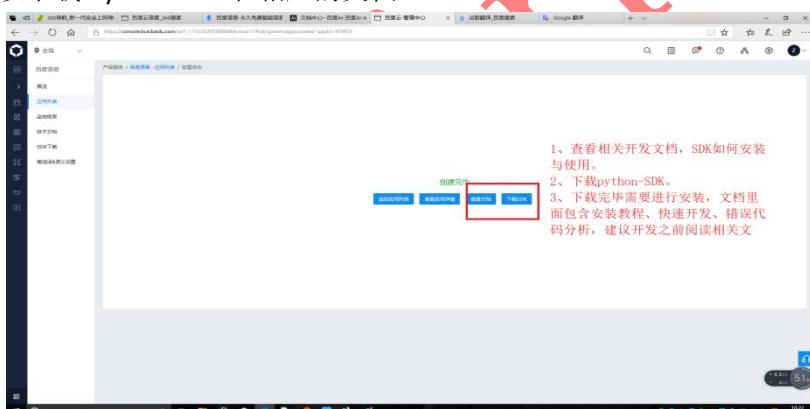


图 3.6 SDK 下载

### 5、Python-SDK 下载



6、SDK 的安装，将下载的 SDK 包拷贝到树莓派 pi 目录下，终端界面解压安装，安装过程如下：

1、解压： `unzip aip-python-sdk-2.0.0.zip`

```
pi@raspberrypi3B: ~
File Edit Tabs Help
pi@raspberrypi3B: ~ $ ls
aip-python-sdk-2.0.0.zip Documents Music Public Templates
Desktop Downloads Pictures python_games Videos
pi@raspberrypi3B: ~ $ unzip aip-python-sdk-2.0.0.zip
Archive: aip-python-sdk-2.0.0.zip
 creating: aip/
 inflating: aip/__init__.py
 inflating: aip/kg.py
 inflating: aip/base.py
 inflating: aip/imageclassify.py
 inflating: aip/nlp.py
 inflating: aip/speech.py
 inflating: aip/imagesearch.py
 inflating: aip/imagecensor.py
 inflating: aip/ocr.py
 inflating: aip/face.py
 creating: bin/
 inflating: bin/aip_client
 inflating: LICENSE
 inflating: setup.py
pi@raspberrypi3B: ~ $
```

2、安装 SDK 包: sudo pip install baidu-aip

```
Successfully built baidu-aip
Installing collected packages: baidu-aip
Successfully installed baidu-aip-2.2.5.2
```

### 安装语音识别 Python SDK

#### 语音识别 Python SDK 目录结构

```
|── README.md
|── aip
| ├── __init__.py //SDK目录
| ├── base.py //aip基类
| ├── http.py //http请求
| └── speech.py //语音识别
└── setup.py //setuptools安装
```

支持Python版本: 2.7.+ ,3.+

安装使用Python SDK有如下方式:

- 如果已安装pip, 执行 `pip install baidu-aip` 即可。
- 如果已安装setuptools, 执行 `python setup.py install` 即可。

图 3.7 SDK 下载安装

遵循以上步骤操作, 就搭建好属于自己的语音识别和语音合成平台, 再添加一个麦克风和一个喇叭或者音箱就可以进行语音识别方面的开发了, 语音控制小车, 控制智能家电等。

### 3.2.2 Python-SDK 实现语音识别和语音合成

#### 1、python-SDK 实现语音识别

遵循 SDK 文档进行快速开发, 务必阅读文档。如果对自己的录音进行测试, 需先采用格式工厂对录音文件进行转换, 格式工厂如何使用出门右拐百度一下。

```
#__*__ coding:UTF-8 __*
@author: zd1
百度云语音识别 Demo, 实现对本地语音文件的识别。
需安装好 python-SDK, 录音文件不超过 60s, 文件类型为 wav 格式。
音频参数需设置为 单通道 采样频率为 16K PCM 格式 可以先采用官方音频进行测试

导入 AipSpeech AipSpeech 是语音识别的 Python SDK 客户端
from aip import AipSpeech
import os

''' 你的 APPID AK SK 参数在申请的百度云语音服务的控制台查看'''
APP_ID = '11472625'
API_KEY = 'NYIvd23qqGAZ1ZPpVpCthENs'
SECRET_KEY = 'DcQWQ9HVc0sqoD091gFxWiCP1iOoNa6u'

新建一个 AipSpeech
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)

读取文件
```

```

def get_file_content(filePath):
 with open(filePath, 'rb') as fp:
 return fp.read()

def stt(filename):
 # 识别本地文件
 result = client.asr(get_file_content(filename),
 'wav',
 16000,
 {'dev_pid': 1536}) # dev_pid参数表示识别的语言类型
 1536 表示普通话
 print(result)

 # 解析返回值，打印语音识别的结果，将结果写入 demo.txt 中
 if result['err_msg']=='success.':
 word = result['result'][0].encode('utf-8')
 if word!='':
 if word[len(word)-3:len(word)]=='':
 print(word[0:len(word)-3])
 with open('demo.txt', 'w') as f:
 f.write(word[0:len(word)-3])
 f.close()
 else:
 print(word.decode('utf-8').encode('gbk'))
 with open('demo.txt', 'w') as f:
 f.write(word)
 f.close()
 else:
 print("音频文件不存在或格式错误")
 else:
 print("错误")

main 函数 识别本地录音文件 yahboom.wav
if __name__ == '__main__':
 stt('yahboom.wav')

```

识别的结果：

```

>>>
RESTART: C:\Users\Administrator\Desktop\AI_Car_class\Chapter_3_Code\stt_tts_demo
\stt.py
{'err_no': 0, 'corpus_no': u'6586503478344071441', 'err_msg': 'success.', 'result': [u'\u4e9a\u535a\u667a\u80fd\u79d1\u6280'], 'sn': u'5768541431815335398
44'}
亚博智能科技
>>>

```

图 3.8 识别结果

```
result = client.asr(get_file_content(filename), 'wav', 16000, {'dev_pid': 1536})
```

上述函数参数描述如下：

| 参数       | 类型     | 描述                                                                                          | 是否必须 |
|----------|--------|---------------------------------------------------------------------------------------------|------|
| speech   | Buffer | 建立包含语音内容的Buffer对象, 语音文件的格式, pcm 或者 wav 或者 amr。不区分大小写                                        | 是    |
| format   | String | 语言文件的格式, pcm 或者 wav 或者 amr。不区分大小写。推荐pcm文件                                                   | 是    |
| rate     | int    | 采样率, 16000, 固定值                                                                             | 是    |
| cuid     | String | 用户唯一标识, 用来区分用户, 填写机器 MAC 地址或 IMEI 码, 长度为60以内                                                | 否    |
| dev_pid  | Int    | 不填写lan参数生效, 都不填写, 默认1537 (普通话 输入法模型), dev.pid参数见本节开头的表格                                     | 否    |
| lan(已废弃) | String | 历史兼容参数, 请使用dev.pid, 如果dev.pid填写, 该参数会被覆盖。语种选择,输入法模型, 默认中文 (zh) 。 中文=zh、粤语=ct、英文=en, 不区分大小写。 | 否    |

图 3.9 sst 参数设置

dev\_pid 的参数描述如下, 一般我们选择 1536, 后期支持自定义词库, 可提高识别率。

| dev_pid | 语言             | 模型    | 是否有标点 | 备注       |
|---------|----------------|-------|-------|----------|
| 1536    | 普通话(支持简单的英文识别) | 搜索模型  | 无标点   | 支持自定义词库  |
| 1537    | 普通话(纯中文识别)     | 输入法模型 | 有标点   | 不支持自定义词库 |
| 1737    | 英语             |       | 有标点   | 不支持自定义词库 |
| 1637    | 粤语             |       | 有标点   | 不支持自定义词库 |
| 1837    | 四川话            |       | 有标点   | 不支持自定义词库 |
| 1936    | 普通话远场          | 远场模型  | 有标点   | 不支持      |

