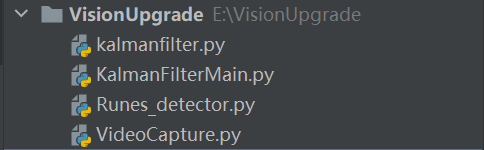
视觉原理说明

封装思路：整个框架主要包含相机驱动、目标检测、单目测距、卡尔曼滤波，单目测距虽然程序较为简单，但需要获取返回值w，因此将其放在遍历各轮廓的循环中即目标检测模块中。最终，我们将其封装成三大类，一个是卡尔曼滤波的predict类，一个是目标检测的detect类，还有一个是相机的驱动与调用VideoCapture类。详细情况如下：

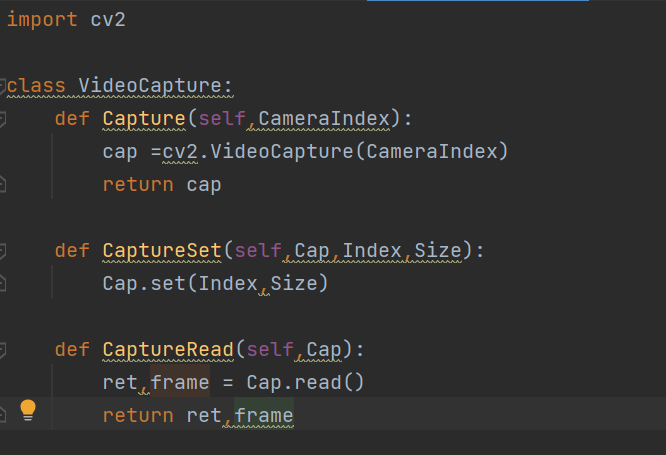


本组视觉使用的是opencv+python,使用树莓派作为视觉主控板，采用基于USB接口的dostyle高清视频摄像头作为计算机视觉传感器。以下将就相机驱动，目标识别，视觉测距，行动预测等四个方面展开成果展示。

1. 相机的驱动：

VideoCapture类提供了构造方法VideoCapture(),用于完成摄像头的初始化工作。具体代码如下：

\*\*\*\*\*

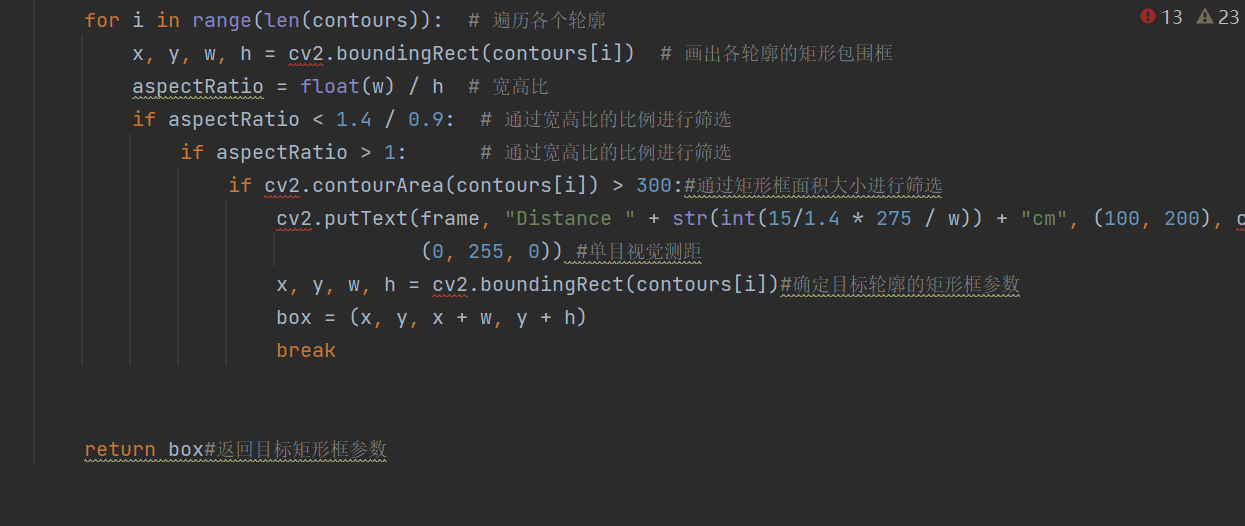
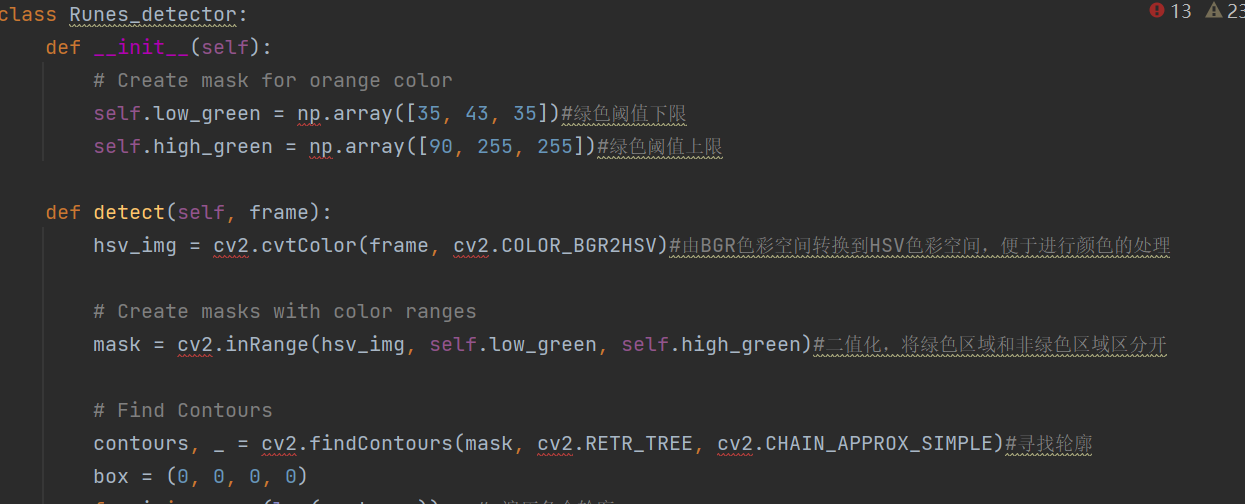


