

# Apollo

阿波罗是百度发布的名为“Apollo（阿波罗）”的向汽车行业及自动驾驶领域的合作伙伴提供的软件平台。

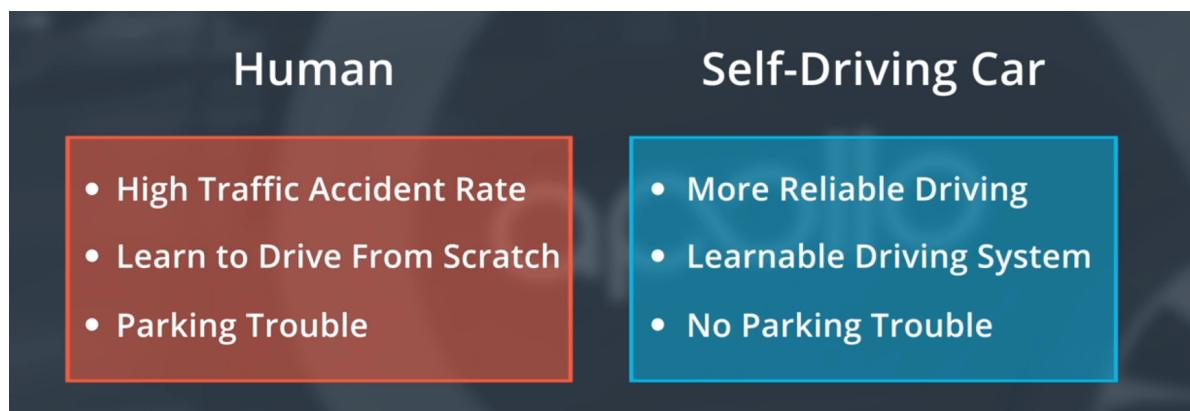
## 第一课：无人驾驶概览

了解无人驾驶车的关键部分与 Apollo 团队架构，开启无人驾驶入门的学习路径。

核心点：

- (1) 高精度地图：几乎支持软件栈的所有其他模块
- (2) 定位：讨论汽车如何确定它所处的位置，利用激光和雷达数据，将这些传感器感知内容与高分辨率地图进行对比，这种对比使汽车自定定位。
- (3) 感知：了解无人驾驶车如何感受这个世界；深度学习；卷积神经网络
- (4) 预测：几种不同的方式用于预测其它车辆或行人可能如何移动；神经递归网络
- (5) 规划：如何将预测和规划相结合以生成车辆轨迹
- (6) 控制：如何使用转向、油门和制动来执行规划轨迹