图 3.10 sst 识别语言设置

## 2、python-SDK 实现语音合成

语音合成较语音识别简单, 合成文本长度必须小于 1024 字节, 如果本文长度较长, 可以采用多次请求的方式。下面是合成本地文件 demo 程序:

```
#_*_ coding:UTF-8 _*
@author: zd1
百度云语音合成 Demo, 实现对本地文本的语音合成。
需安装好 python-SDK, 待合成本文不超过 1024 个字节
合成成功返回 audio.mp3 否则返回错误代码

导入 AipSpeech AipSpeech 是语音识别的 Python SDK 客户端
from aip import AipSpeech
import os

''' 你的 APPID AK SK 参数在申请的百度云语音服务的控制台查看'''
APP_ID = '11472625'
API_KEY = 'NYIvd23qqGAZ1ZPpVpCthENs'
SECRET_KEY = 'DcQWQ9HVc0sqoD091gFxWiCP1i0oNa6u'

新建一个 AipSpeech
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)

将本地文件进行语音合成
def tts(filename):
```

```

f = open(filename, 'r')
command = f.read()
if len(command) != 0:
 word = command
f.close()
result = client.synthesis(word, 'zh', 1, {
 'vol': 5, 'per': 0,
})
合成正确返回 audio.mp3, 错误则返回 dict
if not isinstance(result, dict):
 with open('audio.mp3', 'wb') as f:
 f.write(result)
 f.close()
 print 'tts successful'

main

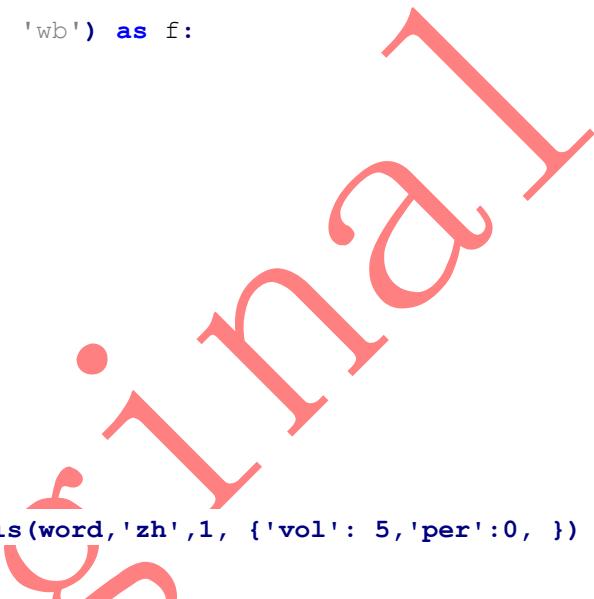
if __name__ == '__main__':
 tts('demo.txt')

```

#### 函数解析:

```
result = client.synthesis(word, 'zh', 1, {'vol': 5, 'per': 0, })
```

参数列表如下:



| 参数   | 类型     | 描述                                                | 是否必须 |
|------|--------|---------------------------------------------------|------|
| tex  | String | 合成的文本，使用UTF-8编码，请注意文本长度必须小于1024字节                 | 是    |
| cuid | String | 用户唯一标识，用来区分用户，填写机器 MAC 地址或 IMEI 码，长度为60以内         | 否    |
| spd  | String | 语速，取值0-9，默认为5中语速                                  | 否    |
| pit  | String | 音调，取值0-9，默认为5中语调                                  | 否    |
| vol  | String | 音量，取值0-15，默认为5中音量                                 | 否    |
| per  | String | 发音人选择, 0为女声, 1为男声, 3为情感合成-度逍遙, 4为情感合成-度丫丫, 默认为普通女 | 否    |

图 3.11 语言合成参数设置

错误码如下:

## 错误信息

### 错误返回格式

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error\_code**: 错误码。
- **error\_msg**: 错误描述信息，帮助理解和解决发生的错误。

### 错误码

| 错误码 | 含义        |
|-----|-----------|
| 500 | 不支持的输入    |
| 501 | 输入参数不正确   |
| 502 | token验证失败 |
| 503 | 合成后端错误    |

图 3.12 tts 错误分析

思考：如何实现实时录音语音识别并实现语音合成（参考解决方案：将实时识别的获得的 **result** 保存为文本，然后对 **result** 文本进行语音合成）

### 3.2.3 语音识别结果及错误分析

语音识别结果返回示例，失败返回的第一条信息为错误代码，参考官方文档的错误代码进行对比分析错误原因。

语音识别 返回数据参数详情

| 参数      | 类型  | 是否一定输出 | 描述                                          |
|---------|-----|--------|---------------------------------------------|
| err_no  | int | 是      | 错误码                                         |
| err_msg | int | 是      | 错误码描述                                       |
| sn      | int | 是      | 语音数据唯一标识，系统内部产生，用于 debug                    |
| result  | int | 是      | 识别结果数组，提供1-5个候选结果，string 类型为识别的字符串，utf-8 编码 |

返回样例：

```
// 成功返回
{
 "err_no": 0,
 "err_msg": "success.",
 "corpus_no": "15984125203285346378",
 "sn": "481D63F-73BA-726F-49EF-8659ACCC2F30",
 "result": ["北京天气"]
}

// 失败返回
{
 "err_no": 2000,
 "err_msg": "data empty.",
 "sn": null
}
```

图 3.13 sst 结果返回

## 错误码

| 错误码  | 用户输入/服务端 | 含义                | 一般解决方法                                                                                                                                    |
|------|----------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 3300 | 用户输入错误   | 输入参数不正确           | 请仔细核对文档及参照demo，核对输入参数                                                                                                                     |
| 3301 | 用户输入错误   | 音频质量过差            | 请上传清晰的音频                                                                                                                                  |
| 3302 | 用户输入错误   | 鉴权失败              | token字段校验失败。请使用正确的API_KEY 和 SECRET_KEY生成                                                                                                  |
| 3303 | 服务端问题    | 语音服务器后端问题         | 请将api返回结果反馈至论坛或者QQ群                                                                                                                       |
| 3304 | 用户请求超限   | 用户的请求QPS超限        | 请降低识别api请求频率 (qps) (appId计算, 移动端如果共用则累计)                                                                                                  |
| 3305 | 用户请求超限   | 用户的日pv (日请求量) 超限  | 请“申请提高配额”，如果暂未通过，请降低日请求量                                                                                                                  |
| 3307 | 服务端问题    | 语音服务器后端识别出错问题     | 目前请确保16000的采样率音频时长低于30s, 8000的采样率音频时长低于60s。如果仍有问题, 请将api返回结果反馈至论坛或者QQ群                                                                    |
| 3308 | 用户输入错误   | 音频过长              | 音频时长不超过60s, 请将音频时长截取为60s以下                                                                                                                |
| 3309 | 用户输入错误   | 音频数据问题            | 服务端无法将音频转为pcm格式, 可能是长度问题, 音频格式问题等。请将输入的音频时长截取为60s以下, 并核对下音频的编码, 是否是8K或者16K, 16bits, 单声道。                                                  |
| 3310 | 用户输入错误   | 输入的音频文件过大         | 语音文件共有3种输入方式: json 里的speech 参数 (base64后); 直接post 二进制数据, 及callback参数里url, 分别对应三种情况: js on超过10M; 直接post的语音文件超过10M; callback里回调url的音频文件超过10M |
| 3311 | 用户输入错误   | 采样率rate参数不在选项里    | 目前rate参数仅提供8000,16000两种, 填写4000即会有此错误                                                                                                     |
| 3312 | 用户输入错误   | 音频格式format参数不在选项里 | 目前格式仅仅支持pcm, wav或amr, 如填写mp3即会有此错误                                                                                                        |

图 3.14 错误码分析

遵循 SDK 开发文档进行开发时, 务必参考接口说明设置好参数, 这一步做好之后出现错误的几率比较小, 常见的错误为代码为 3301 的音频错误。音频错误一般是因为录音质量太差或者录音的频率不对。

参考 SDK 文档网址: <http://ai.baidu.com/docs#/ASR-Online-Python-SDK/top>

识别结果实际为空。可能是音频质量过差, 不清晰, 或者是空白音频。有时也可能是pcm填错采样率。如16K采样率的pcm文件, 填写的rate参数为8000。

## 3.3 离线语音唤醒

基于云方案的语音识别和语音控制, 大家最担忧的问题是自己的隐私问题, 假设没有语音离线唤醒, 使用者的实时语音都会直接上传到云服务器上, 对隐私造成威胁。所以为了避免这种问题, 本项目涉及的语音交互都采用了离线的唤醒引擎实现语音唤醒, 保证隐私安全。

