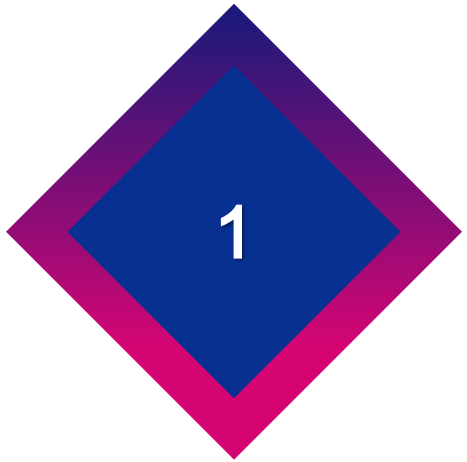
共创AI·耀星际

运动控制与无人驾驶

TAISTAR
钛创星

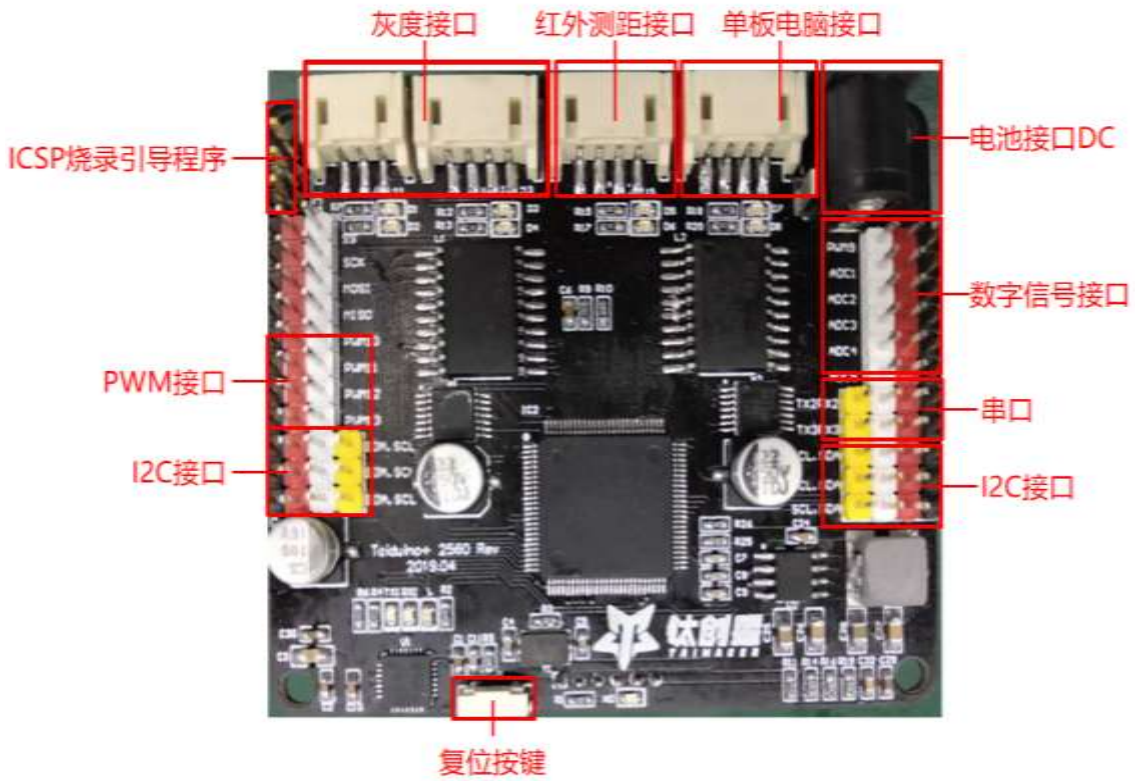
无人驾驶



电子元器件简介

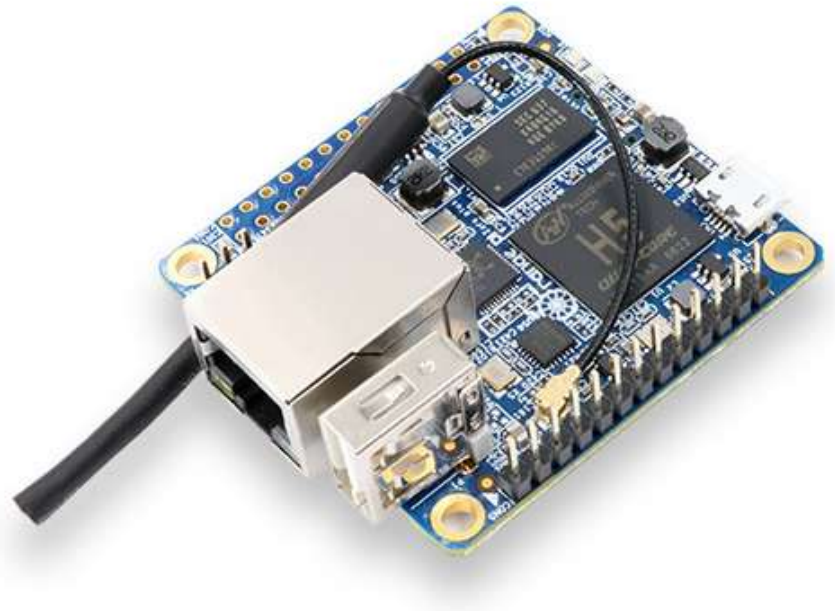
主控板

主控板是机器的控制器，上面有各种各样的端口，既可以接收外界信息，又可以发出控制指令。它的工作是：接收信号，处理信号，发出信号。



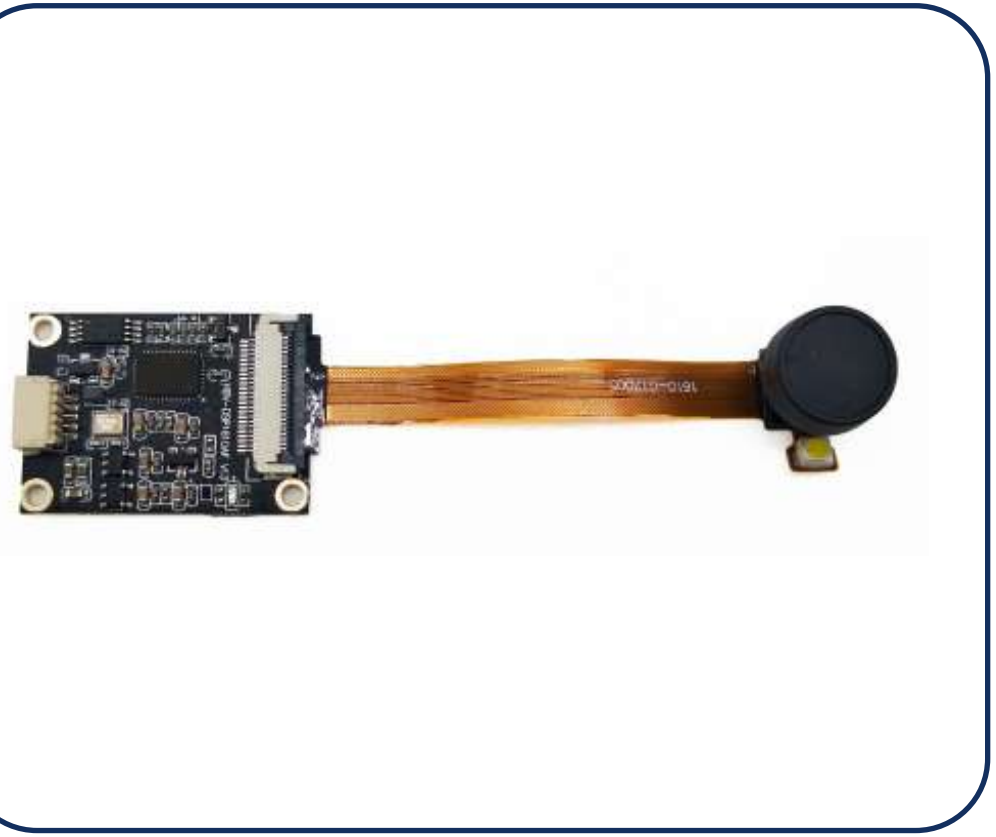
单板电脑

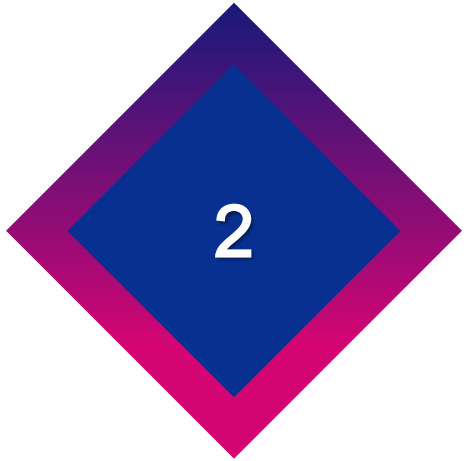
单板电脑就是一台微型电脑，智能车所使用的单板电脑为多核处理器，其中搭载了linux操作系统，用于运行视觉处理的python程序。