为什么需要智能驾驶？安全



6个等级的无人驾驶车0-5

0：驾驶员是系统的唯一决策者

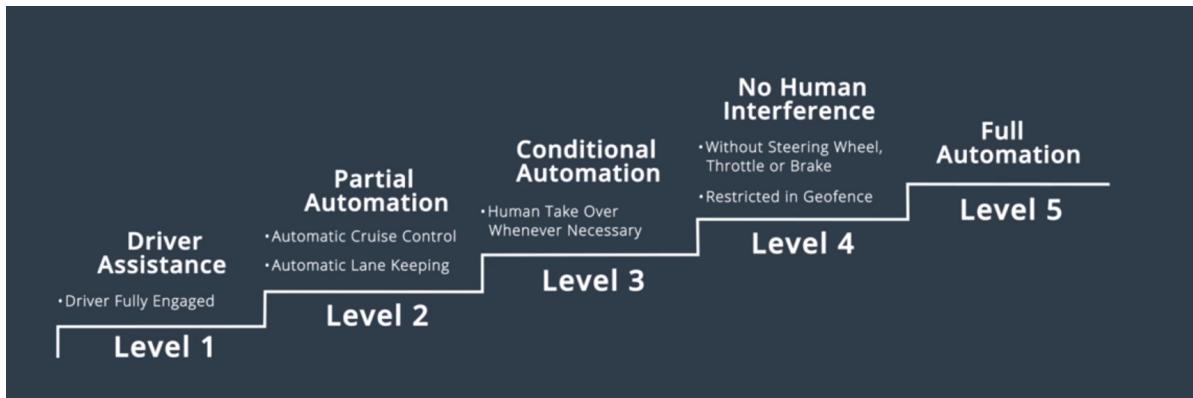
1：驾驶员辅助：车辆为驾驶员提供转向或加速支持

2：部分自动化：车辆控制的几项功能（巡航控制、车道保持），驾驶员必须执行自治系统的任何功能

3：有条件自动化：车辆自主驾驶，驾驶员有必要时随时接管

4：高度自动化：车辆控制、驾驶体验的所有方面不希望驾驶员介入；车辆可能没有方向盘或者控制装置；车辆可能被限制在某些区域内

5：完全自动化：车辆可以在人类驾驶的任何地方完全自主地运行



无人驾驶车包括五个核心部件：计算机视觉、传感器融合、定位、路径规划、控制



Apollo技术框架由四个层面组成：参考车辆平台、参考硬件平台、开源软件平台、云服务平台

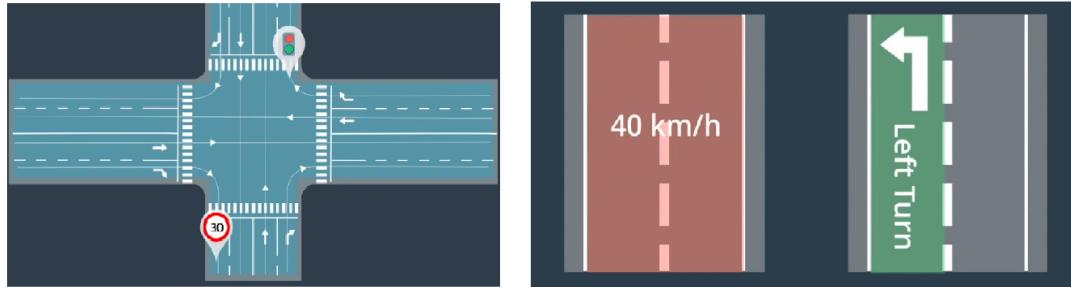
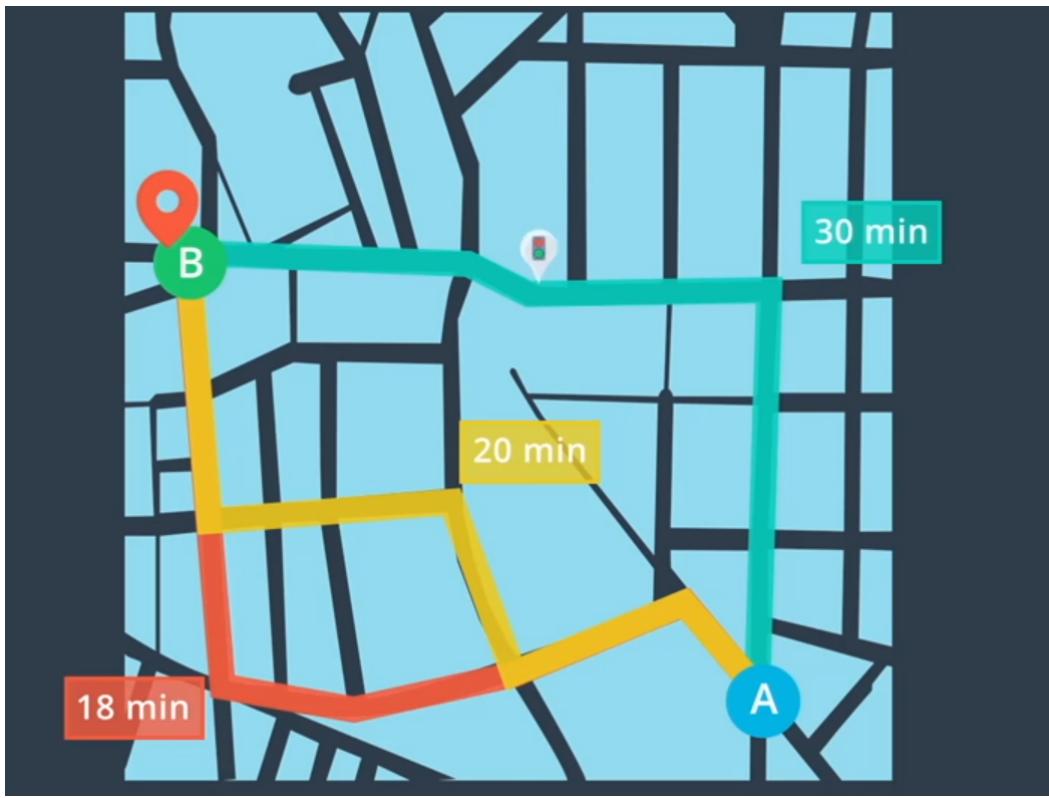
## 第二课：高精地图

了解高精度地图的实现逻辑，这是 Apollo 定位、感知、规划模块的基础。

### 1.地图简介

无人驾驶需要更详细、精确的地图

高精地图VS传统地图



高精地图的重要特征之一是 精度（厘米级误差）

地图与定位、感知与规划的关系

### (1) 地图和定位

无人车通过传感器（camera、雷达、激光点云等）收集信息，再与高精地图上已知的坐标进行比较，这一过程需要**预处理**、**坐标变换**、**数据融合**的复杂过程。