### 3.3.1 离线唤醒引擎 SnowBoy

这里将简单介绍下比较广泛使用的离线唤醒引擎 SnowBoy, 想要全面了解 SnowBoy 参考官网英文文档 <http://docs.kitt.ai/snowboy/>, 文档介绍了如何安装 SnowBoy 和相关的插件及相关错误如何修改等, 最后实现了控制语音唤醒控制 LED 灯。我们在这里简洁介绍 SnowBoy, 然后动手运用 SnowBoy 实现我们自己的离线唤醒控制。

Snowboy 是一款高度可定制的唤醒词检测引擎, 可以用于实时嵌入式系统, 并且始终监听 (即使离线)。当前, 它可以运行在 Raspberry Pi、(Ubuntu) Linux 和 Mac OS X 系统上。唤醒词用于, 发起一个完整的语音交互界面。除了此, 唤醒词还可以用于其他用途, 比如执行简单的命令和控制动作。在一个棘手的解决方案中, 它可以运行完整的自动语音识别 (ASR, Automatic Speech Recognition) 来执行热词检测。在这种情况下, 设备将在自动语音识别转录中观察特定的触发词。转录中观察特定的触发词。另外, 当使用基于云的解决方案时, 它也不会保护您的隐私。幸运的是, Snowboy 被创造出来, 解决这些问题。

Snowboy 具有以下的特性:

- 高度可定制。允许自行修改和定义唤醒词。
- 一直监听, 但是保护您的隐私。Snowboy 不连接到网络, 离线唤醒。

➤ 轻巧的、可嵌入，可以让您在 Raspberry Pi 上运行。

下面让我们动手实践来安装 SnowBoy，然后使用它为我们的语言识别添加隐私保护功能。

#### Decoder Structures

The decoder tarball contains the following files:

```
├── README.md
├── _snowboydetect.so
├── demo.py
├── demo2.py
├── light.py
├── requirements.txt
├── resources
│ ├── ding.wav
│ ├── dong.wav
│ └── common.res
└── snowboy.umdl
 └── snowboydecoder.py
 └── snowboydetect.py
 └── version
```

SnowBoy 文件结构

- 1、demo 一个回调函数，测试SnowBoy
- 2、demo2 包含两个回调函数

图 3.15 SnowBoy 文件结构

1、电脑端下载 SnowBoy 的二进制文件，链接如下：

<https://s3-us-west-2.amazonaws.com/snowboy/snowboy-releases/rpi-arm-raspbian-8.0-1.2.0.tar.bz2>

2、电脑解压下载的文件到 SnowBoy，并上传到树莓派 pi 目录下

3、确保树莓派安装好 ‘swig’，‘sox’，‘portaudio’，‘atlas’。

安装指令如下：  
sudo apt-get install swig3.0 python-pyaudio sox  
pip install pyaudio  
sudo apt-get install libatlas-base-dev

测试录音：rec temp.wav

测试没有问题后进入 SnowBoy 文件夹测试官方 demo，测试时最好接上音频输出装置：

命令：sudo python demo.py resources/snowboy.umdl

```
pi@raspberrypi3B:~ $ cd SnowBoy
pi@raspberrypi3B:~/SnowBoy $ ls
demo2.py README.md snowboydecoder.py snowboydetect.pyc
demo.py requirements.txt snowboydecoder.pyc _snowboydetect.so
light.py resources snowboydetect.py version
pi@raspberrypi3B:~/SnowBoy $ sudo python demo.py resources/snowboy.umdl
```

```
Listening... Press Ctrl+C to exit
```

图 3.16 SnowBoy 测试

说出唤醒词 snowboy 会触发唤醒系统，3.5mm 音频输出 ding 声

### 3.3.2 离线唤醒控制 LED 灯

本小节我们采用 snowboy 实现一个智能家居的一个小项目，实现语音唤醒控制 LED 灯。下面我们分析一下 snowboy 实现离线唤醒的代码：

```
#_*_ coding:UTF-8 _*_
@author: zd1

snowboy demo 代码分析
使用 snowboy 离线唤醒实现控制 LED 灯
reference:
http://docs.kitt.ai/snowboy/#my-trained-model-works-well-on-laptops-b
```

ut-not-on-pi-s

```
导入 SDK 包
import snowboydecoder
import sys
import signal

interrupted = False

def signal_handler(signal, frame):
 global interrupted
 interrupted = True

def interrupt_callback():
 global interrupted
 return interrupted

判断命令行执行指令是否传入唤醒词模型，无则退出程序
if len(sys.argv) == 1:
 print("Error: need to specify model name")
 print("Usage: python demo.py your.model")
 sys.exit(-1)

唤醒词模型为输入的参数，这里可以进行修改
#model = sys.argv[1]
model = 'resources/snowboy.umdl' # 修改 model，指定其文件名

capture SIGINT signal, e.g., Ctrl+C
signal.signal(signal.SIGINT, signal_handler)

唤醒词检测函数，调整 sensitivity 参数可修改唤醒词检测的准确性
detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)
print('Listening... Press Ctrl+C to exit')

main loop
回调函数 detected_callback=snowboydecoder.play_audio_file
修改回调函数可实现我们想要的功能
detector.start(detected_callback=snowboydecoder.play_audio_file,
 interrupt_check=interrupt_callback,
 sleep_time=0.03)

释放资源
detector.terminate()
```



结合 demo 函数的分析，实现属于自己的语音唤醒平台，我们只需要修改三个参数：

- 1、修改 model `model = 'resources/snowboy.umdl' # 修改 model，指定 model 文件名`
- 2、修改热词检测灵敏度，`detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)`  
    `# 修改 sensitivity 的值`
- 3、修改回调函数 `detected_callback= own_callbacks`

下面实现离线唤醒触发点亮 LED 灯：这里我们依然采用 snowboy 官方的唤醒词模型 snowboy.umdl，只是修改回调函数，程序如下：

#### 1、定义 `own_callback`

```
def own_callback():
 LED.lighton() # 点亮 LED 灯
```

#### 2、修改 `detected_callback`

```
detected_callback = own_callback 注意回调函数后面不在加括号
```

程序见课程源码

### 3.3.3 修改个性唤醒词

SnowBoy 的特点之一是高度可定制性，可以自己定义属于自己的唤醒词。同时也可以训练自己的唤醒词，语料越多，识别的准确率就越高，下面是如何修改唤醒词的步骤：

SnowBoy 的官网：<https://snowboy.kitt.ai/>

登陆

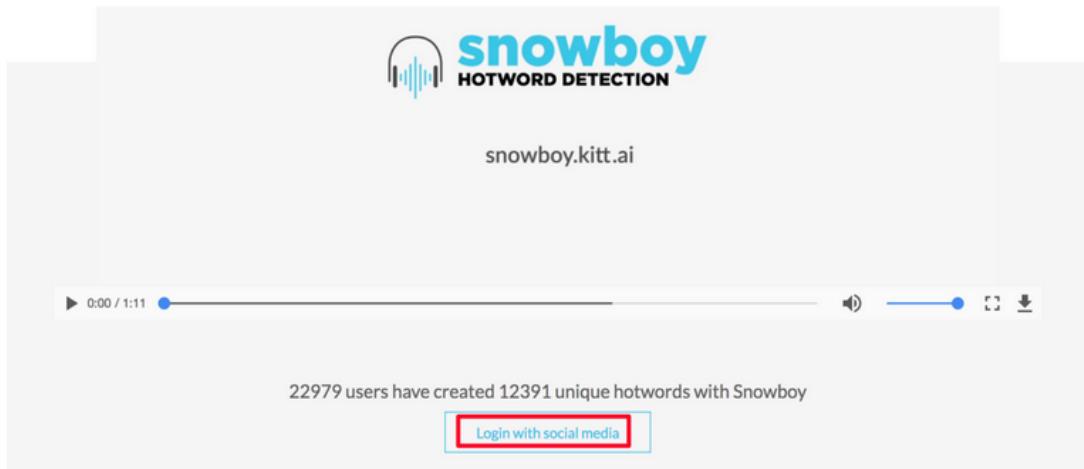


图 3.17 snowboy 官网

三个登录选项：GitHub/Google/Facebook，一般选择注册 GitHub 登录



图 3.18 登录界面

点击创建热词（Hotword） 注意创建热词时最好使用火狐或者 Google chrome 浏览器



图 3.19 热词创建

创建热词需要接入麦克风，需要打开浏览器录音权限，录入三遍你想要定义的唤醒词，上传训练之后会生成模型，下载即可。本实现训练的唤醒词是：你好，亚博。欢迎大家为我们的唤醒词 **yahboom** 添加语料，提高唤醒的准确度。