摄像头

摄像头使用的是角度为120°的广角USB摄像头，通过USB接口可以直接连接到单板电脑并将图像信息回传给单板电脑。





视觉识别

无人驾驶——视觉识别

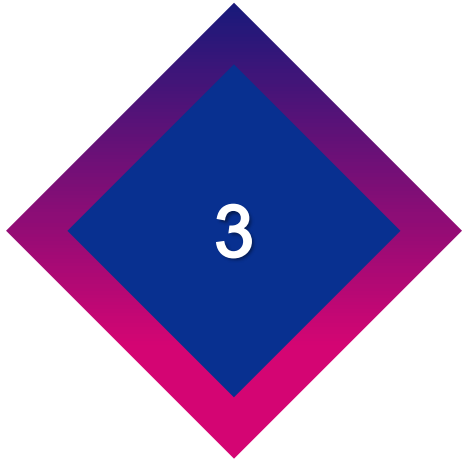
什么是视觉识别?

01 顾名思义就是计算机通过摄像头（视觉）识别、获取外界的实时图像。

02 随后利用单板电脑对获取到的图像进行关键信息提取、筛选、处理分析等。

03 最终将分析过后的图像传输给主控板，主控板则根据接收到的图像进行判断、发布后续指令。

04 探索者智能车就是利用视觉识别去进行车道线检测的。



车道识别

浅析车道识别

通过视觉识别判断智能车是否偏离白色车道线，如果偏离车道线就控制智能车及时进行修正，保证智能车能够一直在白色车道线内行驶，从而实现车道识别的目的。

在行驶过程中，我们规定两个方向，一个是智能车行驶时车头的方向、一个是白色车道线的方向。

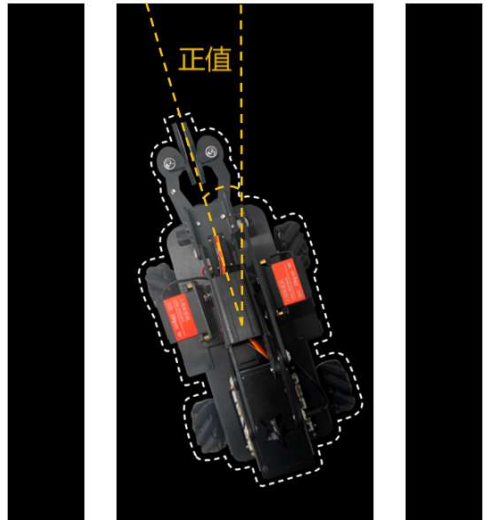
无人驾驶——车道识别

如果智能车车头方向与车道线方向一致，即为正常行驶。

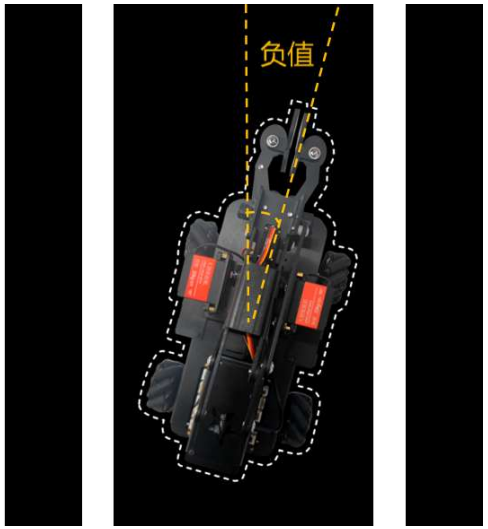
如果车头方向与车道线方向不一致，即为偏离路线 。

无人驾驶——车道识别

如果车头向左偏转，则在摄像头的第一视角中，车道线在车头的右侧，二者产生了右偏差角度，需要智能车向右转向才能回到直线上。



如果车头向右偏转，则在摄像头的第一视角中，车道线在车头的左侧，二者产生了左偏差角度，需要智能车向左转向才能回到直线上。



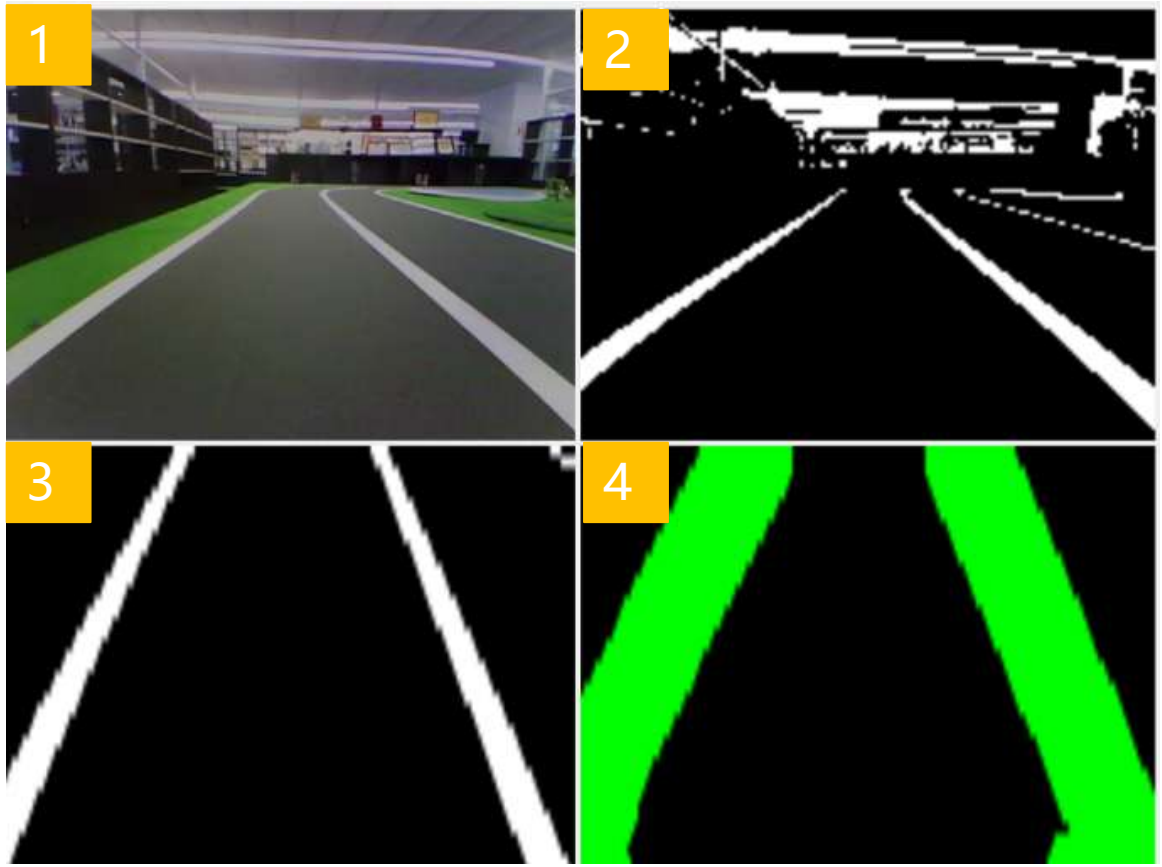
车道识别原理

无人驾驶——车道识别原理

通过视觉巡线进行车道线检测，需要完成以下步骤（Opencv）：

01 安装在智能车模型上的摄像头捕捉到的画面称为第一视角画面，图中1号图像为原始图像。

02 **二值化：**首先对图像进行灰度处理，我们可以设定一个阈值，当灰度值高于阈值时，该点设为白色；当灰度值低于阈值时，该点设为黑色。
通过对阈值的调整，可使图像变为二值化图像，即黑底加白色车道线的图像，图中2号图像为二值化处理后的图像。