**预处理**消除了不准确或质量差的数据；

**坐标变换**将来自不同视角的数据转换成统一的坐标系；

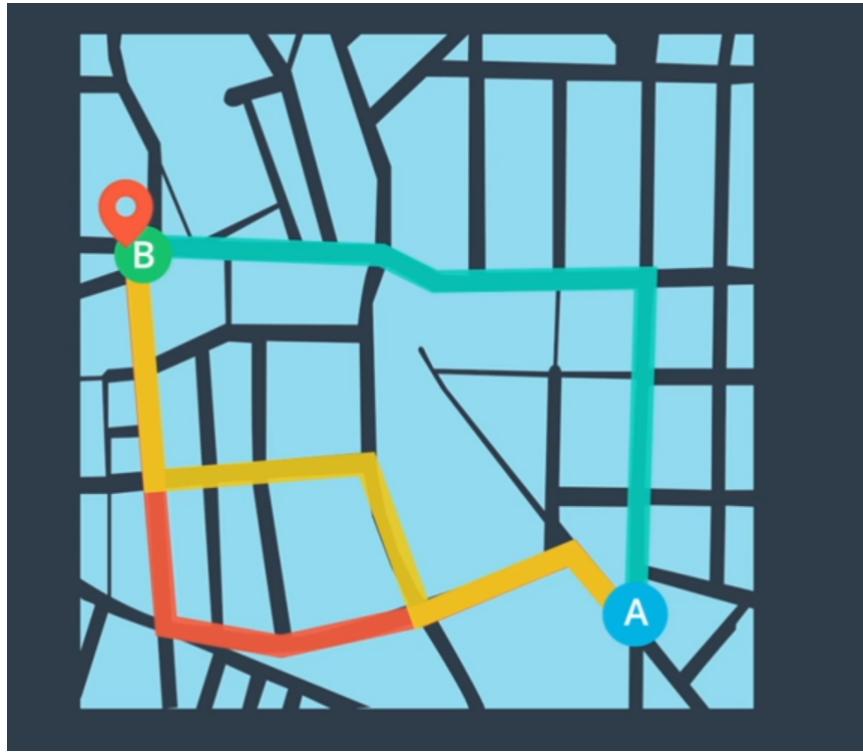
借助**数据融合**，可将来自各种车辆和传感器的数据合并；

一旦无人驾驶车高度精确了其位置，定位也就完成了。

### (2) 地图和感知

摄像机、激光雷达、雷达的探测能力在超过一定距离后都会受到限制，在恶劣的天气和夜间更甚。即使传感器没有检测到交通灯，高精度地图也可以将交通信号灯的位置提供给软件栈的其余部分。另一点，高精地图可以帮助传感器缩小检测范围，例如找停车标志。

### (3) 地图和规划



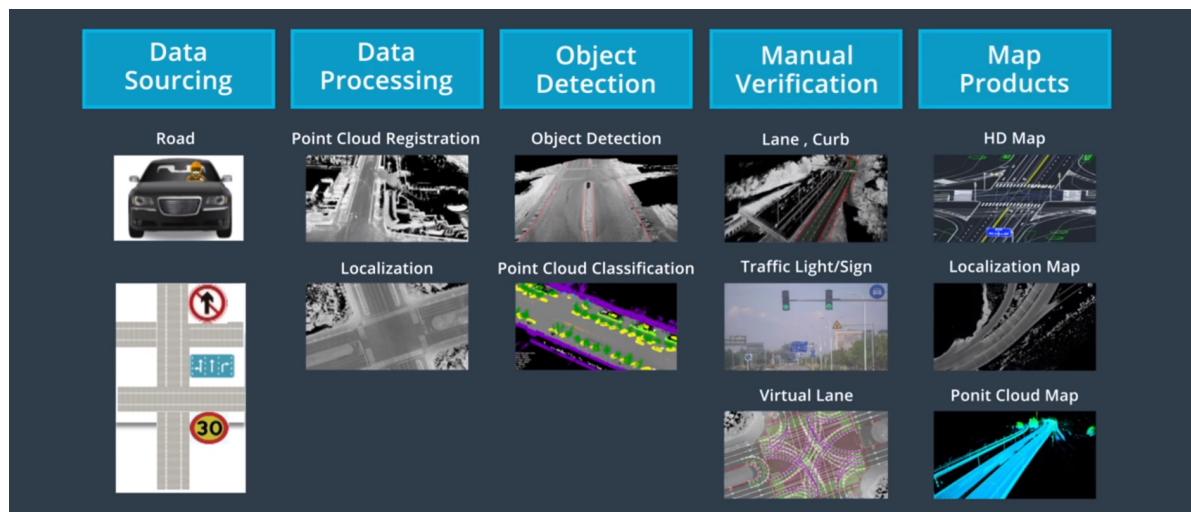
高精地图可以帮车辆找到合适的行车空间，规划不同的路线，并帮助预测软件预测道路上其它车辆在将来的位置。