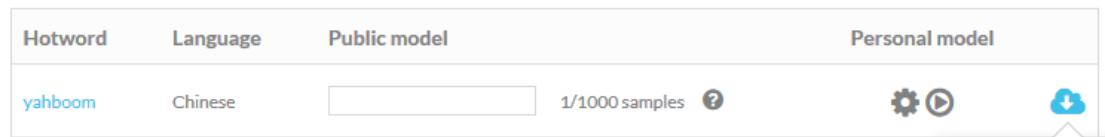


图 3.20 热词下载

## 3.4 语音唤醒及语音识别实现语音交互

### 3.4.1 语音控制及语音机器人

结合语音识别和离线语音唤醒的知识，本节实现一个语音唤醒和语音控制的综合性实验。控制 LED 灯的开关，本实验的 LED 灯不是 220V 家用的，如果需要扩展，添加一个继电器即可。下面我们需要将语音唤醒和语音识别结合起来，并采用我们自己个性唤醒词。上一节探讨过使用语音唤醒实现 LED 灯的控制，我们只是简单的修改了回调函数就实现了功能。那么如何实现语音唤醒和语音识别的结合呢？下面可以分析一下。

语音唤醒的流程如下图：

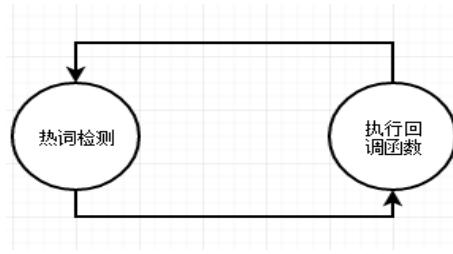


图 3.21 语音唤醒流程

有了这个图是不是比较直观，我们把语音识别放在回调函数中即可实现语音唤醒和语音识别的功能，snowboy 和语音识别都是采用 pyaudio 录音，如果同时使用会出现设备冲突，所以在使用语音识别之前需要关闭 snowboy 功能，语音识别完毕打开 snowboy 功能即可。所以回调函数的编写就比较清晰，程序流程图如下：

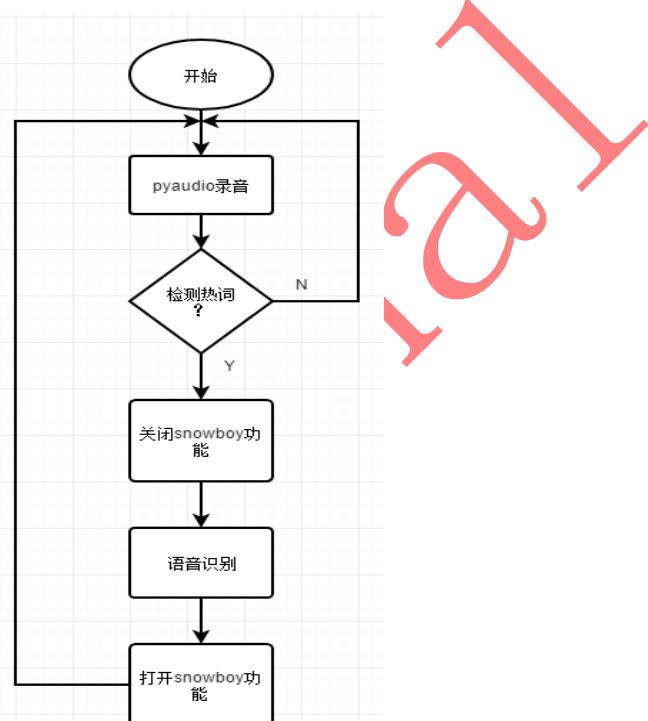


图 3.22 语音唤醒+语音识别交互控制

结合上述的流程图，可以很清晰的实现语音交互控制，所以我们基于上面的流程图实现一个语音交互机器人，可以实现简单的语音对话，语音控制，天气查询等功能，我们这里会讲解一下主程序是如何实现的。下面附上主程序的源码：

```

#_*_ coding:UTF-8 _*_
@author: zd1
实现离线语音唤醒和语音识别，实现一些语音交互控制
voice_robot 实现语音开关控制，语音对话

导入 SDK 包
import snowboydecoder
import sys
import signal
import record_monitor as recordMonitor

```

```
interrupted = False

def signal_handler(signal, frame):
 global interrupted
 interrupted = True

def interrupt_callback():
 global interrupted
 return interrupted

回调函数，语音识别在这里实现
def callbacks():
 global detector

 # 语音唤醒后，提示 ding 两声
 snowboydecoder.play_audio_file()
 snowboydecoder.play_audio_file()

 # 关闭 snowboy 功能
 detector.terminate()
 # 开启语音识别
 recordMonitor.monitor()
 # 打开 snowboy 功能
 wake_up() # wake_up --> monitor --> wake_up

 # 热词唤醒
 def wake_up():
 global detector
 model = 'yahboom.pmdl' # 唤醒词为 你好，亚博'
 # capture SIGINT signal, e.g., Ctrl+C
 signal.signal(signal.SIGINT, signal_handler)

 # 唤醒词检测函数，调整 sensitivity 参数可修改唤醒词检测的准确性
 detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)
 print('Listening... please say wake-up word:你好，亚博')
 # main loop
 # 回调函数 detected_callback=snowboydecoder.play_audio_file
 # 修改回调函数可实现我们想要的功能
 detector.start(detected_callback=callbacks, # 自定义回调函数
 interrupt_check=interrupt_callback,
 sleep_time=0.03)

 # 释放资源
```

```

detector.terminate()

if __name__ == '__main__':
 wake_up()

 # 关闭 snowboy 功能
 detector.terminate()
 # 开启语音识别
 recordMonitor.monitor()
 # 打开 snowboy 功能
 wake_up() # wake_up -> monitor -> wake_up

```

这几句代码实现了重复语音唤醒的功能，snowBoy 和语音识别都采用 pyaudio 实现录音，一起使用时会抛出 IO 异常，所以需要先关闭 snowboy 功能然后开启语音识别功能，语音识别完毕需要释放 pyaudio，然后开启 snowboy 功能。语音识别和语音控制的都封装在 monitor() 函数中。

### 3.4.2 控制 AI 车的运动

语音控制及语音机器人一节中，我们将离线语音唤醒、语音识别和语音合成三个部分融合一起打造了一款简单的语音机器人，这一节我们来实现一个语音控制小车，区别于传统的遥控小车。实现方法可以参考 3.4.1 小节，主要是修改 snowboy 的回调函数部分，程序流程图如下：本小节的内容只是在语音识别后执行的命令有所不同，其他的程序是一致的，结合 AI 车运动控制单元的讲解，可以很轻松实现一款属于自己的语音控制小车，亲自动手实践一下吧。程序见源代码。

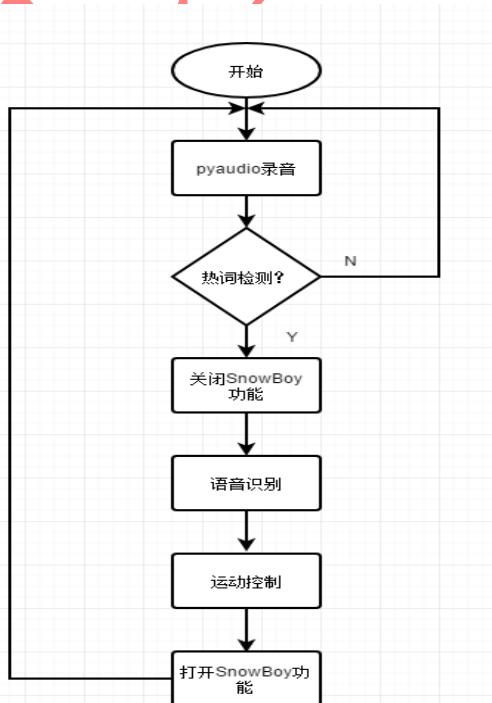


图 3.23 语音控制 AI 车运动

## 第四章 基于人工神经网络的自动驾驶小车

### 4.1 有趣的神经网络

神经网络是什么，神经网络有什么作用，神经网络如何实现？初学神经网络者都存在上述的三大困惑，本章节以上述的三个问题出发，同时也以其作为落脚点。下面开始我们的神经网络探索之旅，笔者也是神经网络初学者，下面只是我个人的开发经验，难免会有出错的地方，还望读者海涵。欢迎一起学习交流。这里推荐一本神经网络入门较好的书籍《Python 神经网络编程》。本章节谈到的神经网络都是指人工神经网络 ANN。