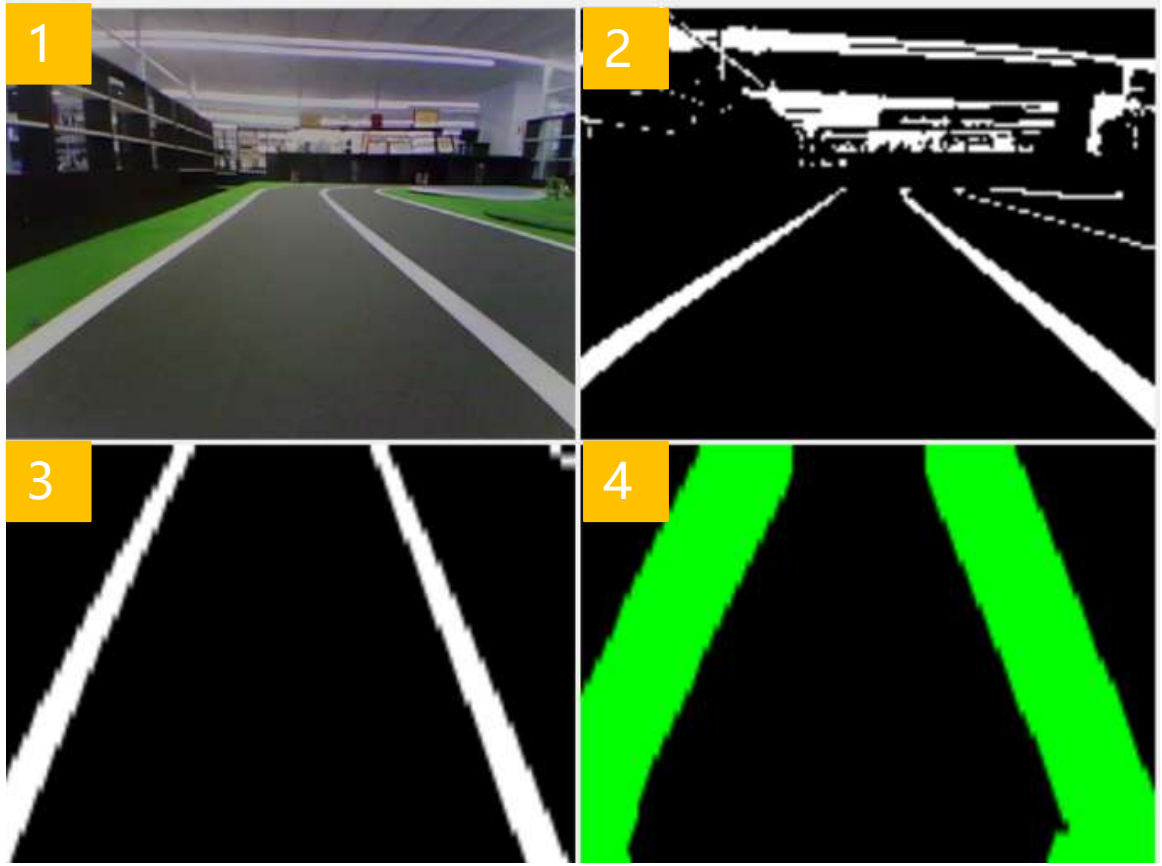


无人驾驶——车道识别原理

通过视觉巡线进行车道线检测，需要完成以下步骤（Opencv）：

03

图像裁剪：在二值化处理后的图像中，一半为道路画面一半为前方环境画面。图像处理仅需要道路画面，因此只需截取道路画面即可，其他画面可省略，从而减少图像中的干扰，图中3号图像为图像剪裁后的画面。



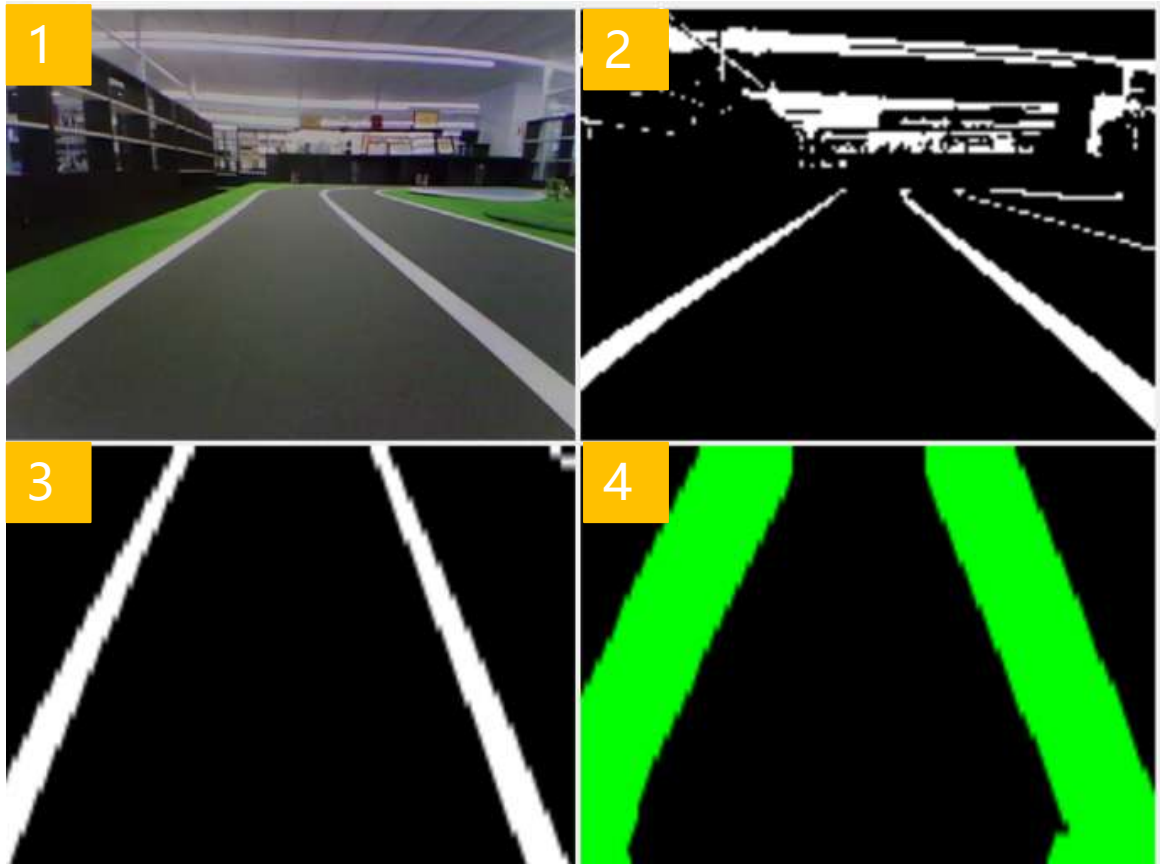
无人驾驶——车道识别原理

通过视觉巡线进行车道线检测，需要完成以下步骤（Opencv）：

04

图像滤波：剪裁后的图像中可能存在一些白色斑点，斑点由于坐标点位置的灰度值与白色赛道线相似，因此无法用二值化消除。

我们对二值化后的图像进行腐蚀，可使用高斯模糊、中值滤波、双边滤波（去除噪点的方式）等去除出两条赛道以外的白色斑点，去除噪点后我们可以看到的是两条清晰的没有斑点的白色赛道线。