高精地图专为无人驾驶设计。Apollo地图包含地图定义、交叉路口、交通信息、车道规则以及汽车导航等。

例如：高精地图会记录交通信号灯的精确位置和高度，大大降低感知难度。

Apollo高精度地图采用了 OpenDRIVE格式，这是一种行业制图标准。

Apollo高精地图构建包括五个过程：数据采集、数据处理、对象检测（静态物体：电线杆、交通标志）、手动验证、地图发布



## 第三课：定位

了解车辆如何以个位数厘米级别的精度进行自定位。

定位是让无人驾驶车知道自身确切位置的方法。

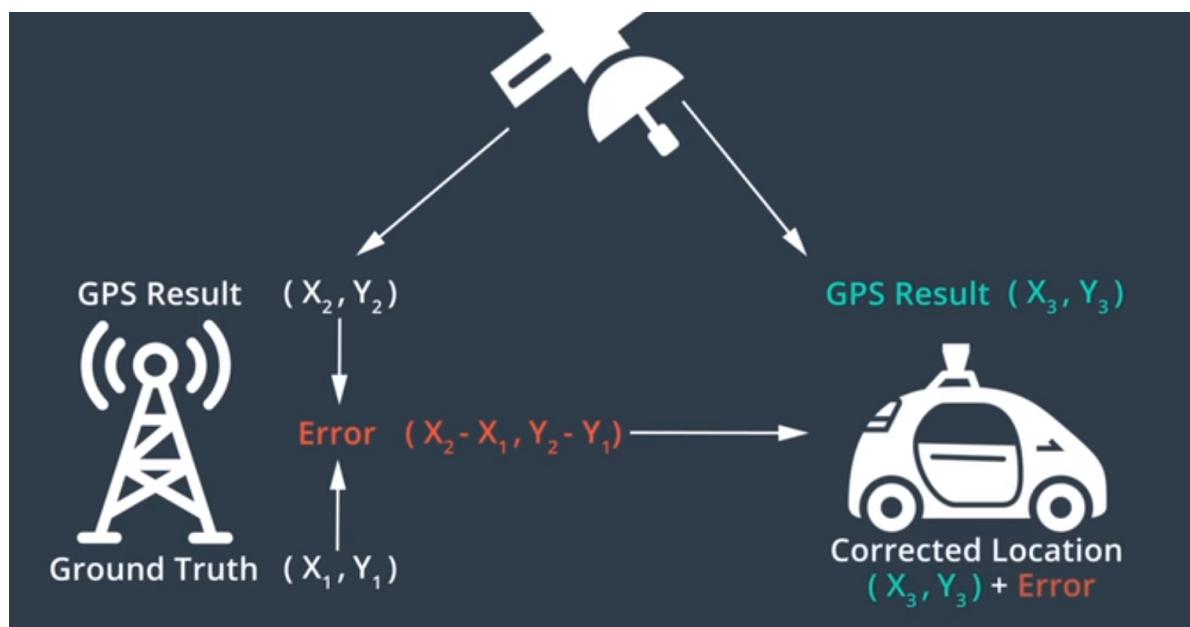
GPS可信度不高。必须要找到了另一种方法来准确地定位车辆在地图上的位置。

常用的方法是：将汽车传感器所看到的内容与地图上所显示的内容进行比较，并将自身坐标系和地图坐标系进行数据转换。



如果有一张地图，里面标明了这些地标在世界上的位置，你也就知道了自己的确切位置。 【三角测量】

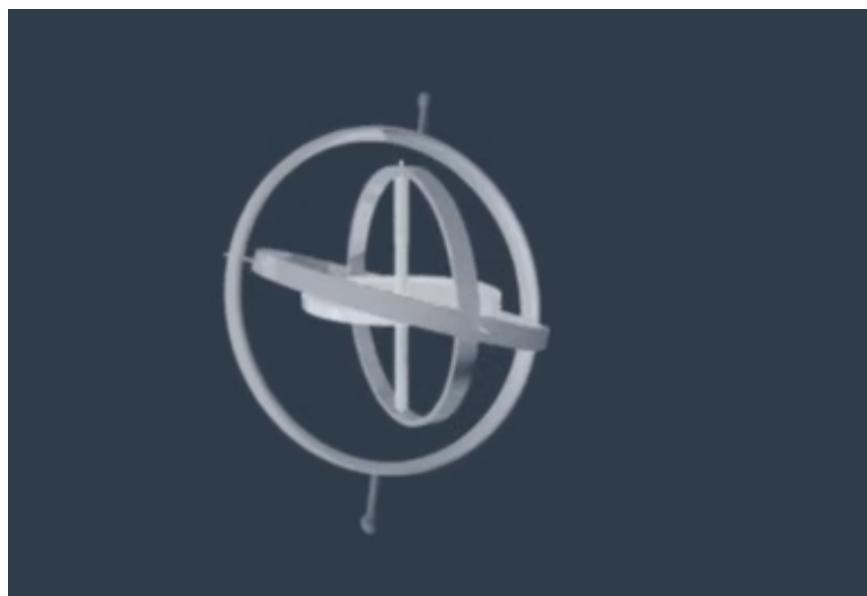
