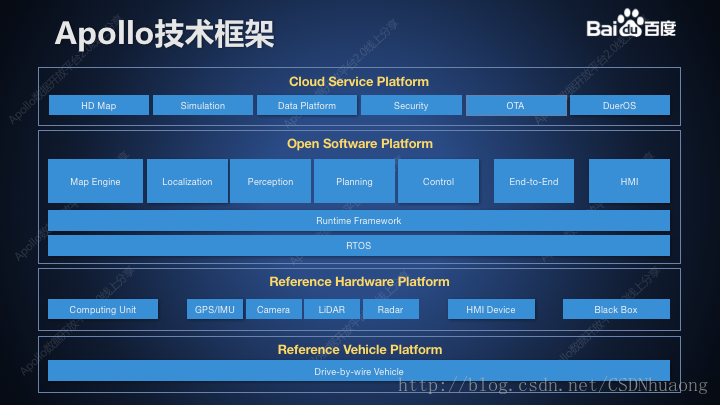
**Apollo技术框架**



**Apollo技术框架由4层构成。分别是**：

Reference Vehicle Platform   
Reference Hardware Platform   
Open Software Platform   
Cloud Service Platform

1. 首先我们需要一辆能够受我们信号控制的车，那么就我们管它叫信号车。然后主要是我们和很多车厂和Tier1的这个方案厂商一同来推进的一层；
2. 然后是Reference Hardware，我们要想让车能动起来，需要计算单元、GPS/IMU、Camera、激光雷达、毫米波雷达、人际交互设备、BlackBox等等硬件的支持
3. 开放软件层。它分成三个子层，有实时的操作系统。有承载所有模块的框架层。高精地图与定位告诉车辆身在何处，感知模块告知车辆周围环境，决策规划模块决定全局与细节规划。控制模块负责将决策规划产出的轨迹生成为具体的控制命令
4. 百度自动驾驶有非常强的能力来源于云端的。高精地图存储于云端，用以模拟驾驶的仿真服务也在云端，它加速了自动驾驶的研发效率。我们积累了大量的数据和对数据的理解。我们在云端也开放了数据平台，把数据加工的主线，在我们云端来体现出来。在云端还有我们的安全服务，保证软件的自动更新。

**Apollo 2.0**



Apollo 2.0新开放的模块包括了Security、Camera、Radar和Black Box，这意味着Apollo平台包括云端服务、服务平台、参考硬件平台以及参考车辆平台在内的四大模块已全部点亮。Apollo 2.0首次开放安全和OTA升级服务，只允许正确和被保护的数据进入车内，并进一步强化了自定位、感知、规划决策和云端方阵等能力。其中Black Box模块包括了软件和硬件系统，能够实现安全存储和大容量数据集传输，可以帮助我们及时发现异常情况，提升整个平台的安全可靠性。

硬件方面，增加两个前向摄像头（长焦+短焦）主要用于识别红绿灯，正前方保险杠上方新安装了毫米波雷达。所以很明显，在Apollo 2.0开放了Camera和Radar的模块后，整个平台更强调了传感器融合的能力，已进一步增加其对昼夜简单城市道路工况的适应能力。

## 640?wx_fmt=jpeg&wxfrom=5&wx_lazy=1一、硬件

**计算中心**：就是整套系统大脑了，使用的是Neousys Nuvo-6108GC，这是一款性能强大的X86解构工业控制计算机，强大到什么程度呢？支持至强E3和I7，支持GTX1080显卡，玩游戏的朋友都知道这款显卡，强大的性能我就不多说了，所以比较适合用来做自动驾驶。

