大作业：树莓派小车对操场跑道进行测距

1.简介：使用树莓派小车的摄像头（或者巡线模块），实现在循环在操场上行进，测算操场跑道。

2.设计方案：

使用小车以额定的速度的转速行进，测得运动的时间，就能轻易算出就是操场的具体长度.同时，使用拍取的照片来实现巡线。（来自于yzy\_1996的教程实现）

3.技术原理：

速度的获取：根据不同的跑道材质，通过超声波的测绘距离，使用一张挡板（或者自己的腿），就能通过一定的电压下，超声波测得的距离的值的变化速度，得出我们想要的小车的速度

但是考虑到速度的开始和结束时，存在加速度的变化，我们决定去掉这个部分，我们打算使用，在一次测速区间内多次在相同的时间间隔内取得超声波测得的值，存入同一个列表。在测速结束时，我们通过取平均值，获得一个较为准确的在特定材质路面下的平均速度。同时，让所有的数据留到最后处理，可以提高我们实验的准确性。

第三次测量

第二次测量

第一次测量

开始运动

结束

时间的获取：通过python中的time类中的time（）函数,我们读取系统自带的系统时间，通过两次系统时间的差，我们可以得整个小车运行的具体时间。

重点：

巡线的方式：我们通过摄像头拍照可以获取跑道的样子，并且进行数据分析，就能得出跑道中小车的正确的行进轨迹。如下，跑道中有白色的边框，而其他的部分我们都可以看成非白色的，这样很自然我们就可以将传入的图片的色素二值化，这里采用的是大津法二值化。

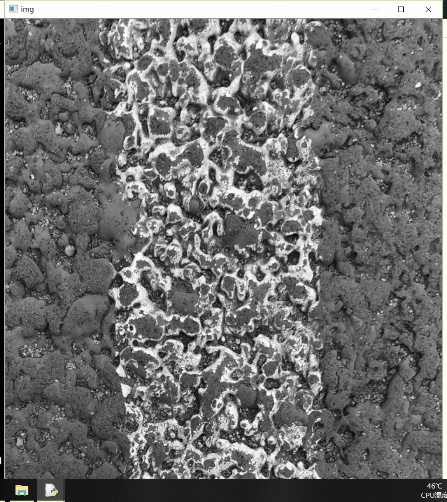
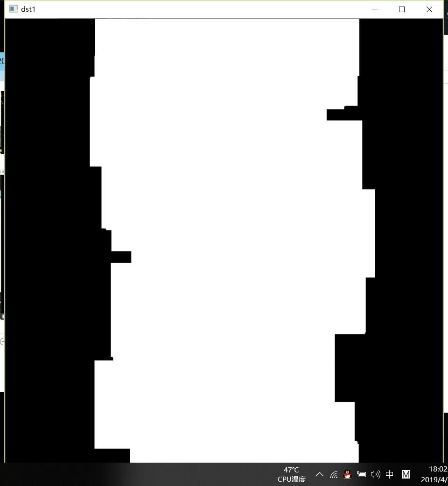
同时，是视觉利用Opencv来实现。我们不需要分析整个摄像头的范围，只需要取图片中的一行像素，就能进行分析了，这类似于线阵摄像，找到中间的白色的色素就好了。（作者：yzy\_1996, CSDN,链接：https://blog.csdn.net/yzy\_1996/article/details/85318179）

首先，调用cvtColor函数，使用bgr2gray的方式把将彩色图片变为灰度图。效果如下。

cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

接着使用cv2.threshold(gray, 0, 255, cv2.THRESH\_OTSU)cv库中的threshold调用大津法对图片进行二值化。于是乎，图片效果如下。