全球导航卫星系统或者GNSS，GPS是使用最广泛的GNSS系统。



惯性导航：

可以使用加速度、初始速度和初始位置来计算汽车在任何时间点的车速和位置。

如何测量加速度呢？三轴加速计->陀螺仪



三个外部平衡环一直在旋转，旋转轴始终固定在世界坐标系中，计算车辆在坐标系中的位置是通过测量旋转轴和三个外部平衡环的相对位置来计算的。加速度计和陀螺仪是惯性单元（IMU）的主要组件。

IMU的一个重要特征是它以高频率更新，达到1000Hz，可以提供实时的位置信息。缺点：运动误差随时间累加。

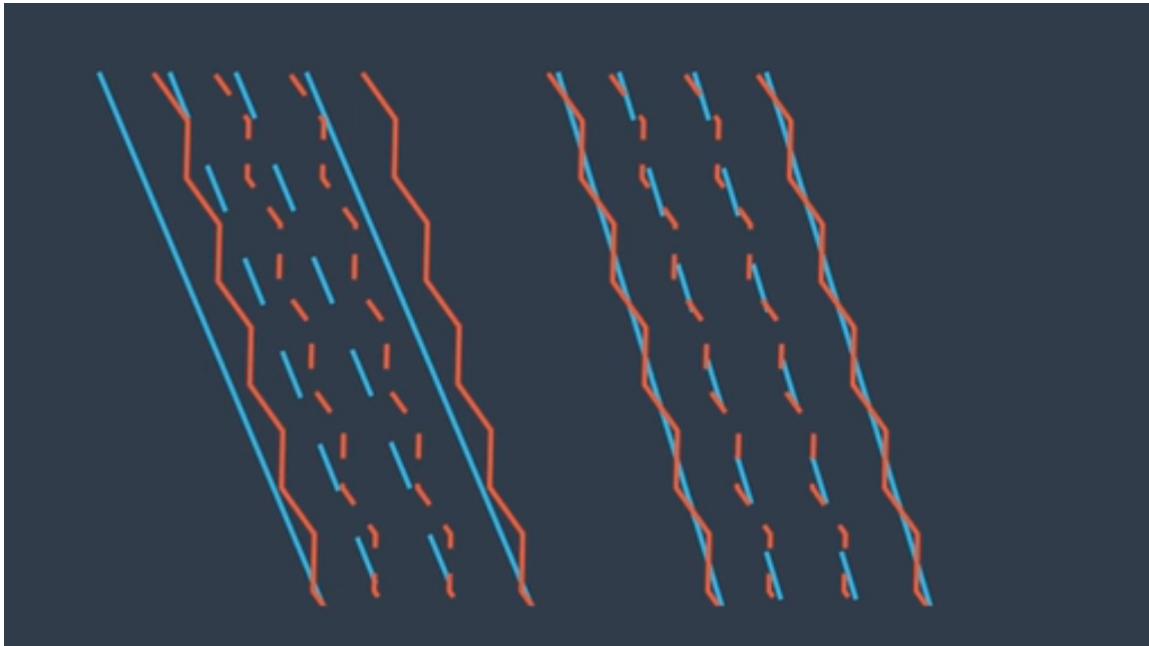
只能依靠IMU单元在很短的时间内进行定位。

结合GPS和IMU：一方面，IMU弥补了GPS更新频率较低的缺陷；另一方面，GPS纠正了IMU的运动误差。

如果在山间或者隧道里，可能长时间没有GPS更新；利用激光雷达，可以通过点云匹配来对汽车进行定位，该方法将来自激光雷达传感器的检测数据与预先存储的高精度地图连续匹配，通过比较，可以知道位置和方向