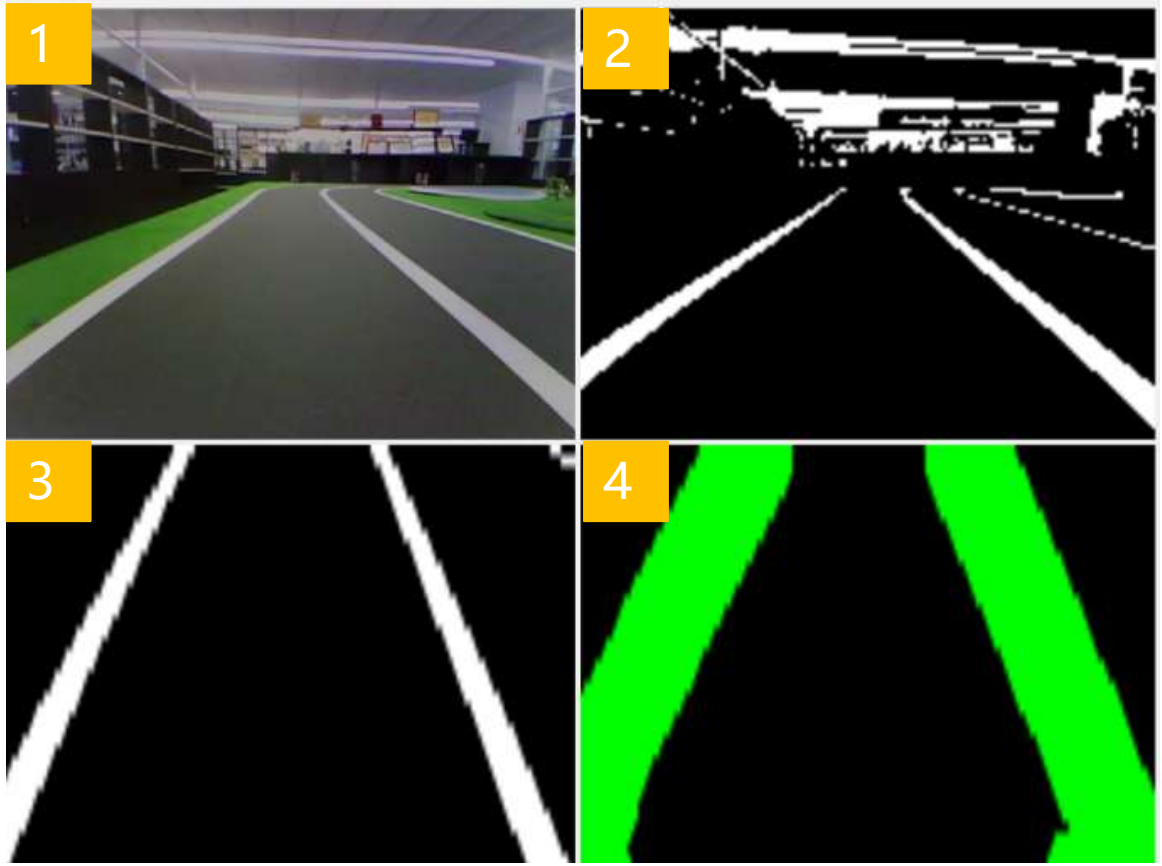


无人驾驶——车道识别原理

通过视觉巡线进行车道线检测，需要完成以下步骤（Opencv）：

05

边缘检测：滤波后的图像已经可以进行数据分析了，但此时状态能获取到的信息依然只是图像中每个坐标所对应的HSV值（色调、饱和度、明度）。因此我们需要提取各个坐标点之间的联系，此时可以使用canny变化（Opencv中自带的一种边缘检测方式）将两条直线提取出来。

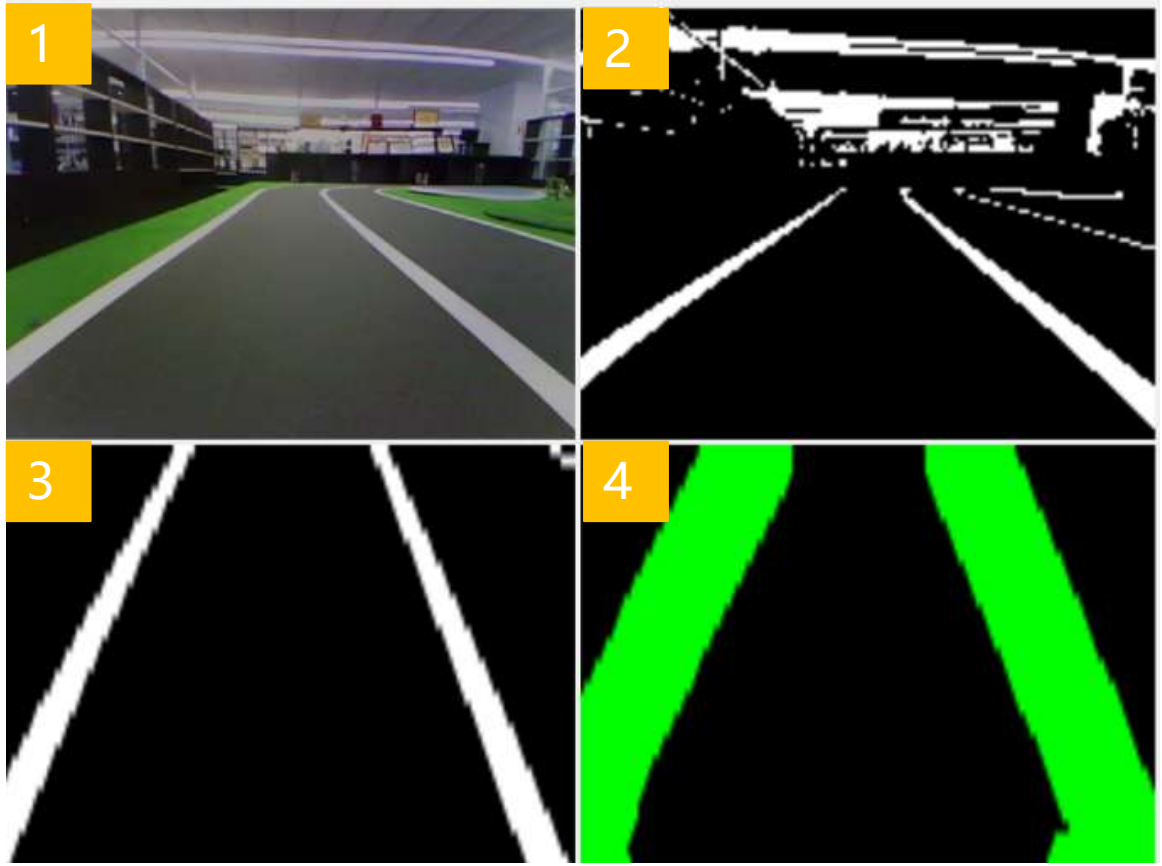


无人驾驶——车道识别原理

通过视觉巡线进行车道线检测，需要完成以下步骤（Opencv）：

06

直线检测：边缘检测将获取车道检测所需的两条直线，并获取车道线的斜率，根据该斜率，即可判断智能车模型与车道之间的偏差角度。两条车道中某一条斜率增大，则表示智能车模型偏向该侧车道，此时需向控制电机向另一边转向。图中4号图像为直线检测后的图像。



