# 小车控制系统文档

tandf*

2019 年 6 月 21 日

## 1 各部分功能介绍

本项目将 Vicon 系统（用于定位）、蓝牙小车进行了封装，并在此基础上定义了定位系统、小车、控制器三个对象，并开发了实时显示车辆轨迹、鼠标点击设置目标位置等功能，并能提供运行中产生的定位数据用于后期分析，为两轮小车控制系统的测试与开发提供了简单易用的平台。

为便于系统仿真与 Vicon 定位系统数据读取，项目使用 Matlab 开发。

项目主要分为三个模块：

- Vicon 定位系统接口，代码见 *Matlab/VData.m*
- 小车控制接口，代码见 *Matlab/Car.m*
- 控制器模块，代码见 *Matlab/Controller.m*

模块间的关系如图1所示。给定目标位置后，控制器将结合 Vicon 定位系统提供的实际位置与障碍物位置计算出小车两车轮的转速，并使用小车控制接口将指令发送至小车，控制小车到达指定位置。

*https://github.com/tandf

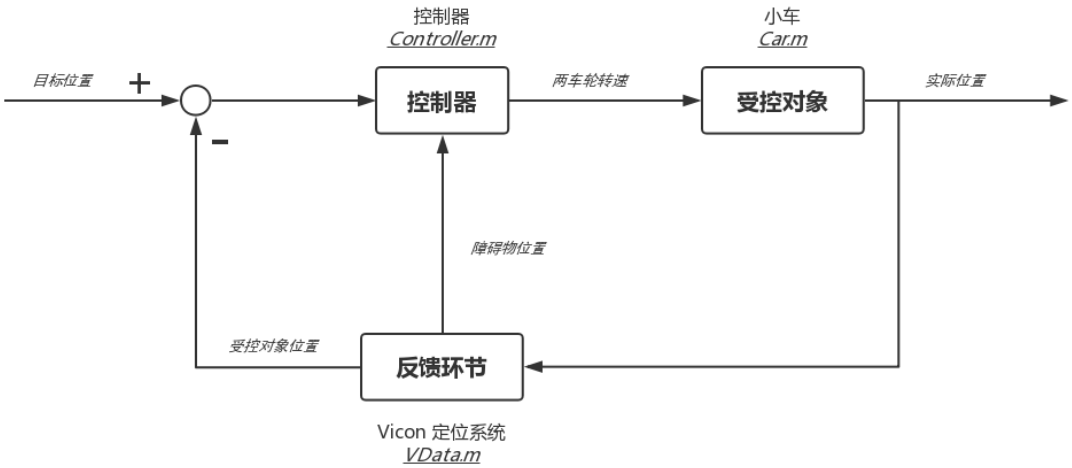


图 1: 模块间关系

### 1.1 Vicon 定位系统接口

Vicon 定位系统是一套动作捕捉系统，它提供了多种运动捕捉的功能 [1]。它通过红外摄像头追踪使用红外反光材料制作的定位球，并实时反馈定位球的位置信息。

使用 Vicon 定位系统的软件可将多个定位球进行绑定，形成一个虚拟刚体，并设置一个名称用于识别刚体。定位系统会根据定位球的相对位置，判断出虚拟刚体的位置和姿态，用户则使用刚体名称获取刚体的定位信息。

本项目中，我们在小车上安装多个定位球（五个以上效果较好），在定位系统中将同一个小车上的定位球进行绑定，形成一个虚拟的刚体，定位球即为刚体的顶点。小车运动时，定位系统会实时追踪刚体位置，通过读取、处理刚体位置，则可得到小车的位置、方向信息。

在 Vicon 定位系统接口部分定义为 *VData* 类，接口的所有功能通过使用此类的对象来实现。如可获取某刚体的位置及方位，同时可以通过获取的位置信息，实时显示小车的轨迹。

Vicon 定位系统模块主要工作流程如图2。在每个控制周期内，从 Vicon 系统读取新的数据（某次获取的数据被称为一帧），并刚体名称作为键值存储到 Matlab 中的 Map 数据结构中。之后每次查询某小车位置时，则从此数据结构中查询，而不从 Vicon 系统读取最新的数据。如果有绘制轨迹的要求，则可调用函数将所有小车的历史轨迹绘制出来。

