在提取了轮廓之后，首先要知道每个轮廓对应了魔方面上的哪个位置的小方格，之后要获得这个小方格的颜色，首先介绍定义的类：

class cubespot(object):

def \_\_init\_\_(self,x,y,cont,colour="",pos=-1,face=-1,mean=[0,0,0]):

self.colour=colour

self.x=x

self.y=y

self.cont=cont

self.pos=pos

self.face=face

self.mean=mean

主要有用的参数有pos，用序号1-9代表小方格的位置，self.mean是小方格hsv获得的平均值，用来进行颜色的计算，colour是计算得到的值。

首先要确定轮廓中的颜色

mask = np.zeros(gray.shape,np.uint8)

cv2.drawContours(mask,[cnt],0,255,thickness=cv2.FILLED )

这两行程序定义了一个和img图片大小一样的全0数组mask，之后用drawContours参数设置为cv2.FILLED将mask中识别出的“[cnt]”轮廓的内部填充，并画到mask上，此时mask为一张轮廓内为1轮廓外为0的数组，将这个数组与原图片相乘，可以得到原图片在对应位置的像素点，再采用

mean\_val = cv2.mean(hsv,mask=mask)

将像素点的hsv颜色的平均值取出并代入判断颜色的函数，获得返回值赋给colour

之后对9个小方块根据位置进行排序：

cube=sorted(cube,key=lambda one:(int(one.x/80),one.y))

首先对x做标排序，之后对y坐标排序，其中x不是准确度x值而是一个大概的先后范围（考虑到图片的旋转和噪声）

这样排序好后就可以知道轮廓对应的位置。

最后把获得的参数赋值到类中再作为返回值返回给调用的主函数