#### 4.1.1 浅谈神经网络

本小节致力于解决神经网络的前两个问题，神经网络是什么？神经网络有什么用。

神经网络是一种模拟人脑的神经网络，以期能够实现类人工智能的机器学习技术。下图表达了神经网络与人工智能之间的关系，由此可见人工神经网络在机器学习、人工智能领域的重要性，但是我们还是没有解决神经网络是什么的问题，下面我们慢慢阐述。在理解人工神经网络之前，我们先来看一下生物的神经网络，从里面寻找灵感。

图 4.2 描述的是一个生物神经元的模型，基本上所有的神经元都是将电信号从一端传输到另一端，沿着轴突，将电信号从树突传到树突。单个神经元之间也会相互连接，生物电信号从一个神经元传递到另一个神经元。人的身体感知周边环境并反馈给大脑作出相应反应就是基于神经元之间的信号传递，实现人体感知的功能。然而，如何实现机器感知呢，如果能够实现一个‘人造神经元’不就可以实现机器感知。于是科学家们基于生物神经网络进行建模，构造了一个‘人造神经元’模型，也叫感知器（perceptron）。这个上世纪六十年代建立的感知器模型，在今天还在广泛使用。图 4.3 是感知器的模型，圆圈代表一个感知器，它可以接受多个输入，但只产生一个输出。这里我们来举个例子，考虑输出只有 0 和 1 两种情况。例子的背景是去看一部新上映的电影，所以输出为去或者不去看电影，确定去不去看电影我们需要考虑以下几个因素：1、天气是不是舒适，2、是否有人陪我一起去，3、电影院的消费贵不贵。以上的三个因素作为感知器的输入，看电影或者不看电影为感知器的输出，这样就构建了一个简易的感知器，模型图如 4.4 所示。

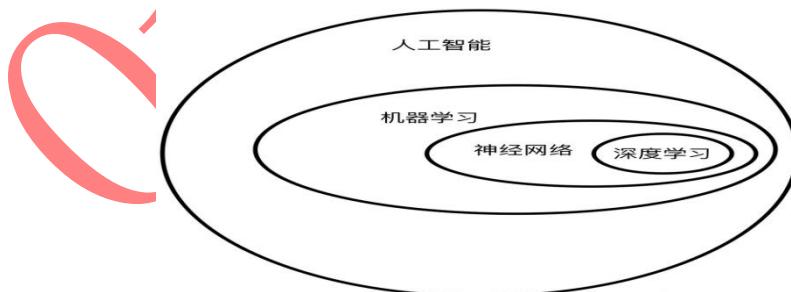


图 4.1 关系图

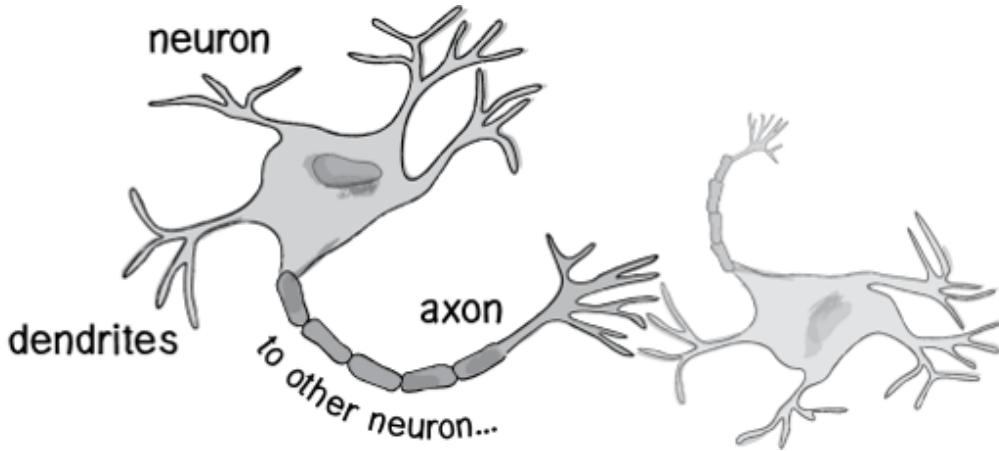


图 4.2 生物神经元模型

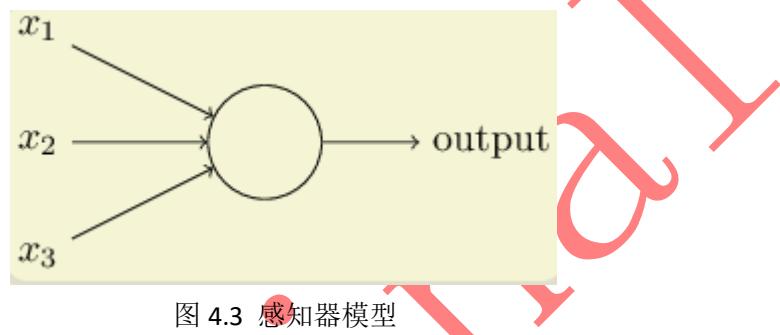


图 4.3 感知器模型

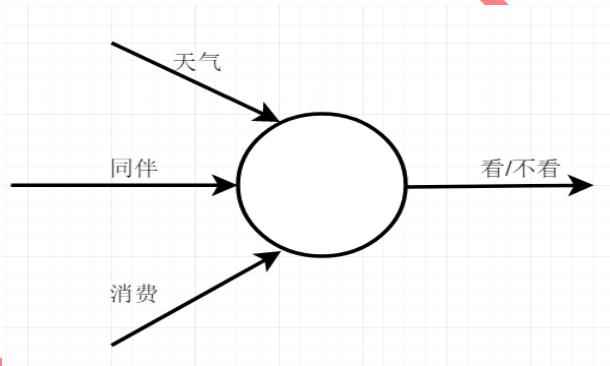


图 4.4 简易感知器模型

图 4.4 描述的是一个简易的感知器模型，然而这个并不准确，因为每个因素的重要性不一致，比如说女朋友说要去看电影，其他两个因素基本不需要考虑。所以需要给这个感知器的输入加上权重。我们假设输出和输入是线性的关系，那么输出是输入的代数表达式，只考虑输出只有 0 和 1 两种情况，那么输出肯定需要设置阈值高于阈值为 1 否则为 0，那么我们可以修改一下这个简单的感知器，模型图如下：