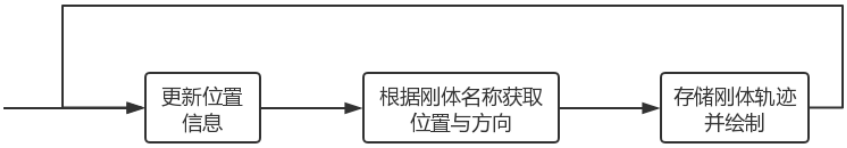


图 2: Vicon 定位系统工作流程

### 1.1.1 对象成员变量及初始化

*VData* 对象的主要成员变量有

- *MyClient* Vicon 系统对象
- *data* 某次更新后存储的一帧数据
- *fig*和 *ax* 小车轨迹图窗
- *history* 小车位置的历史数据，用于轨迹绘制
- *MAXPOINTS* 轨迹显示的长度
- *mousePos* 鼠标在轨迹图窗上点击的位置

*VData* 对象初始化时，使用 *MyClient* 变量存储 Vicon 对象，加载 Vicon 系统 SDK，并对 Vicon 系统进行相关设置。应注意，Vicon 对象由 Vicon 公司提供，使用时应将 *Matlab/include* 目录加入路径。

此后，对其他成员变量进行简单的初始化即可。

### 1.1.2 更新 Vicon 数据

使用 *read_data* 成员函数更新 Vicon 数据。

使用 Vicon 公司提供的 SDK 读取小车的名称、位置、欧拉角、四元数，存储在 *data* 成员变量中。从 SDK 获取的信息中，除了位置信息以外，还有一个 *Occluded* 变量，这个变量指示定位球是否被运动捕捉系统捕捉到。由于系统瞬间的误差，或者定位球被其他物体遮挡住，Vicon 系统便不能获取定位球的位置。如果 *Occluded* 变量为 1 ，说明运动捕捉系统未能捕捉到刚体位置，且返回的位置坐标为 $(0, 0, 0)$。如果不加以判断直接使用，这个错误的坐标会导致控制器的错误。

这里使用的解决办法是，重复读取位置信息，直到所有的小车都被读到，或者重复次数达到五次。如果重复次数达到五次，则可以判断定位失败不是由于 Vicon 系统暂时的错误，或者定位球被意外遮挡导致的，而更有可能是因为小车钻进桌下等需要操作者介入的原因导致，故不应反复尝试读取位置，可直接重复使用上一次读取到的位置作为此刻的位置。Vicon 系统的读取时间约为 $ms$ 级别，而整个控制系统的响应速度约为 $100ms$ 级别，所以重复读取数据并不会给系统造成影响。

更新信息后，调用 *updata_history* 成员函数更新历史轨迹。

#### 1.1.3 获取小车位置、障碍位置、小车角度

分别使用 *get_translation*、*get_obstacles*和 *get_rotation* 成员函数获取小车位置、障碍物位置、小车角度。三个函数的参数均为小车的名称。

其中，获取障碍物位置即查找所有名称不为给定名称的刚体位置，将这些位置打包返回。

#### 1.1.4 绘制小车轨迹

使用 *updata_trajectory* 函数绘制最新的小车轨迹。此函数使用 *updata_history* 函数中更新的 *history* 成员变量，绘制出其中最新的 *MAXPOINTS* 个定位坐标。

#### 1.1.5 获取鼠标点击位置

基于轨迹绘制功能，此模块还提供了鼠标点击设置目标位置的功能。使用 Matlab 函数可获取鼠标在轨迹显示图窗中最后点击的位置，并判断点击的位置是否为有效位置。如果目标位置超出 Vicon 定位系统的可探测范围，若仍使小车前往此位置，则有失控的危险，所以此时应当将 *readMouse* 成员变量设置为假，表示没有读取到有效的位置信息。否则将变量设置为真，表示可以使用读取到的鼠标位置。

使用这一函数，可以达到鼠标点击控制小车位置的效果。

### 1.2 小车控制接口

小车通过蓝牙接收指令，指令的格式为 *{333%+dd%+dd}* ，其中 *+dd* 代表两位有符号整数，指定左右轮的速度，符号代表方向。如 *{333%+50%-50}*代表左轮以 50 的速度反转，右轮以 50 速度正转。

小车内部已经设置有速度闭环，指令给定 $-99 \sim 99$ 范围的速度时，小车便会闭环控制使得两轮的速度达到目的值。速度的单位为某个固定值，经测试拟合可知约为 $12.7mm/s$ ，也即小车最快速度约为 $1.3m/s$ 。