然后，我们使用膨胀和侵蚀，使用两次侵蚀，去除外部存在的小白点。接着使用两次膨胀，让白线的轮廓明显，效果如下。

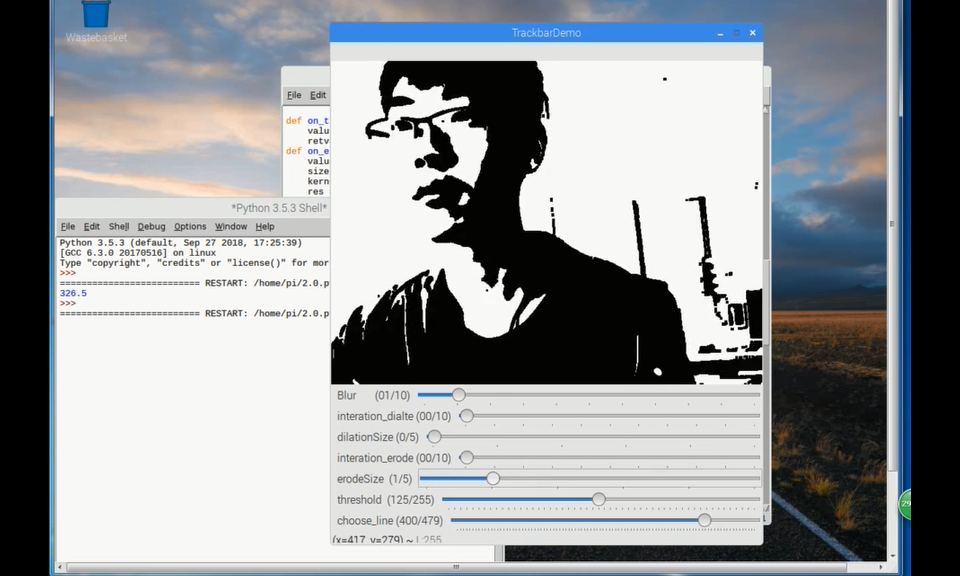


得到最后一张图以后，我们就可以获得这样的导航的效果。

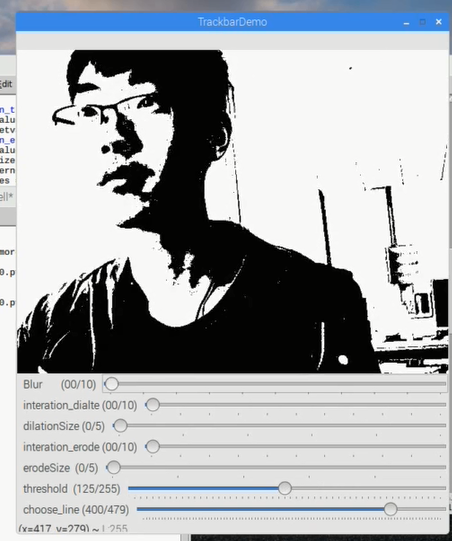
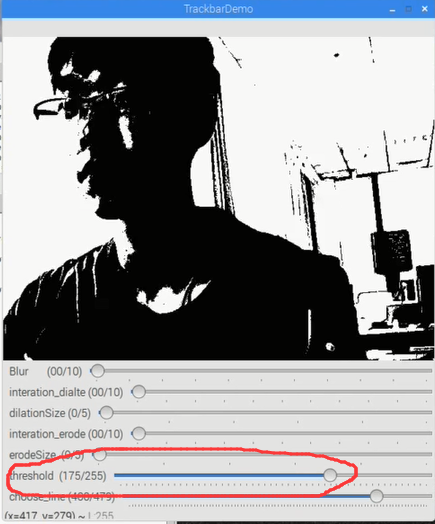
接着我们取300行的色素，找到白色素的第一个位置，接着找到白色素的最后一个位置，取中点，我们就可以找到轨迹的中线。如此，如果白色素小于中线320时，车实际上偏离轨迹向左。但是，实际操作时，只有小于200，我们再修正方向，让小车向右偏转即可。同理，大于400时，我们向左修正航向。

根据以上的原理，我们可以完成巡线的操作。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在实地测试中，我们发现由于场地曝光和种种环境原因的问题，仅靠小幅修改网上的案例难以满足项目需求。于是我们想到可以创建若干个滑动条，用可视化界面来调整二值化后的图像。

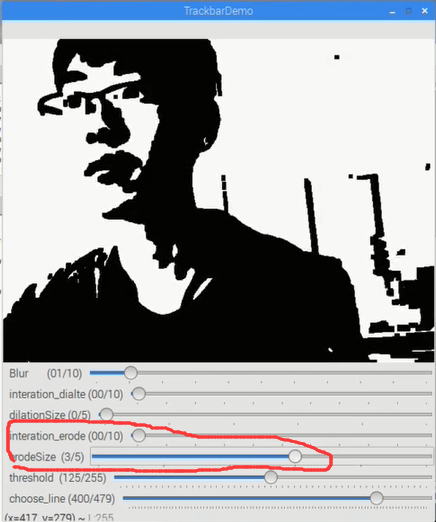
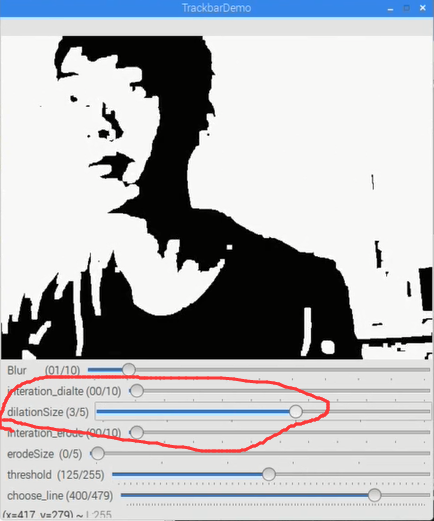
在将图像转换为灰度图后，先通过控制二值化时的阈值确定大致黑白分界图像，减弱甚至消除环境曝光变化所带来的影响；



再使用opencv库中的中值滤波器先对图像做一次降噪处理（通过控制中值降噪法所生效的单位大小来改变降噪强度），减少甚至完全消去噪点；



如果前面两步得到的图像还不足以满足需求，我们还提供了腐蚀和膨胀的调整（提供了控制迭代次数和控制核大小的滑动条）做进一步处理。



由于实际场地时的转弯灵敏度和转弯速度暂时还未确定，以及处于提高程序适用性的考虑，我们将转弯速度、偏航灵敏度以及所选取行数也通过滑动条的形式来控制，最大程度的保证最后调试时的方便性。

最后，根据速度和时间，就能算出距离。