## 视觉定位

图像是要收集的最简单的数据类型。可以将摄像头与地图进行比较



优点：图像数据很容易获得；

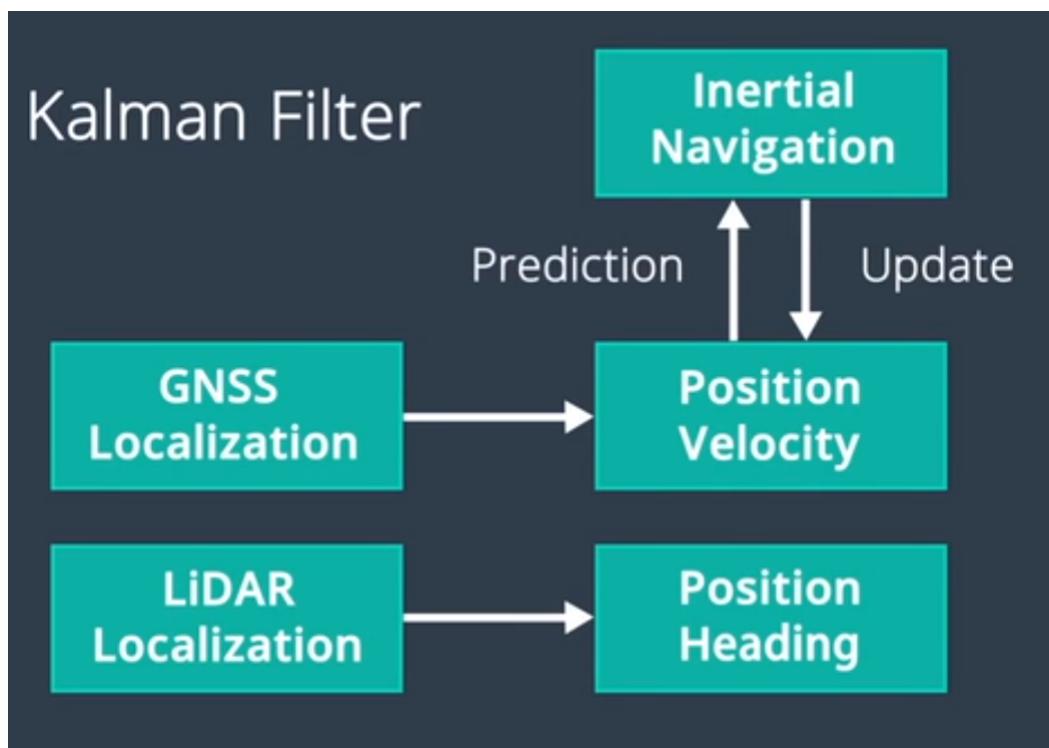
缺点：缺乏三维信息和对三维地图的依赖。

Apollo使用基于GPS、IMU和激光雷达的多传感器融合的定位系统。

Apollo定位模块依赖于IMU、GPS、激光雷达、雷达、高精地图。

这些传感器同时支持GNSS定位和LiDAR定位。

GNSS定位输出位置信息和速度信息； LiDAR定位输出位置和行进方向信息



## 第四课：感知

了解不同的感知任务，例如分类、检测和分割，并学习对感知而言至关重要的卷积神经网络。

汽车通过静态摄像头和其它传感器来感知周围环境：计算机视觉

## 计算机视觉

无人驾驶车有四个感知世界的核心任务

检测：找出物体在环境中的位置；

分类：明确对象是什么；

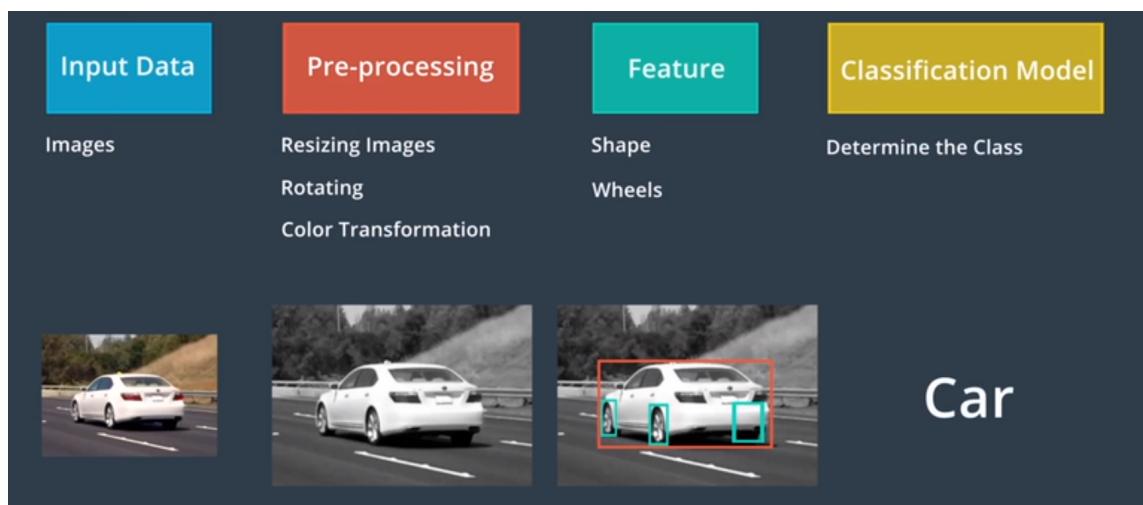
跟踪：随时间推移观察移动物体；

语义分割：将图像中的每个像素与语义类别进行匹配。



图像分类器是一种将图像作为输入，并输出标识该图像的标签或者“类别”的算法。

过程：



### 摄像头图像

多数颜色和形状转换都是通过对图像进行数学运算以及逐一像素进行更改来完成。

### LiDAR图像

激光雷达传感器创建环境的点云表征，提供了难以通过摄像头图像获得的信息，如高度和距离。

通过激光反射来测距离

### 机器学习

使用特殊的算法训练计算机从数据中学习计算机科学领域