1. 目标识别：

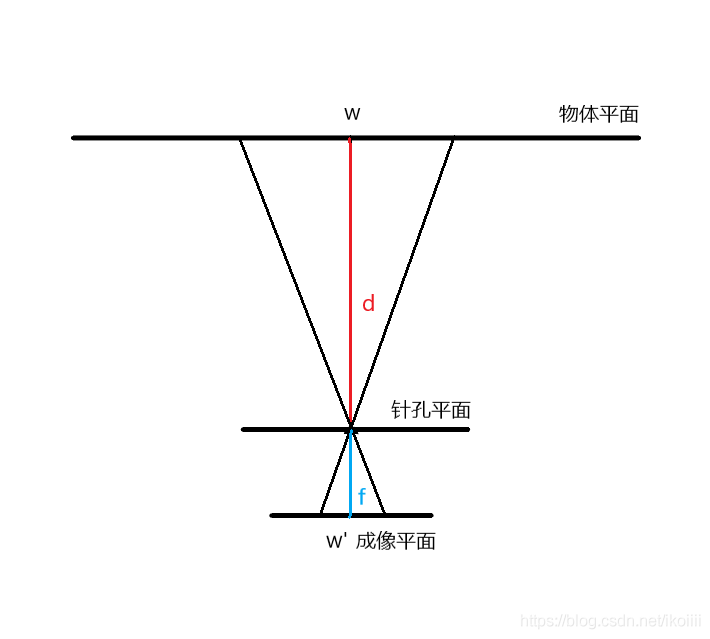
首先设定绿色的阈值上限和阈值下限。接着由BGR色彩空间转换到HSV色彩空间，便于进行颜色的处理和提取。然后将图像二值化，使绿色区域和非绿色区域分离。鉴于第一次拍摄时会有误识别现象发生，我们对二值化之后的图像进行腐蚀操作和膨胀操作，尽可能的去除噪点。接着寻找图像的所有轮廓并对各个轮廓进行遍历。紧接着便是对各个轮廓进行相应操作，画出矩形包围框，并通过矩形包围框的长宽比和面积进行目标轮廓的提取，最终返回目标轮廓的矩形包围框参数。

具体代码如下：





3．视觉测距：原理图如下



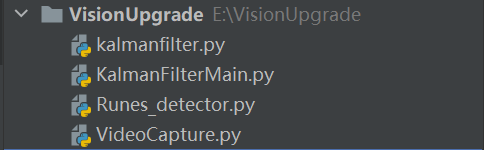
这是一个典型的小孔成像模型，与单目相机的成像原理类似。中间通过红蓝的垂线是相机的主光轴，d是被测物体至镜头的距离，f为相机镜头的焦距，w为被测物体的实际宽度（高度），w'为物体在成像平面（感光元件）上的宽度（高度）。 根据相似三角形公式可得：

**f / d = w' / w**

**摄像头焦距f可以通过公式测量出，已知目标物体的尺寸大小，确定好焦距后便可再利用公式测量出目标到摄像头的距离。具体代码如下：** 

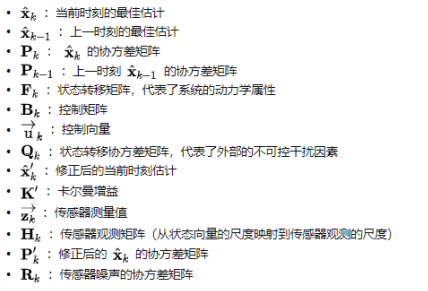
1. 行动预测：

利用卡尔曼滤波进行轨迹预测，共包含以下三个工程文件：



其中卡尔曼滤波的参数文件为1\_kalmanfilter.py，我们小组直接调用了封装好的kf库。首先设置转移矩阵维度为4，测量矩阵为2，以完成从(x,y,w,h)向量向预测位置（x,y）的计算。

在说明卡尔曼滤波的代码实现之前，先给出可能用到的参数：

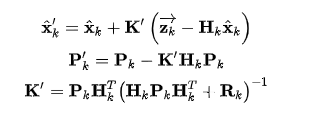


以下为卡尔曼滤波的预测方程：



在封装函数中，上一时刻是的最佳估计默认为第一帧的（x，y）坐标。一开始，我们首先设定了状态转移矩阵，但此处处于对计算效率的考虑，我们未设置方程的控制向量，因此默认为0，确定了系统的动力学属性，将当前预测值映射到未经修正的估计值。Kf库的一大优势就是帮助我们省略了复杂的矩阵变换和协方差计算，简化了运算。因此，封装函数帮助我们自动完成了协方差的计算。

以下为卡尔曼滤膜基于当前估计的修正过程：



在此，我们设定了传感器的观测矩阵（又称测量矩阵），从状态向量的尺度映射到传感器观测的尺度，在封装函数内部计算出了卡尔曼增益并与测量误差加权完成了修正当前估计的工作。

以下为在kf类中预测当前估计的代码实现：