**CAN通讯卡：**用来和汽车进行通讯以控制汽车的加速、制动、档位、方向等信号，使用的是ESD CAN-PCIe/402-B4，CAN卡直接插在主机内。

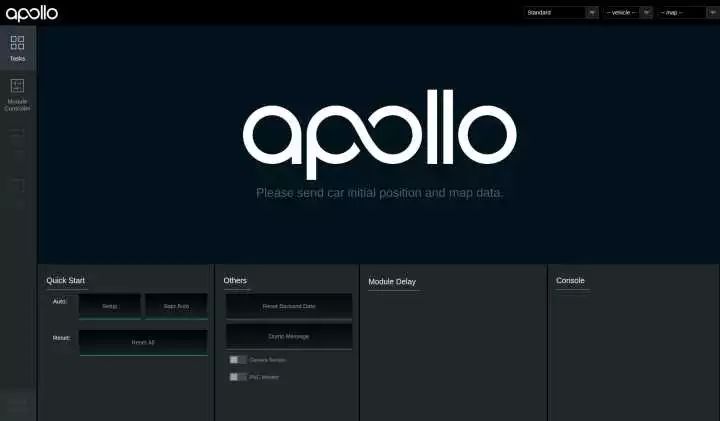
**GPS和IMU：**定位系统和惯性制导，当然就是用来进行GPS定位和惯性定位，使用的是NovAtel SPAN-IGM-A1或者NovAtel SPAN® ProPak6™ and NovAtel IMU-IGM-A1，通过串口连接主机。

**激光雷达：**扫描距离达到120米，水平360度扫描，垂直FOV26.9度，型号为Velodyne HDL-64E S3，通过以太网连接计算中心。

**摄像头：**用于视觉，Leopard Imaging LI-USB30-AR023ZWDR with USB 3.0 case，通过USB连接主机。

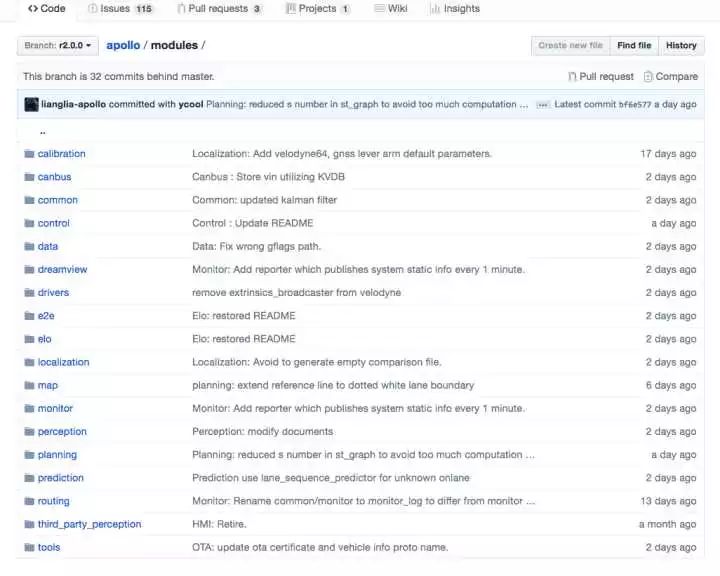
**毫米波雷达：**用于探测前方，安装在车辆前端，为大陆集团的ARS408-21，连接至CAN卡。

除去以上这些，还需要鼠标键盘显示器，当然就是平常用的那种。



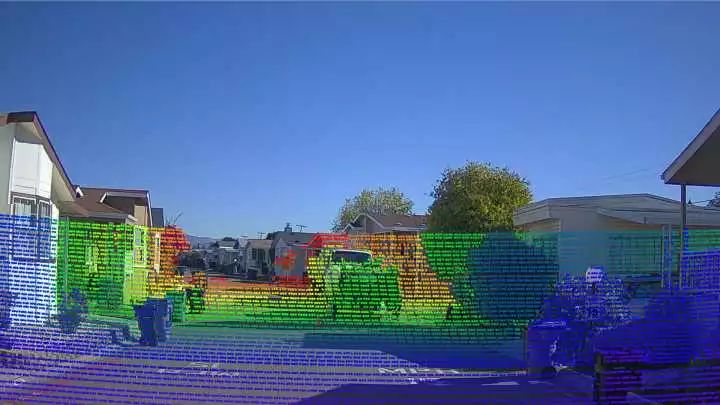
操作系统：Ubuntu Linux 推荐版本14.04

整个软件包含如下模块：



**网站：https://github.com/ApolloAuto/apollo/tree/r2.0.0?files=1**

**calibration：**校准模块，使用前必须对系统进行校准和标定，包括激光雷达与摄像头、毫米波雷达与摄像头等。所谓校准就是要对齐激光雷达、摄像头以及毫米波雷达获得的信息，我们知道激光雷达可以获得详细的3D信息，但是不能获得颜色信息，摄像头可以获得颜色信息，但是无法获得深度等3D信息，毫米波雷达不能获得颜色信息，但是可以获得3D信息，三者获得的信息对齐后，就可以同时获得实际环境中的3D信息和颜色信息。下图左就是对齐后的激光雷达和摄像头画面，下图右为对齐后的毫米波雷达和激光雷达画面。