无人驾驶模块介绍

视觉识别车道偏差角度模块

视觉识别车道偏差角度

- 该模块在【钛星库】——【探索者×1】中。
- 该模块代表摄像头识别车道后回传给主控板的车头与车道线间的偏差角度。
- 正值代表探索者向左偏转，产生了右偏差角度，负值代表探索者向右偏转，产生了左偏差角度，数值越大代表偏差角度越大
- 该模块为视觉模块，因此上传至智能车后，智能车会有一段等待启动的时间，大约30~50秒。启动过程中两侧灯带会一直闪烁，当启动完成后，智能车播报语音“摄像头已启动”，两侧灯带会常亮1秒后熄灭，表示启动完成。

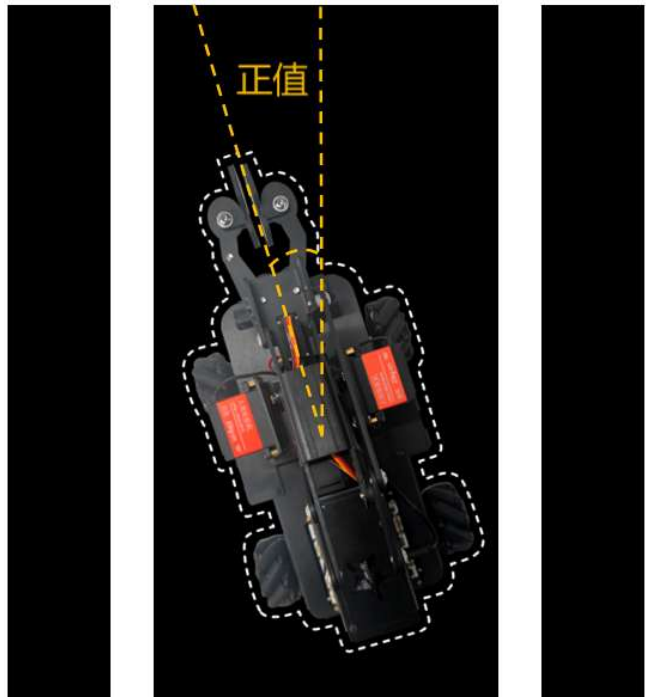
任务解析

无人驾驶——任务解析

以“视觉识别车道偏差角度”模块获取到的转向角度作为参数依据，来修正两侧电机的转速。

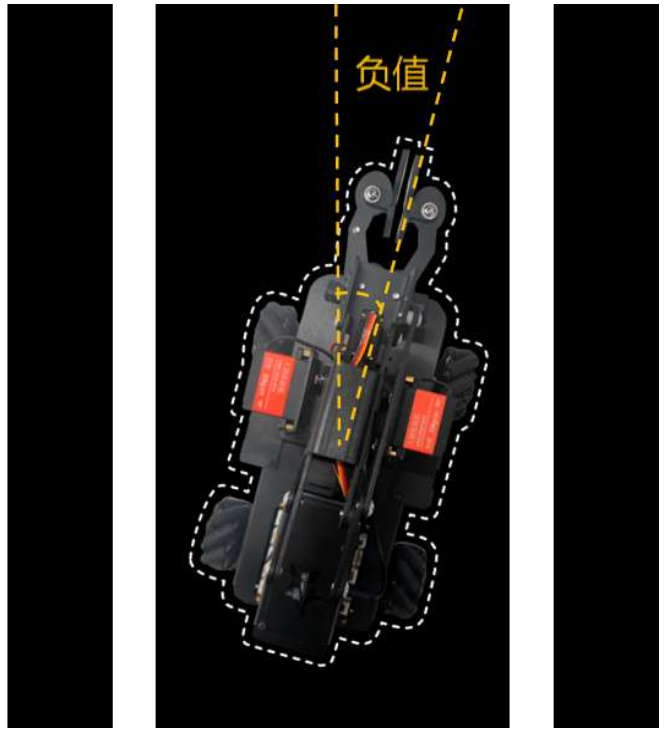
无人驾驶——任务解析

当“视觉识别车道偏差角度”模块检测到智能车模型向左偏转时，则在摄像头的第一视角中，车道线在车头的右侧，二者产生了右偏差角度，需要加快左侧电机转速，减慢或维持右侧电机转速，使智能车模型向右转向，从而修正智能车的偏离路线。



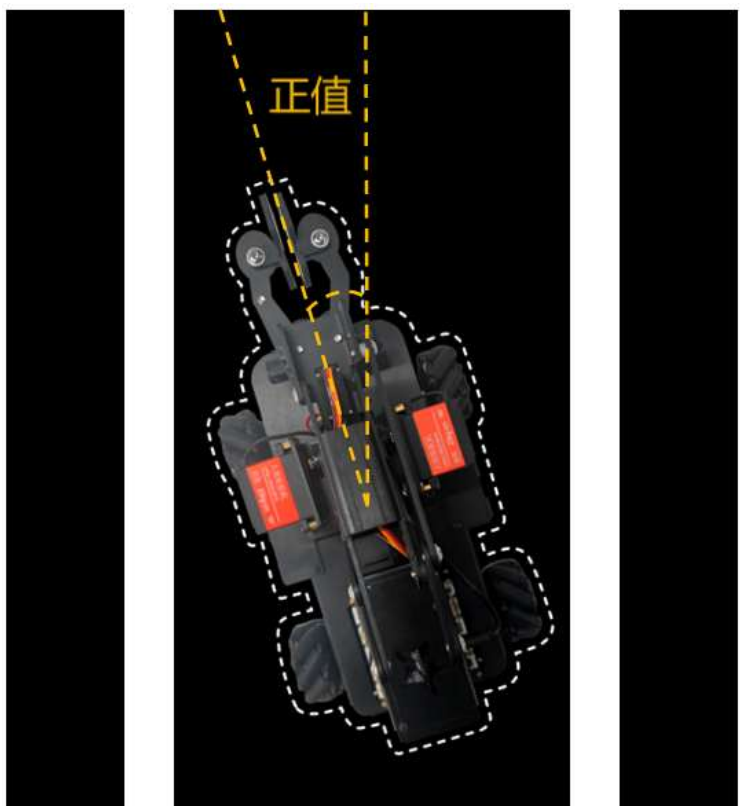
无人驾驶——任务解析

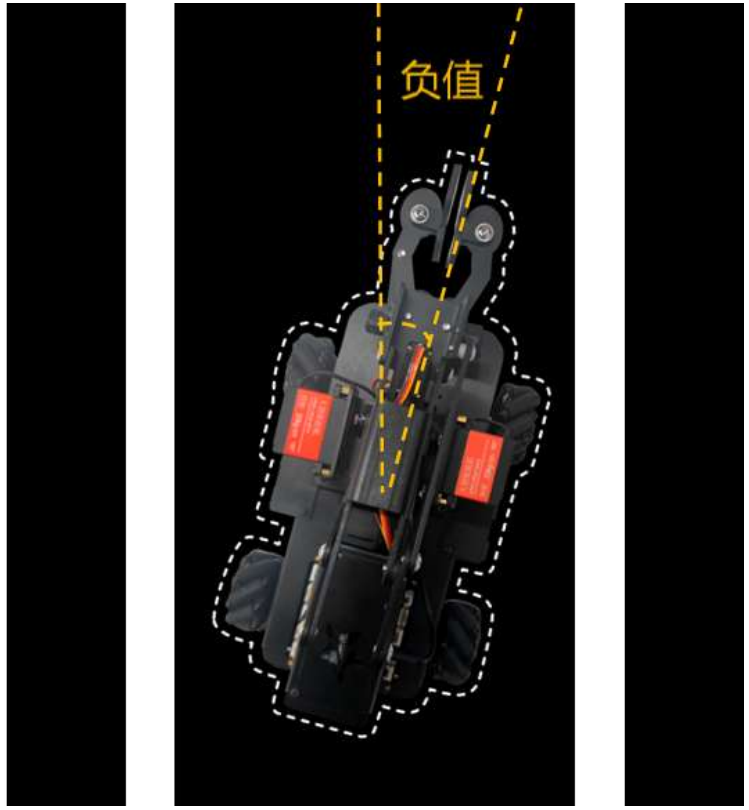
当“视觉识别车道偏差角度”模块检测到智能车模型向右偏转时，则在摄像头的第一视角中，车道线在车头的左侧，二者产生了左偏差角度，需要加快右侧电机转速，减慢或维持左侧电机转速，使智能车模型向左转向，从而修正智能车的偏离路线。