小车控制模块定义了 *Car* 类，用于完成蓝牙连接、速度指令发送等功能。

#### 1.2.1 蓝牙连接及初始化

*Car* 类的构造函数接收小车蓝牙的 id 及频道，并连接到小车蓝牙。

应当注意的是，Vicon 定位系统获取的角度信息为虚拟刚体的角度，但是虚拟刚体的朝向与小车的前进方向朝向没有关系，因此获得的角度与小车的前进方向相差一个固定角度。定义 *get_angle* 成员函数获取这个固定的角度。为了得到这个角度，控制小车以恒定速度前进，并通过 Vicon 系统获取位置信息。使用直线拟合的方法，得到小车轨迹的角度，并计算出此角度与 Vicon 读取角度的差值，即为所求角度。

初始化小车需要上述连接蓝牙、获取偏差角度两个步骤。这两个步骤比较耗时，尤其连接蓝牙这步需要花费大量时间，若每次运行程序都重新连接则会浪费较多时间。且如果未断开蓝牙，无法再次连接上小车。为解决这个问题，利用 Matlab 全局变量不会被自动清理的特性，仅在首次使用时连接到小车，并测定小车方向与 Vicon 系统读取方向的偏差。一个例子如 Listing 1所示，尝试控制小车，若无法控制或变量不存在，则连接蓝牙并获取方向偏差。

```
try
    car.stop();
catch
    fprintf('Connecting to car... ');
    car = Car('btspp://AB5BC3563402', 1);
    thetaFixCar = car.get\_angle(vicon, 'car');
    fprintf('Done\n');
end
```

Listing 1: 小车连接及初始化示例

#### 1.2.2 速度指令发送

如前所述，小车驱动接收 $-99 \sim 99$ 范围的整数速度值，并可认为实际速度在极短时间内可达到给定值。但控制器计算所得的速度值单位和小车驱动所接受的速度单位并不相符，给定速度时需要左转换。使用 *set_speed* 成员函数设定小车速度，函数参数为 $mm/s$ 做单位的速度值。函数接收参数后，将实际速度除以 12.7 并取整，转化为小车驱动要求的速度格式。

由于实验场地及其有限，若不对小车速度加以限制，容易发生碰撞等现象。因此在设置小车速度时，判断设定速度是否大于预先设定的最大值，如果速度在要求的范围内，则直接设置速度，否则将两轮速度按比例缩放到阈值。

使用 *set_MAX_SPEED* 函数，调整小车的最大速度限制。

除了给小车设定目标速度以外，直接让小车停止也是一个常用的功能。*stop* 成员函数直接调用 *set_speed* 函数，将速度设置为 0，使小车停止。

#### 1.2.3 对象析构

如前所述，如果某台电脑连上了小车蓝牙，在没有手动断开蓝牙之前，无法再次连接到小车。因此此类的对象需要有特别的析构方法。在 *delete* 函数中，定义了关闭蓝牙的正确方法。

### 1.3 控制器模块

控制器模块接收小车的实际位置、方向，以及目标位置、方向和障碍物位置，计算出小车两轮的速度。此项目中将多辆车分别进行控制，对某辆车来说，其他车为障碍物，故控制器只需要完成

一辆车的控制。

为了将控制器设计与定位、小车指令发送部分分割开，不让负责算法设计的同学关注其余部分的技术细节，将这一模块定义为 *Controller* 类。这一做法的另一个好处是，当需要对控制器进行仿真时，只需使用相同的控制器代码即可。

注意，小车实际方向与定位系统读取方向的偏差记录于此类中。

## 2 示例程序

程序文件为 *Matlab/demo.m* ，程序框图见图3。

此 demo 展示了如何使用系统框架完成初始化、数据记录，并设置小车编队，追逐鼠标设定的目标点，同时实时绘制小车轨迹。

程序初始化部分，加入 Vicon 系统的 SDK 等相关文件路径，并初始化 VData 对象，用于读取定位系统信息。打开数据记录文件，记录实验时间。

尝试连接三辆小车。若某辆小车未曾初始化，或蓝牙异常断开等原因，无法正常控制小车，则连接到此辆小车，并获取小车方向与定位系统读取方向的夹角。记录三辆小车的角度偏差，以便于实验结束后的分析。使用三辆小车的角度偏差，分别初始化控制器，用于控制小车。

打开停止程序对话框，使得用户能即使控制程序停止。

程序主循环部分，判断停止程序对话框是否被关闭，如果被关闭，则说明用户希望停止程序，则跳出循环，否则继续执行循环中的语句。类似的，如果轨迹绘制图窗被关闭，也可以做出相同的判断，退出循环。