机器学习涉及使用数据和相关的真值标记来进行模型训练。监督学习、无监督学习和半监督学习

### 神经网络

人工神经元负责传递和处理信息，也可以对这些神经元进行训练。

通过密集训练，计算机可以辨别汽车行人等

神经网络从图像中提取许多特征，但是这些特征可能是人类无法描述甚至无法理解的特征。

## 反向传播算法

学习由三步循环组成：前馈、误差测量、反向传播

## 卷积神经网络

### 检测与分类

对交通信号灯检测分类：使用计算机视觉对交通信号灯进行定位，根据灯光的颜色对交通灯进行分类

先使用CNN来检查图像中的对象位置，在对图像中的对象进行定位之后，再将图像发送给另一个CNN进行分类。也可以使用单一CNN体系结构对对象进行检测和分类

### 跟踪

检测完对象后，需要跟踪它们。

追踪的第一步是确认身份，通过查找特征相似度最高的对象，再将之前帧中所检测的对象与之进行匹配。

再确认对象身份后，可以使用对象的位置并结合预测算法，以估计下一个时间的速度和位置，这个预测可以帮助我们识别下一帧中的相应用对象。

### 分割

语义分割涉及对图像的每个像素进行分类。

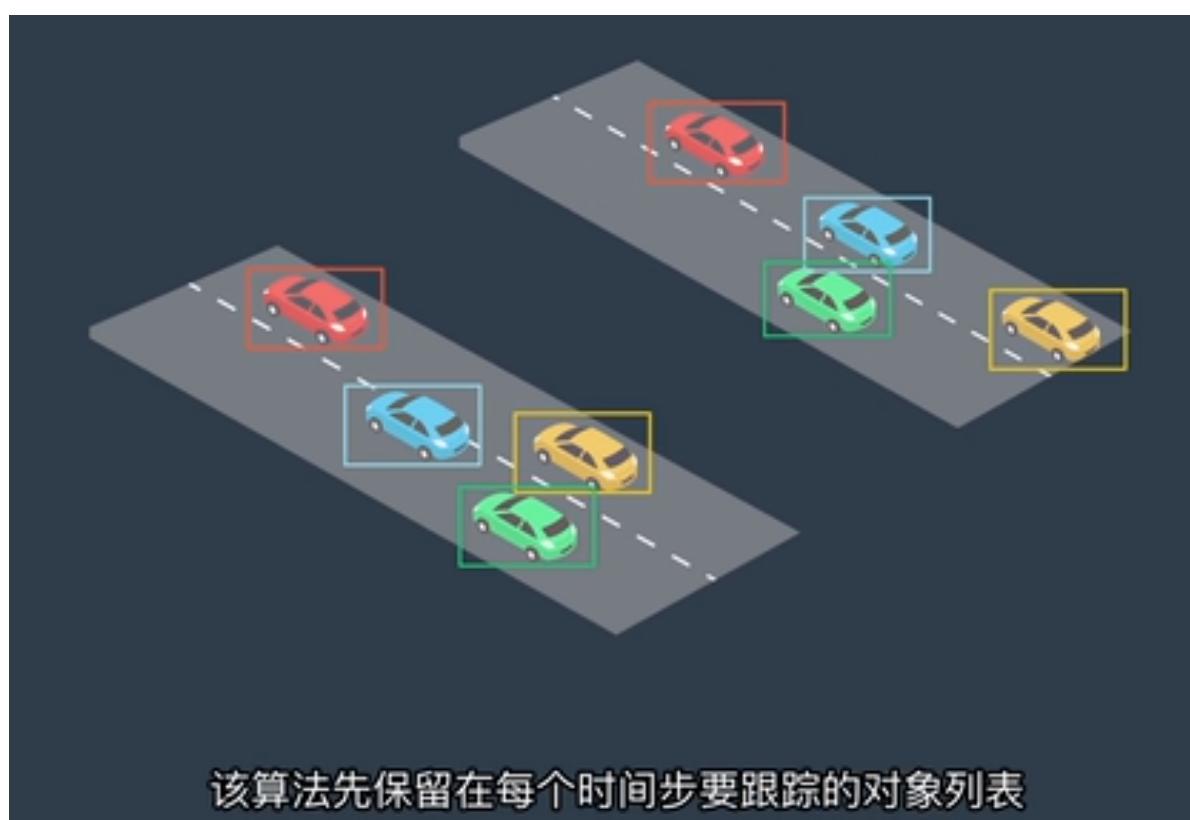
语义分割依赖一种特殊的CNN，它被称作全卷积网络或FCN

### Apollo感知

Apollo在高精地图上使用感兴趣区域（ROI）来重点关注相关对象。

Apollo将ROI过滤器应用于点云和图像数据，以缩小搜索范围并加快感知，然后通过检测网络馈送已过滤的点云....

