# Rm视觉程序文档

目录

[Rm视觉程序文档 1](#_Toc17647377)

[1.环境与运行方法 2](#_Toc17647378)

[1.1环境及语言 2](#_Toc17647379)

[1.2运行方法 2](#_Toc17647380)

[2.程序方法介绍 3](#_Toc17647381)

[2.1小车识别程序介绍 3](#_Toc17647382)

[2.2大风车识别程序介绍 4](#_Toc17647383)

[2.3 线程与程序运行 6](#_Toc17647384)

[3.注意与补充 7](#_Toc17647385)

[3.1视频输入 7](#_Toc17647386)

[3.2串口输入输出问题 7](#_Toc17647387)

[3.3全局变量 7](#_Toc17647388)

[3.4 matplotlib显示的图像 7](#_Toc17647389)

[3.5现在依然存在问题 7](#_Toc17647390)

## 1.环境与运行方法

### 1.1环境及语言

1. 所用语言：python

2. 所用库：math，用于程序中数学计算，

serial，python的串口连接库，用于串口通信，

opencv3，用于视觉识别，

numpy，python科学计算库，可提升python程序性能，降低硬件需求，

time，python时间方法，可进行时间方面的获取。

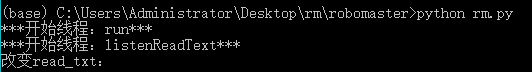
matplotlib. Pyplot，图像显示库，可方便快捷的显示图像（编写程序与调试使用，正式程序需要去除以提升性能）。

### 1.2运行方法

1.环境配置完毕后，从命令行进入robomaster目录下

2.目录下有测试视频（大风车的4个为官方视频，效果较好，小车的没有小车角度视频，用手机拍摄的1个暂时代替，效果较差）、程序文件（rm.py为最新整理的文件，逻辑思路比较清晰；rm\_old.py为之前的文件，串口方面的应用出现问题可以去该文件找找灵感）和该文档。

3.运行python rm.py 命令行可看到如下内容：



1. \*\*\*开始线程：run\*\*\* ——提示已经开启 识别线程
2. \*\*\*开始线程：listenReadText\*\*\* ——提示已经开启 数据监听线程
3. 改变read\_txt：——可键盘输入 “内容” 改变识别线程逻辑(当前视频识别完才会生效)。

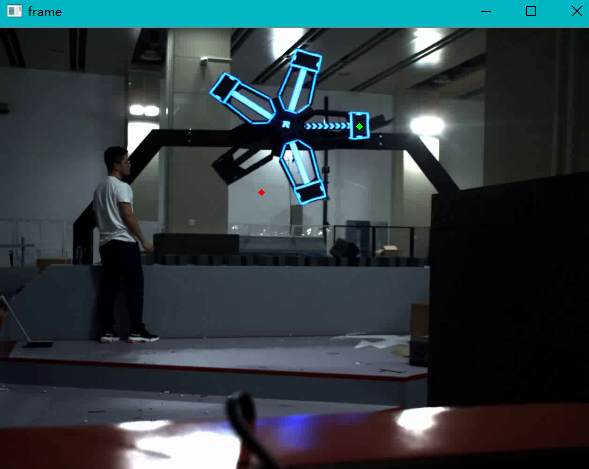
“内容”可为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 内容 | 小车/大风车 | 敌方颜色 | 跳帧 | 预测延迟(秒) (仅大风车生效) |
| 参数 | c/w | r/b | int | float |