**canbus：**管理CAN卡和CAN通讯，把接受到的信号传递给相应模块，同时将Control模块的命令下发到车辆。

**common：**其他模块之外的代码都在这里。

**control：**主控制模块 ，基于车道规划和车辆当前状态，输出转向、加速和制动控制信号到CAN卡。

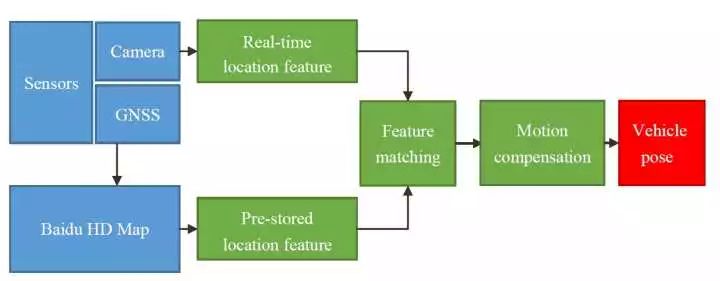
**data Data:**收集、存储、处理收集到的各种数据。

**dreamview：**这是一个Web应用，可以帮助用户随时掌握系统的输出数据，包括车道、位置、车身等情况。

**drivers：**此模块包含CAN卡、激光雷达、毫米波雷达、GPS以及摄像头等相关设备的驱动程序。

**e2e：**end to end，端到端深度学习，所谓e2e指的是由传感器的输入，直接决定车的行为，例如油门，刹车，方向等。也就是机器学习的算法直接学习人类司机的驾驶行为。这部分在代码中需要另外下载，学习的数据主要来源于传感器的原始数据，包括图像、激光雷达、雷达等。end-to-end输入以图像为主。 输出是车辆的控制决策指令，如方向盘角度、加速、刹车。 连接输入输出的是深度神经网络，即通过神经网络直接生成车辆控制指令对车辆进行横向控制和纵向控制，中间没有人工参与的逻辑程序。横向控制，主要是指通过方向盘控制车身横向移动，即方向盘角度。纵向控制，是指通过油门和刹车控制车身纵向的移动，即加速、刹车等。横向模型的输出没有采用方向盘角度，而是使用要行驶的曲率（即拐弯半径的倒数）。

**elo：**百度的车辆自身定位部分，结构如下。



**localization：**车辆定位服务，包含两种定位方式，一种是GPS和IMU，另一种是多传感器融合。输出位置估算结构体。

**map：**地图

**monitor：**此模块用于监测硬件状态及整个系统的健康程度。

**perception：**此模块用于障碍物感知和红绿灯等交通信号的感知，主要依靠摄像头数据以及激光雷达和毫米波雷达的数据，当然还有高精度地图，输出3D的障碍物信息，包括方向、速度和障碍物类型，当然还有红绿灯等交通信号。模块可以将障碍物标注为机动车、非机动车、行人和其他，使用的是激光点云算法。目前程序只能识别白天的红绿灯信号。

**planning：**根据车辆位置和车辆状态、地图、障碍物、导航信息等计算具体的车道。规划车道有两种方式，一种是提前把轨迹存入程序，然后根据车辆状态和位置提取轨迹另外一种是实时计算。

**prediction：**根据障碍物的位置、航向、速度、加速度计算障碍物的可能轨迹。**routing：**路径规划，根据地图和起点终点位置计算出具体的导航信息。

**third\_party\_perception：**第三方感知

**tools：**工具

**整体流程**

1.首先，用户输入目的地，routing模块就可以根据终点位置计算出具体的导航信息。激光雷达、毫米波雷达和摄像头拍摄到的数据配合高精度地图由percepting模块计算出3D的障碍物信息并识别交通标志及交通信号，这些数据进入perdiction模块，计算出障碍物的可能轨迹，如此就可以结合以上信息并根据车辆定位模块localizationg提供的车辆位置由planning模块得到车辆应该走的具体车道。

2.得到车道后车辆control模块结合车辆的当前状态计算加速、刹车和方向的操作信号，此信号进入CAN卡后输出到车内，如此实现了车辆的自动驾驶。

3.在整个流程中，monitor模块会及时监测硬件及系统的健康状况，出现问题肯定就会中止驾驶过程。对于驾驶中的信息，用户可以通过web应用dreamview来查看，下图就是实际驾驶过程中的dreamview界面。