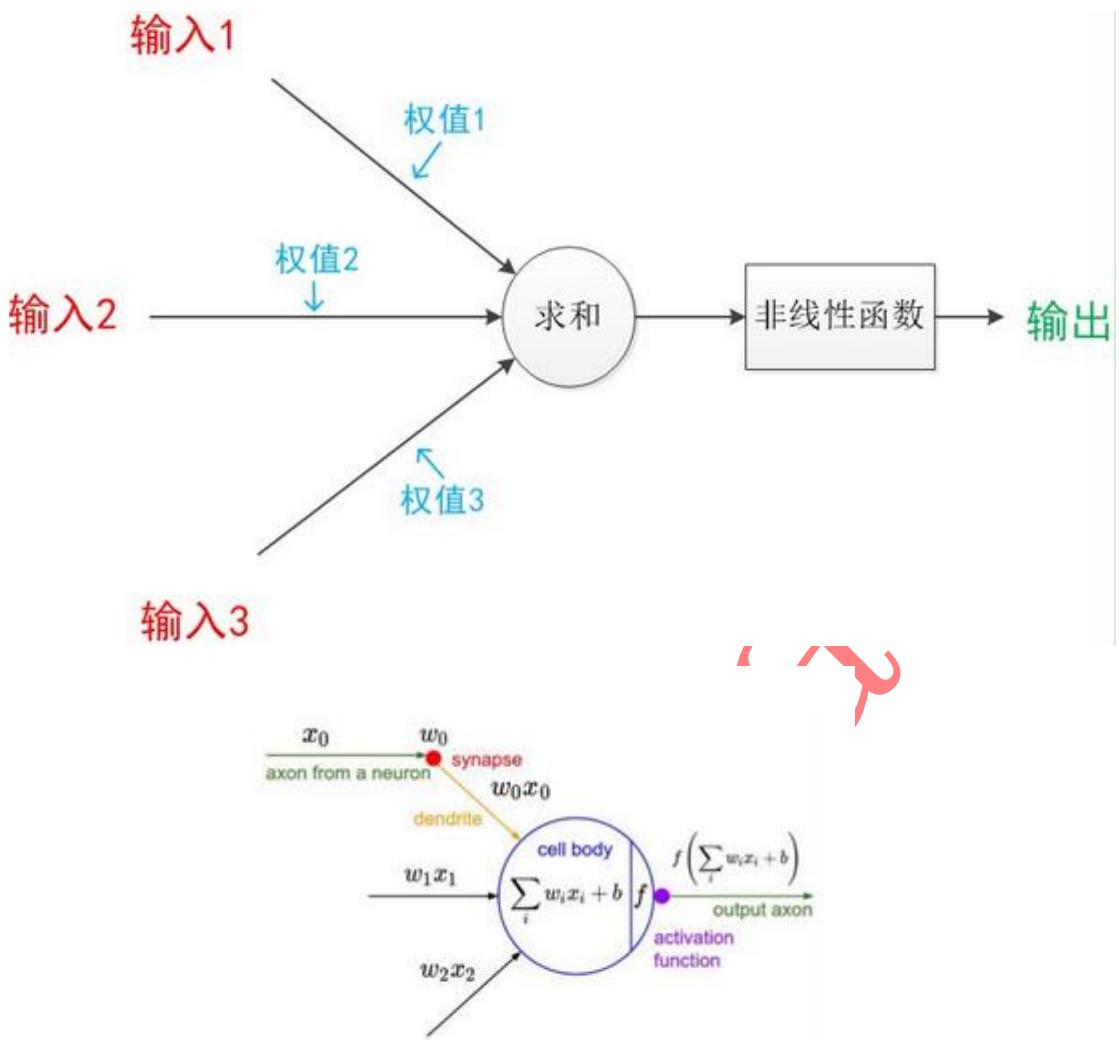


图 4.5 高级感知器

图 4.5 描述的一个高级的感知器模型，其本质为神经元模型，是组成神经网络的最小单元，由多个单元可以组建属于自己的神经网络。由单个感知器的知识，下面我们来认识人工神经网络的构成，并阐述其作用。图 4.6 描述的是一个经典的神经网络结构，即一个神经网络包括输入层、隐藏层、输出层，每一层的每一个单元都是一个高级感知器，上面几部分构建了神经网络的主体结构。下面总结以下神经网络的构成：

- 1、结构。输入层数量、隐藏层数量、输出层数量。
- 2、高级感知器。权重分配，感知器的输出函数（激活函数）
- 3、学习方法。反馈机制、监督学习、无监督学习。

神经网络的作用：

- 1、实现分类器 线性或者非线性分类器
- 2、实现控制器，预测控制
- 3、机器学习、深度学习

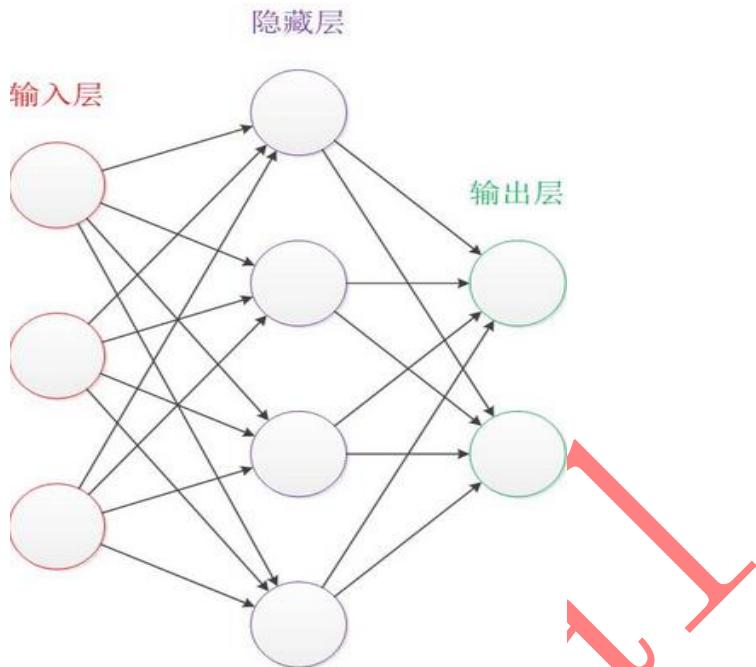


图 4.6 神经网络结构

#### 4.1.2 构建自己的神经网络

本小节致力于解决如何实现神经网络的问题。

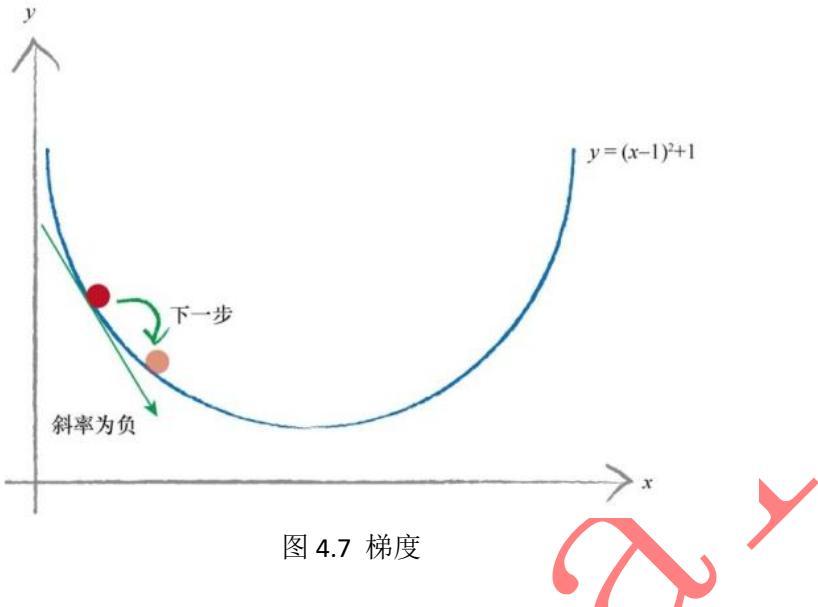
我们已经初步了解了神经网络的结构组成和神经网络的左右，那么这一节我们来实现搭建自己的神经网络模型，为后面采用神经网络实现分类器打下基础。这里不涉及具体的公式推导和解释，具体可以参考本章开头推荐的书籍。下面我们为编写神经网络的代码做点准备，分析一下神经网络程序的构成，其结构如下：

- 1、神经网络的初始化。输入层、隐藏层、输出层数量的确定、初始化感知器权重、初始化学习率，初始化激活函数。
- 2、神经网络的训练。权重更新的机制（梯度下降法）
- 3、神经网络的输出

对于神经网络，我们比较关心的问题是单个神经元（感知机）的权重是如何更新的，下面我们就简要介绍一下梯度下降法和反向传播过程。

**梯度下降法（瞎子下山法）：**梯度的本意是一个向量（矢量），表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）。对于单变量函数，梯度就是函数的导数，对于多变量函数，梯度对应函数方向导数的最大值。下图为函数  $y=(x-1)^2+1$  的图像，该函数在某点的梯度就是函数在某点的导数（斜率），如何求解该函数的最小值呢，传统的方法当然是采用求导极值法了，当然这里我们不这样做，我们采用梯度下降法来实现一下。何为梯度下降法，我们举个例子，一个人被困在山上，夜晚没有灯光，只有手电筒，现实条件决定他的可视度很低，想要快速到达山底的路径无法确定，所以他必须利用周围的环境信息来帮助自己，这个时候就可以采用梯度下降的方法快速下山。具体而言就是，以他当前的位置为基准，寻找这个位置最陡峭的地方（梯度最大），然后朝山高度下降的方向走（梯度相反的方向），每走一段距离重复上述方法一次，最后就能成功的到达山底。所以梯度下降法和快速下山的原理很类似，我们假设山就是一个函数，山底目标就是函数的最小值，对应到函数中，就是找到该点