无人驾驶——任务解析

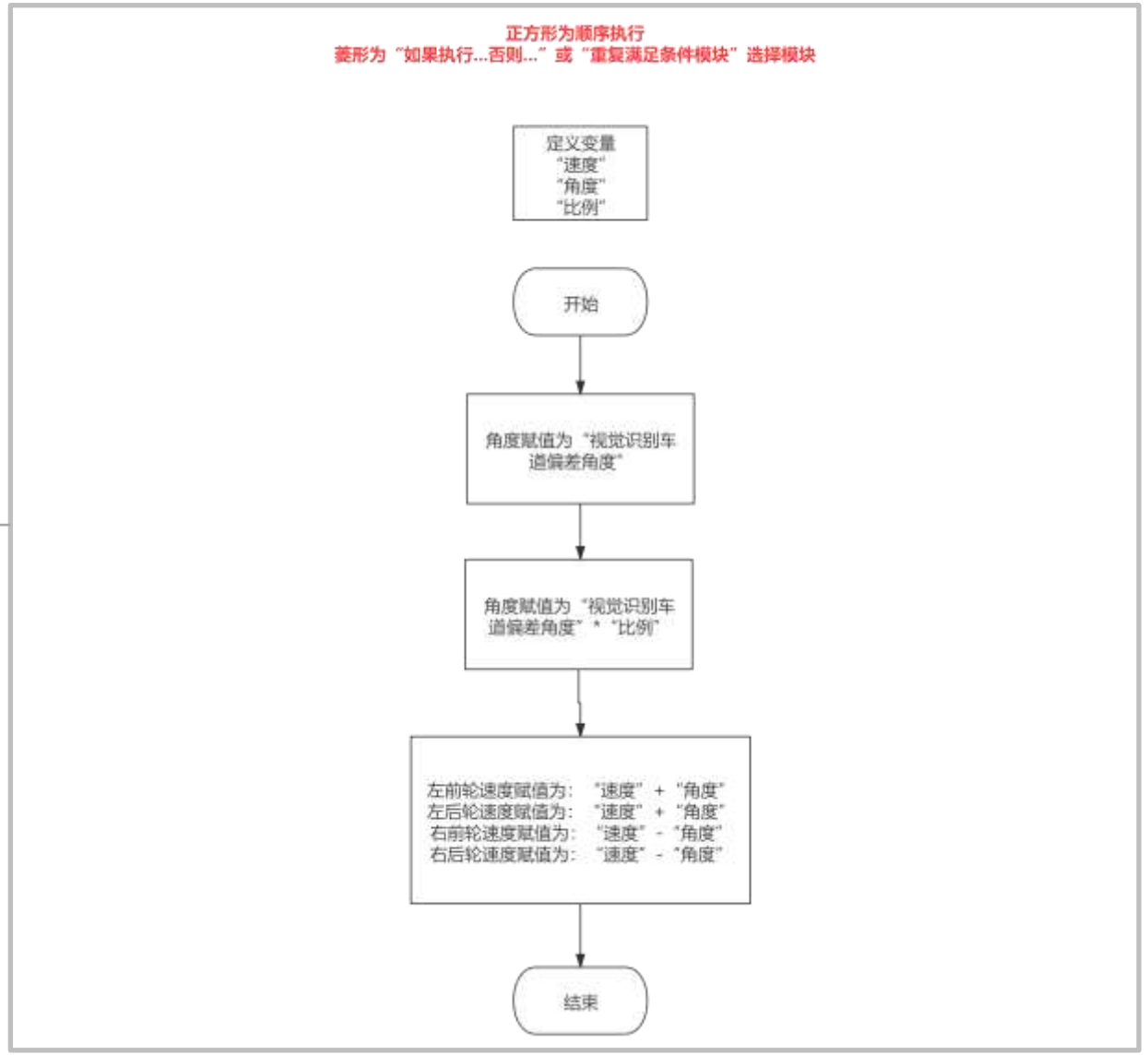
注意：偏离量越大，两侧电机转速差值需要越大，否则智能车模型无法修正偏离量。





无人驾驶——任务解析

任务流程图

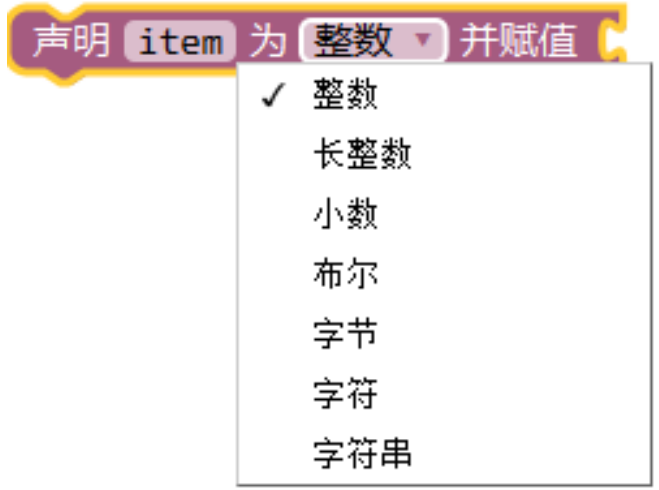


无人驾驶——任务解析

除了“视觉识别车道偏差角度”模块，想要完成任务，还须使用以下模块

声明变量模块

- 该模块在【变量】功能中。
- 该模块给一个变化的量起一个固定的名字并且定义一个类型（整数、小数、长整数等），通过下拉菜单可选择变量类型：



声明变量模块

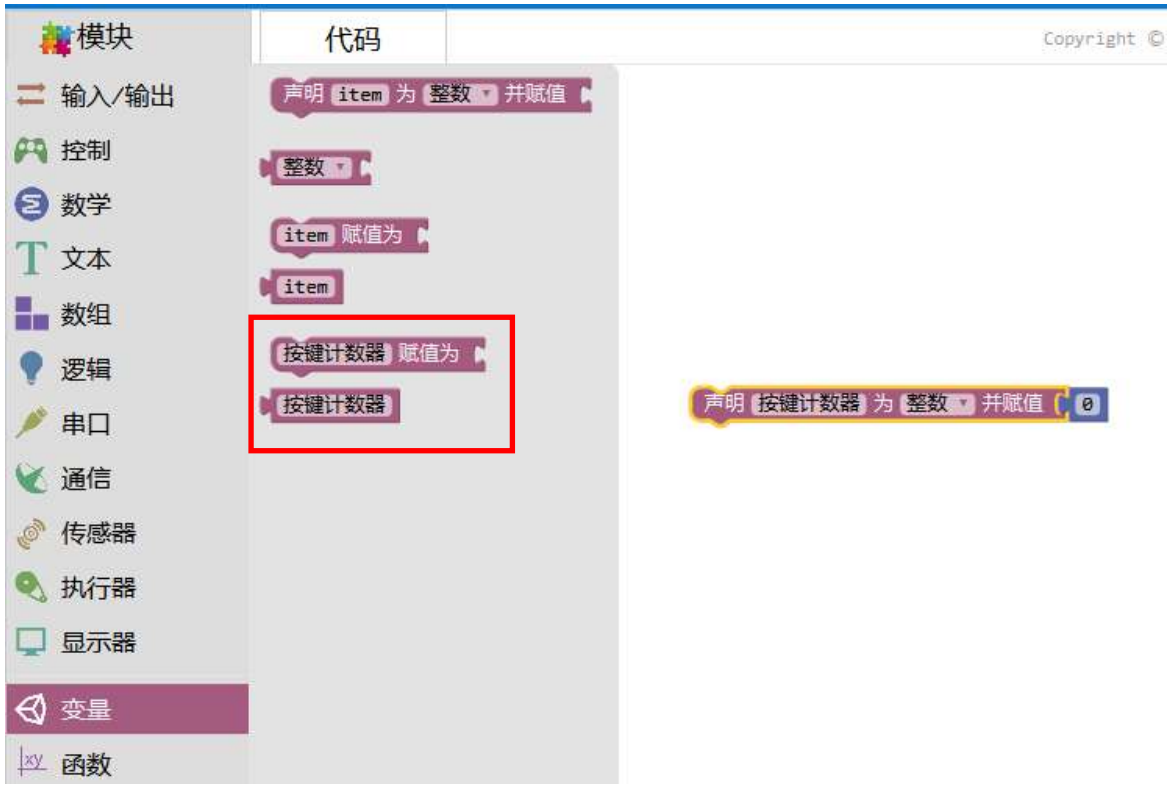
- 每个类型都有一定的数值范围，变量的数值可在此数值范围内变化。
- 定义变量后在 变量 功能列表中会多出两个模块，一个为“赋值为”模块，可通过该模块修改相应变量的数值；另一个为“嵌入”模块，可通过该模块获取变量的数值。