例如： w,b,1,3.2, 为 大风车，蓝色，跳帧1，预测延迟3.2秒

c,r,3, 为 小车，红色，跳帧3

大风车新窗口可看到视频与标点，绿色点为当前打击点，红色点为延迟时间预测点。



小车新窗口会看到绿色打击点如：



## 2.程序方法介绍

### 2.1小车识别程序介绍

#### 2.1.1 car\_main(video,color\_flg,skip)

1.简介：小车识别程序主方法

2.入参：video，opencv读取的视频对象；

color\_flg，颜色标识（要识别的颜色；

skip，跳帧（因为板子性能跟不上，所以采用跳帧处理的方式减小延迟）。

3.出参：无

#### 2.1.2 find\_lamp(image,color\_flg)

1.简介：寻找疑似小车甲板的区域

2.入参：image，opencv读取的图片；

color\_flg，颜色标识（敌方颜色）。

3.出参：无

#### 2.1.3 largest\_rect(image, rect1, rect2)

1.简介：根据寻找到的两个疑似甲板灯画出甲板位置矩形，得出打击点位置

2.入参：image，opencv读取的图片；

rect1，寻找到的第一个灯坐标集；

rect2，寻找到的第二个灯坐标集。

3.出参：无（全局变量的形式更新小车打击点位置）

#### 2.1.4 ellipse\_distance(contours1, contours2)

1.简介：根据两个灯之间的预设比例，寻找疑似小车甲板的位置

2.入参：contours1，第一个疑似灯的位置；

contours2，第二个疑似灯的位置。

3.出参：权重值

#### 2.1.5 ellipse\_angle(contours1, contours2)

1.简介：根据两甲板灯的角度差，寻找小车甲板的位置

2.入参：contours1，第一个疑似灯的位置；

contours2，第二个疑似灯的位置。

3.出参：权重值

### 2.2大风车识别程序介绍

#### 2.2.1 wheel\_main(cap,color\_flg,delay\_time,skip)

1.简介：主运行方法

2.入参：cap，视频对象；

color\_flg，颜色标识（敌方颜色）；

delay\_time，延迟时间（用来生成预测点位置）；

skip，跳帧（因为板子性能跟不上，所以采用跳帧处理的方式减小延迟）。

3.出参：无

#### 2.2.2 get\_good\_img(frame,min\_hsv,max\_hsv)

1.简介：图像处理得到较好处理的二值图。

2.入参：frame，opencv读取的图像；

min\_hsv，hsv颜色范围的最小值；

max\_hsv，hsv颜色范围的最大值。

3.出参：处理好的numpy.array格式的图像。

#### 2.2.3 get\_center\_point(box,wheel\_center\_point,center\_point)

1.简介：获取打击中心点。

2.入参：box，打击臂box；

wheel\_center\_point，打击臂中心；

center\_point，风车中心。

3.出参： f\_point，打击点

#### 2.2.3 get\_wheel\_main(wheel\_cnt,wheel\_center,get\_num,contours,center\_point,y)

1.简介：获取被打击臂极其信息。

2.入参：wheel\_cnt，储存搜寻到的五个打击臂box；

wheel\_center，打击臂中心；

get\_num，获取到的打击臂计数，

contours，该帧图像所有待搜寻轮廓（从中搜索打击臂）；

center\_point，风车中心（更新）；

y，整幅图像高度（选区图像1/3以上的区域，以排除测试视频中的反光问题）。

3.出参： wheel\_cnt，储存搜寻到的五个打击臂box；

wheel\_center，打击臂中心；

get\_num，获取到的打击臂计数，

np.int0(center\_point)，风车中心（更新）；

max\_idx，最新应打击点下标。

#### 2.2.4 get\_anticipation\_point(origin\_point,delay\_time,center\_point)

1.简介：根据圆方程获取预测点的位置，实际因为程序处理延迟、子弹飞行延迟等问题，应该瞄准预测点。

2.入参：origin\_point，原打击点；

delay\_time，延迟时间；

center\_point，风车中心。

3.出参： anticipation\_point，预测点(x,y)。

### 2.3 线程与程序运行

#### 2.3.1 CreateThread类，继承threading.Thread

##### 2.3.1.1 \_\_init\_\_( self, threadID, name, defName)

1.简介：构造方法

2.入参：threadID, 线程id

name, 线程名称

defName ，方法名

3.出参：无

##### 2.3.1.2 run(self):

1.简介：运行线程

2.入参：无

3.出参：无

#### 2.3.2 run()

1.简介：识别程序线程入口

2.入参：无

3.出参：无

#### 2.3.3 listenReadText()

1.简介：数据监听线程入口,通过改变全局变量read\_txt，控制识别程序线程。

2.入参：无

3.出参：无

#### 2.3.4 runCarOrWheel()

1.简介：根据输入内容判断是运行小车识别还是大风车识别。

2.入参：无

3.出参：无

## 3.注意与补充

### 3.1视频输入

切换视频文件在runCarOrWheel()方法中，如果调用摄像头，将video = cv2.VideoCapture("XXX.XXX ")改为video = cv2.VideoCapture(0)即可（调用编号为0的摄像头）。

### 3.2串口输入输出问题

由于在自己电脑上调试时，无法利用串口相关内容，而且串口输入输出非常简单。所以为了便于修改和调试，我将串口相关内容全部注释或删除了，如需添加，可从网上查询。

如：<https://www.cnblogs.com/yaner2018/p/9818066.html>

<https://blog.csdn.net/as472780551/article/details/79126927>

等等。

### 3.3全局变量

1.point：存储最终打击点或预测点信息，格式：大风车w或小车c标识:x点坐标,y点坐标

如：w:100,120、c:320:311

2.read\_txt：用于监听键盘输入或串口输入，以改变识别程序所在线程状态。

共四个参数，后面各跟一个英文逗号

第一个参数 c:小车程序 w:大风车程序

第二个参数 b:蓝色 r:红色

第三个参数 跳帧

第四个参数 延迟时间 （只有大风车需要加，用来确定预测点位置）

若 read\_txt 第一个参数为f则退出程序

如： w,b,1,3.2, 为 大风车，蓝色，每1帧处理一次，预测延迟3.2秒

c,r,3, 为 小车，红色，每3帧处理一次

### 3.4 matplotlib显示的图像

matplotlib. Pyplot显示的图像最后应用时都要去掉，因为显示图像会占用大量内存、显存等，程序的预测点或打击点的识别不需要视频的显示，从而提升性能。

### 3.5现在依然存在问题

1.反光问题：地面、墙体的反光难以处理，需想办法改进。

2.运动模糊：当小车或者大风车运动时，非专业运动相机拍摄的画面会产生运动模糊，如有可能先进行运动模糊去除识别效果会大大增加。解决方法更换更好的摄像头或者添加运动模糊去除算法。

3.程序测试数据较少，还需大量现场或仿现场数据进行测试改进，提升程序健壮性和稳定性。