的梯度，然后朝着梯度相反的方向改变，我们选取一定的间隔重复上述的过程，就能到达函数局部的最小值，直至迭代到函数的最小值。



梯度下降的数学解释：

$$\Theta^1 = \Theta^0 - \alpha \nabla J(\Theta) \quad \text{evaluated at } \Theta^0$$

此公式的意义是：J 是关于  $\Theta$  的一个函数，我们当前所处的位置为  $\Theta^0$  点，要从这个点走到 J 的最小值点，也就是山底。首先我们先确定前进的方向，也就是梯度的反向，然后走一段距离的步长，也就是  $\alpha$ ，走完这个段步长，就到达了  $\Theta^1$  这个点！

$$\Theta^1 = \Theta^0 - \alpha \nabla J(\Theta) \quad \text{evaluated at } \Theta^0$$

下面就这个公式的几个常见的疑问：

$\alpha$  是什么含义？

$\alpha$  在梯度下降算法中被称作为学习率或者步长，意味着我们可以通过  $\alpha$  来控制每一步走的距离，以保证不要步子跨的太大扯着蛋，哈哈，其实就是要走太快，错过了最低点。同时也要保证不要走的太慢，导致太阳下山了，还没有走到山下。所以  $\alpha$  的选择在梯度下降法中往往是很重要的！ $\alpha$  不能太大也不能太小，太小的话，可能导致迟迟走不到最低点，太大的话，会导致错过最低点！

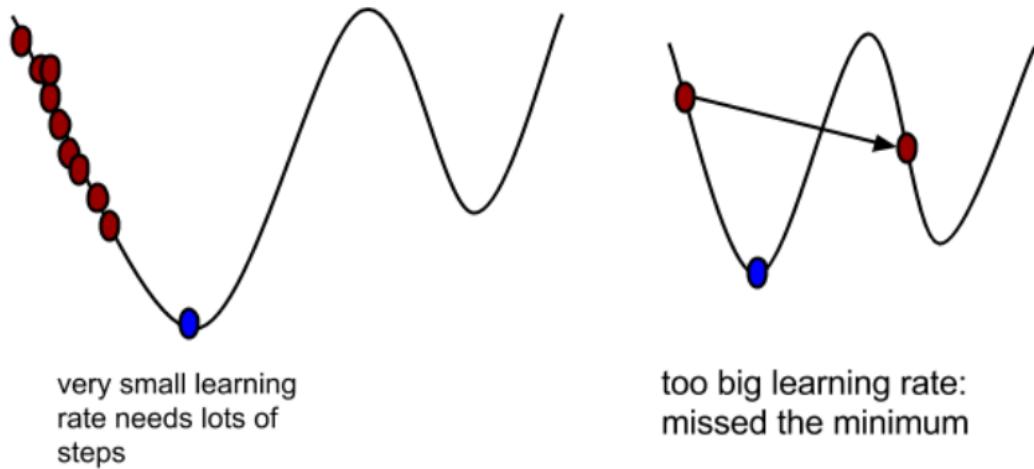


图 4.9 学习率作用

为什么要梯度要乘以一个负号？

梯度前加一个负号，就意味着朝着梯度相反的方向前进！我们在前文提到，梯度的方向实际就是函数在此点上升最快的方向！而我们需要朝着下降最快的方向走，自然就是负的梯度的方向，所以此处需要加上负号。

对于 ANN 而言是如何选取我们的目标函数，然后通过寻找目标函数的最小值实现权重的更新。这里我们选择神经网络目标值和实际输出值误差的平方来作为 ANN 的目标函数，那么其梯度为：

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial}{\partial w_{j,k}} \sum_n (t_n - o_n)^2$$

根据梯度下降的数学解释，权重更新的公式如下：

$$\Delta w_{j,k} = \alpha * E_k * \text{sigmoid}(O_k) * (1 - \text{sigmoid}(O_k)) \cdot O_j^T$$

下面是程序源码的讲解。

上述参考：<https://www.jianshu.com/p/c7e642877b0e> 和《Python 神经网络编程》

### 4.1.3 识别手写数字

基于前两节我们已经了解了神经网络的作用，神经网络如何实现，下面我们来实现一个神经网络入门级的小项目识别手写数字，基于 mnist 数据集的。神经网络实现的过程如下：

1、数据集准备与制作，标注数据 label。

2、初始化神经网络，输入层为 784，隐藏层随机选取，学习率 0.05-0.3 之间，输出层

10 层分别表示 0-9。

3、神经网络训练

4、神经网络测试

实测结果如下：

5

```
[[3.74033705e-06]
 [4.14609979e-05]
 [7.40444442e-06]
 [1.55752203e-08]
 [3.43761199e-08]
 [2.38251050e-02]
 [6.61436111e-09]
 [1.90192510e-05]
 [1.06345355e-10]
 [6.15987870e-10]]
识别的数字是： 5
>>> |
```

程序源码见源文件

Minst 数据集采用黑底白字的方法，我们需要将灰度图像反色

## 4.2 神经网络构建自动驾驶小车

### 4.2.1 自动驾驶小车的工作原理

基于神经网络知识的学习，我们的终极目标是实现一个自动驾驶小车。那么其背后的原理是什么？我们可以将神经网络看作一个黑箱子，我们输入采集赛道的图片，黑箱子对其进行分类输出赛道类型，如下图。



图 4.10 自动驾驶工作原理

从上图来看，实现自动驾驶的原理很简单，摄像头采集的图像经过黑箱，黑箱输出预测的赛道类型，完成预测，后边就是电机控制的问题了，其本质是图像的分类。我们的重点是黑箱的制作和使用。上述过程中黑箱的作用只是预测的功能，并没有完成学习的功能。所以基于神经网络的自动驾驶小车，我们主要完成三部分的工作，一是赛道的学习构建模型、二是模型加载实时视频预测、三是小车的控制。

### 4.2.2 自动驾驶小车的实现

#### 一、赛道的预学习构建模型

我们已经知道如何运用神经网络实现手写数字体的识别，识别手写数字体的过程本质上是图像分类的过程，所以自动驾驶小车的核心是实现赛道的分类。我们采用 ANN 来实现图像的分类，神经网络训练的架构如下图：