首先更新定位系统的数据，用于此次循环中的控制器计算。并根据最新的定位系统数据，绘制小车的轨迹图。读取鼠标在轨迹图窗中点击的坐标，用于控制第一辆小车的运动。另两辆小车跟随第一辆车运动。记录三辆车的位置、姿态、控制指令信息，以及时间。由于控制器中设计的控制间隔为 $0.1s$ ，读取此次循环的时间，若不足 $0.1s$ ，则等待至 $0.1s$ ，再进入下一次循环。

循环结束后，关闭数据记录文件，并停止三辆小车的运行。

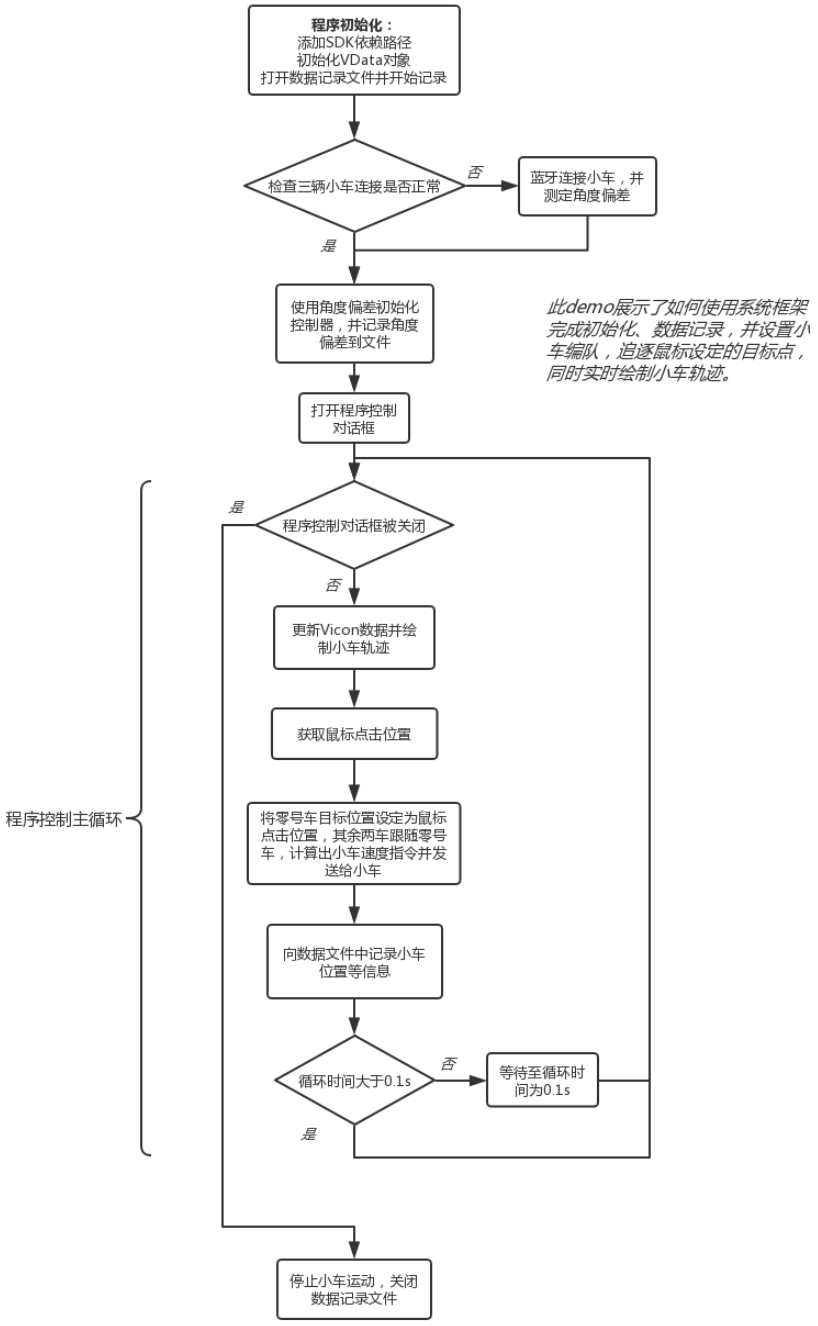


图 3: 示例程序框图

## 参考文献

[1] Motion Capture Systems | VICON. https://www.vicon.com/. Accessed on 2019-06-21.