声明变量模块

例：假设我们在程序中要使用到“按键计数器”这个变量，那么在程序最初先要对其进行声明，即给“按键计数器”这个变量取个名字，并选择其所属类型（例子里选择类型为整数），一般初始数值都为0：

声明变量模块

声明“按键计数器”这个变量后，在 变量 功能列表里就会出现对应的变量。如下：

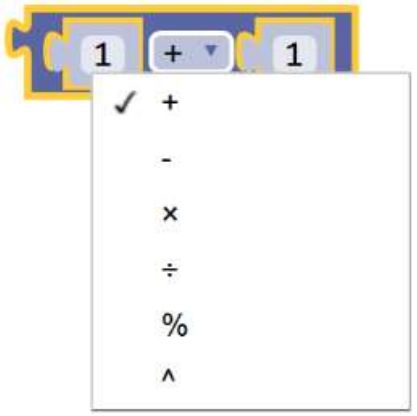


数字模块

- 该模块在【数学】功能中。
- 该模块会提供一个数值，可作为其他模块的参数或条件

运算模块

- 该模块在【数学】功能中。
- 该模块能够进行“+” “-” “*” “÷” “%”取余、“^”次方运算。运算结束后将结果数值赋值给所需要的内嵌模块处。



程序解析

无人驾驶——程序解析

首先定义“速度”、“角度”、“系数”三个变量，并给“速度”和 ”系数“赋一个初始值（初始值需要自行调节）

第二、我们需要为“角度”赋值，“角度”赋值为“视觉识别车道偏差角度”

无人驾驶——程序解析

第三、我们分别给左前轮、左后轮速度赋值为：

右前轮、右后轮速度赋值为：

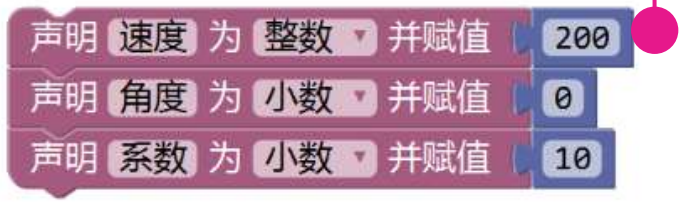
通过偏差角度角度*系数，可以扩大两侧轮子的差速，增加智能车的转向效果。（系数就像是车子的方向盘，打得越多修正的越多，这个系数需要自行调节，若智能车在巡线时左右摇摆可将系数调小，若智能车在巡线时看到弯路直接冲了出去可将系数调大）

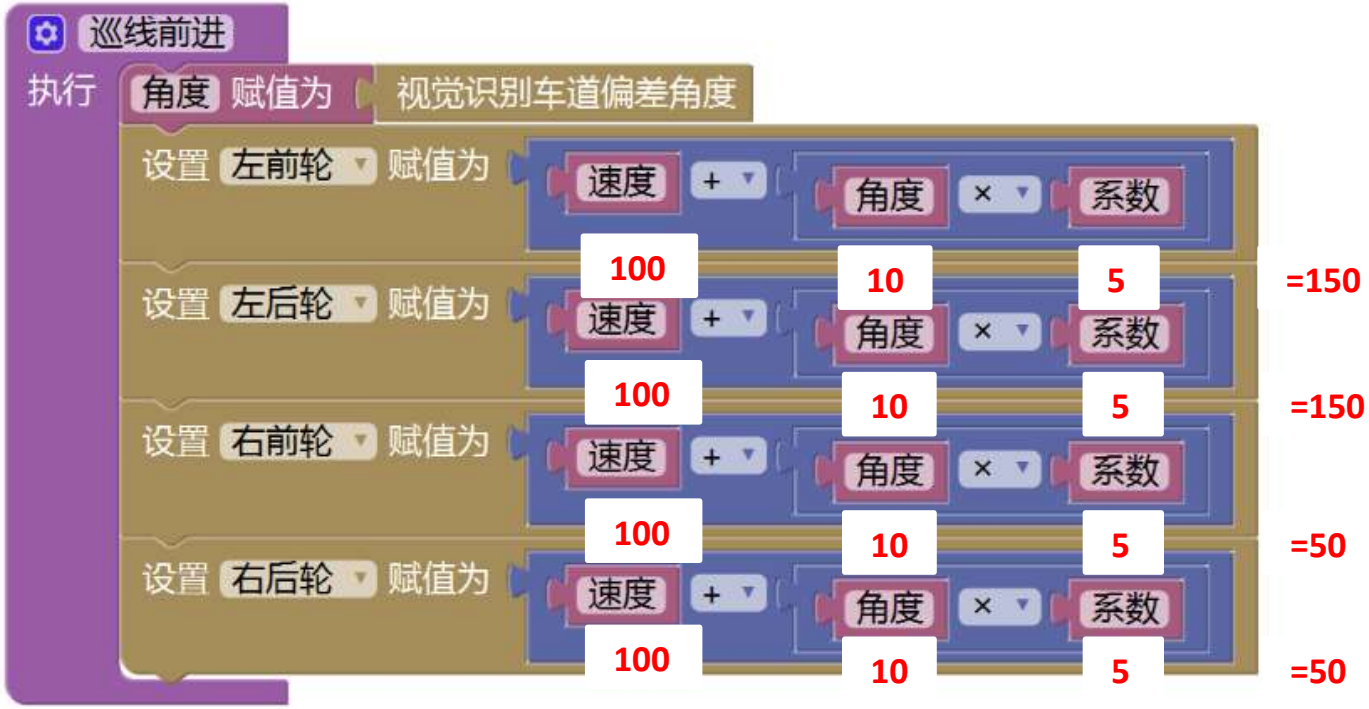
如果车头向左偏转，则在摄像头的第一视角中，车道线在车头的右侧，二者产生了右偏差角度（正值），需要智能车向右转向才能回到直线上，即左侧轮子速度大于右侧轮子速度。