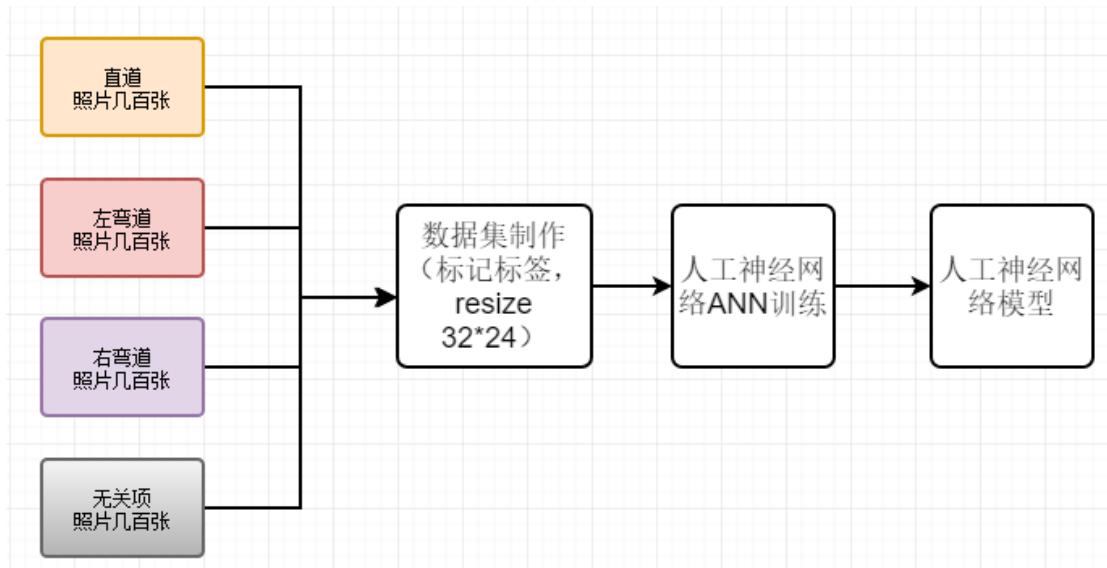


图 4.11 神经网络训练结构

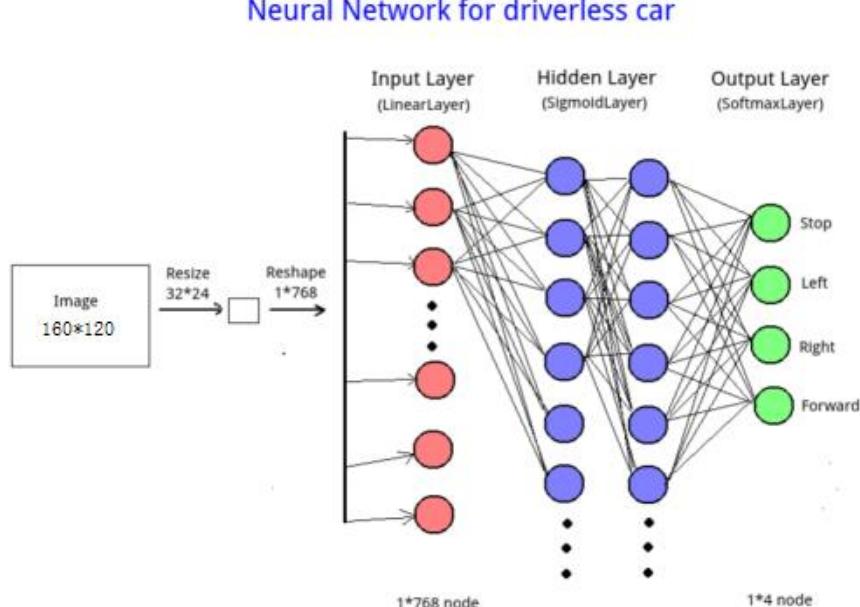


图 4.11 ANN 的组成

输入层由训练输入的照片尺寸决定，为  $32 \times 24$  即输入层为 768 层。

隐藏层初步可选 100 层左右。

输出层（目标层）初步为 4 层，即停止、前进、左转、右转。

将数据标记完成后输入至神经网络中进行训练迭代，实现监督学习。

训练的过程数据量较大最好在 PC 机上完成。

训练完成后要测试模型的正确率，神经网络的准确率较差时，修改学习率、隐藏层或者训练世代的大小重新训练模型，直至找到一个准确率较好的模型。

## 二、实时赛道视频预测

这一部分和训练过程是分开的，我们将训练好的模型加载和神经网络预测函数复制到树莓派平台，在树莓派上实现实时赛道预测。预测实现的流程如下：

1、初始化神经网络，与训练时的神经网络参数（层数、学习率）相同。

- 2、加载神经网络模型
- 3、摄像头实时采集赛道图像，并 resize 图像为 32\*24 大小
- 4、神经网络预测，根据概率值输出当前赛道类型。

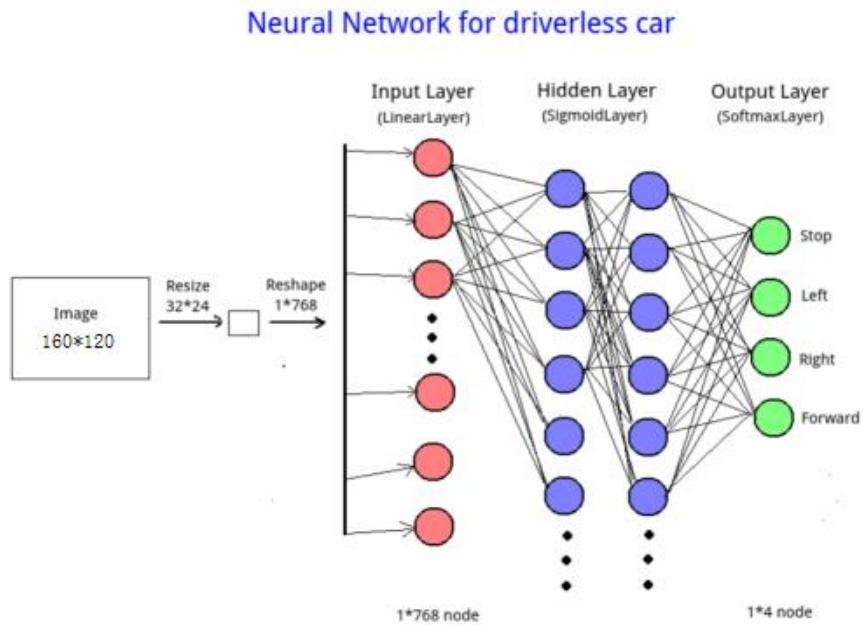


图 4.12 神经网络预测流程

### 三、小车运动控制

结合神经网络的输出结果可实现实时小车行驶轨道的道路情况分析，预测当前道路为直道、左弯道、右弯道、非轨道等情形，对神经网络输出的结果进行指令解析，在定时器中实现小车的实时控制。这个可以结合前面的课程学习。

下面是采用 ANN 实现的自动驾驶的源码，其他程序文件见源文件。

```
-*- coding: UTF-8 -*-
auto_control.py python2.7.13
ANN 实现自动驾驶
@author: zd1
import sys
import cv2
import threading

from ANN_predict import Predictor # 导入神经网路
import AICarRun as Run # 导入电机控制

小车状态值定义
enSTOP = 0
enRUN = 1
enBACK = 2
enLEFT = 3
enRIGHT = 4
```



```
global g_CarState
g_CarState = 0
time_flag = 0
timer
def time_interrupt():
 global timer
 global time_flag
 time_flag = 1
 timer = threading.Timer(0.1, time_interrupt)
 timer.start()
motor control
def motor_control(command):
 if command == 0:
 g_CarState = enRUN
 print g_CarState
 elif command == 4:
 g_CarState = enSTOP
 elif command == 2:
 g_CarState = enLEFT
 elif command == 3:
 g_CarState = enRIGHT
 else:
 g_CarState = enSTOP
 print g_CarState
 if g_CarState == enSTOP:
 Run.forward(0, 0)
 elif g_CarState == enRUN:
 Run.forward(70, 70)
 print 'Run'
 elif g_CarState == enLEFT:
 Run.forward(0, 60)
 elif g_CarState == enRIGHT:
 Run.forward(60, 0)
 else:
 Run.forward(0, 0)
main
def main():

 global time_flag
 Run.motor_init() # 电机初始化
 model = None
 if len(sys.argv) > 1:
 model = sys.argv[1]
 print 'model load success'
```

```

font = cv2.FONT_HERSHEY_SIMPLEX
加载模型实现预测
predictor = Predictor(model)
开启摄像头 设置分辨率为 160*120
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
if cap.isOpened() != 1:
 print 'camera is error'
time_interrupt() # 定时器初始化
while (cap.isOpened()):
 ret, frame = cap.read()
 img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
 if time_flag == 1:
 # 预测控制
 direction, command = predictor.predict(img) #stream
 motor_control(command)
 time_flag = 0
 # display
 cv2.putText(frame, direction, (20, 40), font, 1, (0, 255, 0), 2)
 cv2.imshow('image', frame)
 if cv2.waitKey(1) & 0xFF == 27:
 break
 # release
Run.GPIO_release()
cap.release()
cv2.destroyAllWindows()
main
if __name__ == '__main__':
 main()

```

### 4.2.3 测试

测试效果如下：

```

probability value: 0.756739624298
direction: forward
probability value: 0.583436360269
direction: forward

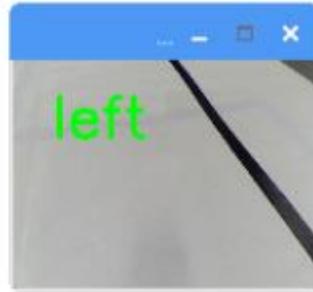
```



```
probability value: 0.299375580414
direction: right
probability value: 0.299375581264
direction: right
probability value: 0.299375580414
direction: right
```



```
probability value: 0.387616373647
direction: left
probability value: 0.387616373647
direction: left
probability value: 0.238080798429
direction: left
probability value: 0.387616385136
direction: left
probability value: 0.387616373647
direction: left
probability value: 0.387616373647
direction: left
```



```
probability value: 0.942410543936
direction: idle
```



OR