如果车头向右偏转，则在摄像头的第一视角中，车道线在车头的左侧，二者产生了左偏差角度（负值），需要智能车向左转向才能回到直线上，即右侧轮子速度大于左侧轮子速度。

无人驾驶——程序解析

例1：速度为100，向右偏差角度为10，将这两个数字带进程序里，可发现车轮向右转来修正方向。

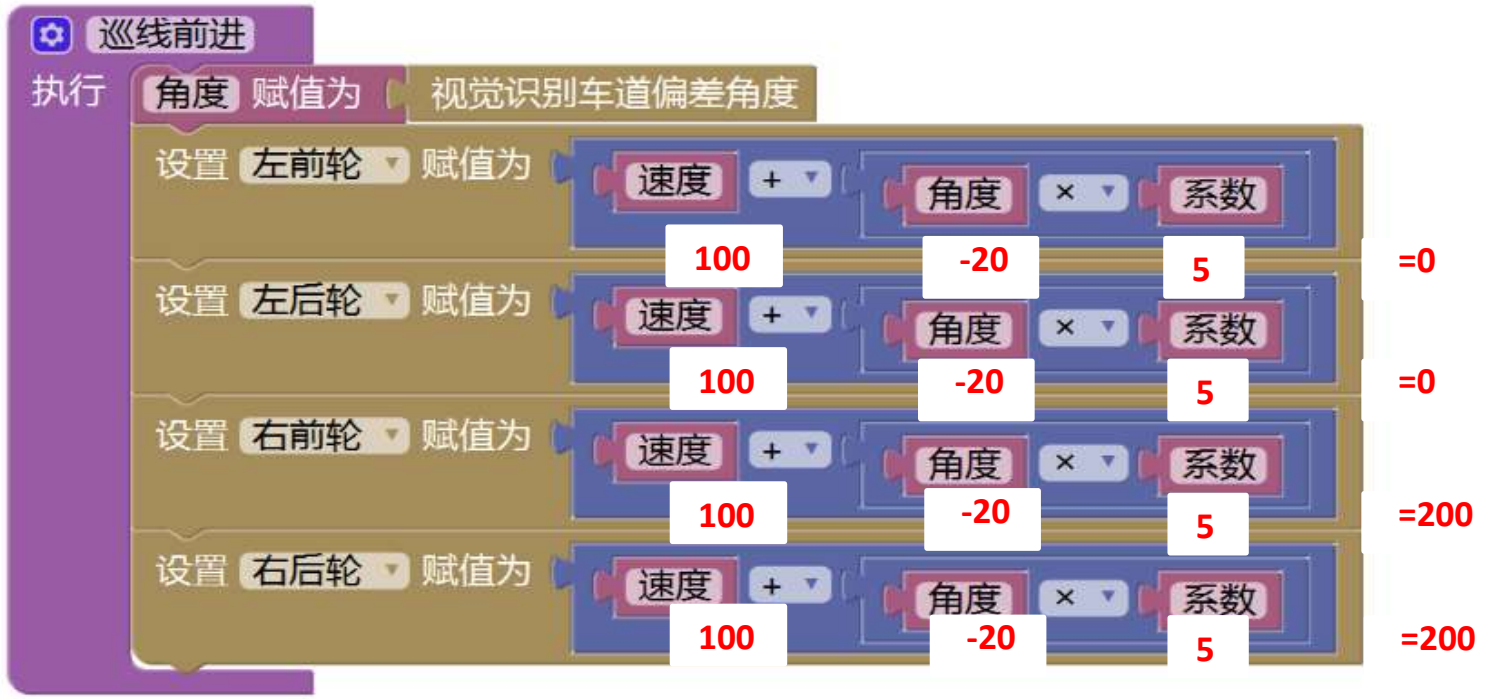




左轮速度大于右轮速度，向右转

无人驾驶——程序解析

例2：速度为100，向左偏差角度为-20，将这两个数字带进程序里，可发现车轮向左转来修正方向。

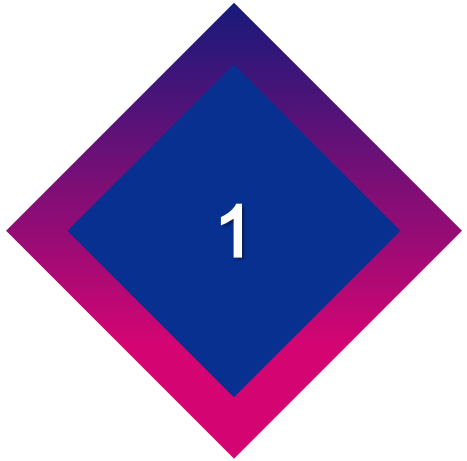


右轮速度大于左轮速度，向左转

无人驾驶——程序解析

参考程序

固定距离移动



模块介绍

“获取前进距离”模块

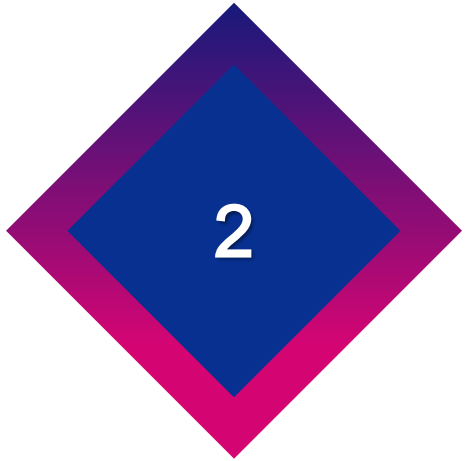
- 该模块位于【钛星库】——【探索者×1】中。
- 该模块可返回当前智能车向前/向后移动的距离，通过获取智能车的移动距离，判断是否到达任务中需要的距离，并执行相应任务，向前移动距离为正值，向后移动距离为负值，单位cm。
- 返回的距离从智能车启动开始计算或在使用“清空移动距离”模块后开始计算的智能车向前移动距离（包含转向时向前的距离）。

“获取水平距离”模块

- 该模块位于【钛星库】——【探索者×1】中。
- 该模块可返回当前智能车水平方向上的平移的距离，通过获取智能车的移动距离，判断是否到达任务中需要的距离，并执行相应任务，向右平移距离为正值，向左平移距离为负值，单位cm。
- 返回的距离从智能车启动开始计算或在使用“清空移动距离”模块后开始计算的智能车水平方向上的平移的距离。

“清空移动距离”模块

- 该模块位于【钛星库】——【探索者×1】中。
- 该模块用于清除当前已经记录的距离数值，将所有距离数值归0。
- “获取前进距离”、“获取水平移动距离”均可用“清空移动距离”做清0，一旦使用该模块，则之前获取的“水平移动距离”、“前进距离”均清0。
- 在获取距离之前，最好都要清空移动距离，方便智能车测量不同任务点中的距离。

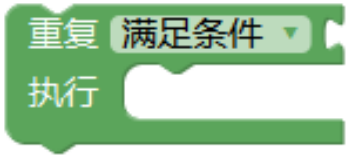


综合实践

固定距离移动——综合实践

任务介绍：利用“获取前进距离”模块，使智能车巡线前进100厘米（1米）的距离后停车。除了 获取前进距离(厘米) **还需认识以下新的模块。**

“重复...满足...”模块



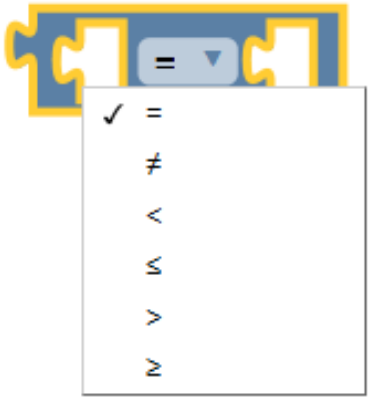
- 该模块在【控制】功能中。
- 该模块是用于实现“满足条件”或“不满足条件”的循环结构，单击参数中的下拉菜单箭头可选择“满足条件”或者“不满足条件”。

- “满足/不满足条件”后可添加一个条件值。例如，当“满足条件”为真时，即条件成立，则执行循环体中的模块；当“满足条件”值为假时，即条件不成立，则跳出循环中的模块，结束循环。

- 在知道判断条件的情况下可使用该模块，注意确保执行中的内容可以改变条件中的内容，并能跳出循环。

“条件判断”模块

- 该模块位于【逻辑】中。
- 该模块能够实现两个数值之间的比较，用来判断两个数值是否相等、哪个数比较大、哪个数比较小等等。
- “单击模块中的下拉菜单箭头可选择“=”“≠”“>”“<”等运算符号：



固定距离移动——综合实践

示例程序：

该程序为智能车巡线前进100厘米（1米）的距离后停车的程序。获取前进距离并判断是否大于100厘米，若大于100厘米，则将所有速度调至0，即停车。