最后使用检测跟踪关联的算法来跨时间步识别单个对象。



然后在下一个时间步中找到每个对象的最佳匹配。

对交通灯的分类：Apollo先使用高精度地图来确定前方是否存在交通信号灯。如果前方有交通信号灯，则高精地图会返回灯的位置，这侧重于摄像头搜索范围，在摄像头捕获到交通信号灯图像后，Apollo使用检测网络对图像中的灯进行定位，然后提取交通信号灯，再分类。

Apollo使用YOLO网络来检测车道线和动态物体（车辆、卡车、人等），经过YOLO网络检测后，在线检测模块会并入来自其他传感器的数据，对车道预测进行调整，车道线最终被并入名为“虚拟车道”的单一数据结构中；同样，也通过其他传感器的数据对YOLO网络所检测到的动态对象进行调整，以获得每个对象的类型、位置、速度和前进方向。虚拟通道和动态对象均被传递到规划和控制模块。

	Camera	LiDAR	Radar	Camera+Radar+LiDAR
Object Detection	Yellow	Cyan	Yellow	Cyan
Object Classification	Cyan	Yellow	Red	Cyan
Range of Visibility	Yellow	Yellow	Cyan	Cyan
Lane Tracking	Cyan	Red	Red	Cyan
Functionality in Bad Weather	Red	Yellow	Cyan	Cyan
Functionality in Poor Lighting	Yellow	Cyan	Cyan	Cyan

● Good ● Mixed ● Poor

## 第五课：预测

学习不同的预测方式，让 Apollo 无人驾驶车预测其他车辆或行人是如何移动的。

### 预测简介\_a

预测行人、汽车等的行为，确保做出最佳决策。通过生成一条路径来预测一个物体的行为。

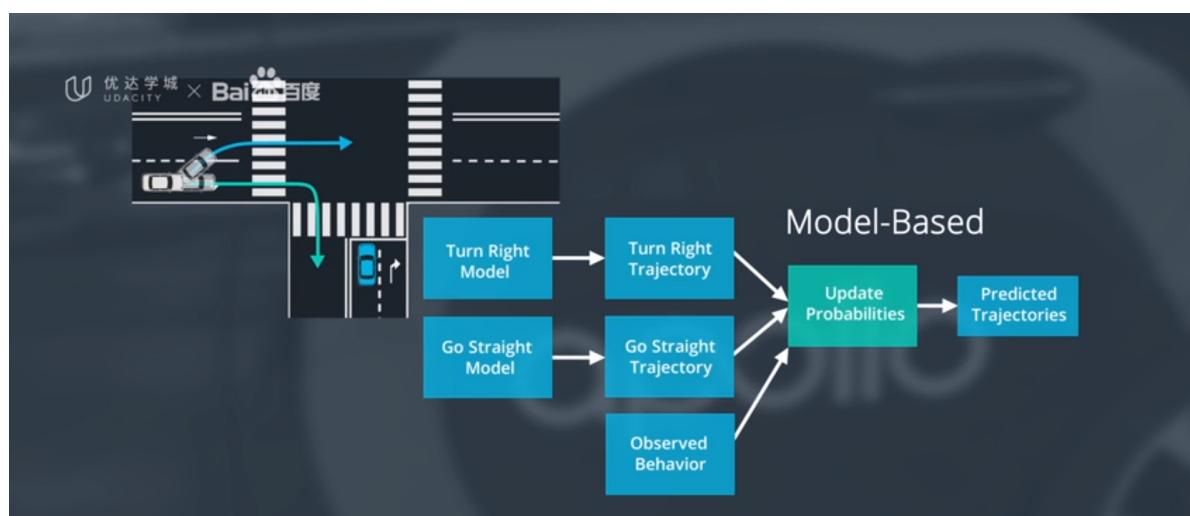
学习Apollo如何根据其他移动物体的状态以及无人车的位置去预测这些移动物体的路径。

### 预测简介\_b

预测路径有实时性的要求，延迟越短越好。

准确性

#### (1) 基于模型的预测



构建两个候选的预测模型。一个模型描述该车进行右转弯；另一个模型描述该车继续直行。

并且认为任意一种模型发生的概率是相同的。继续观察车的运动，看它和哪一条轨迹更匹配。

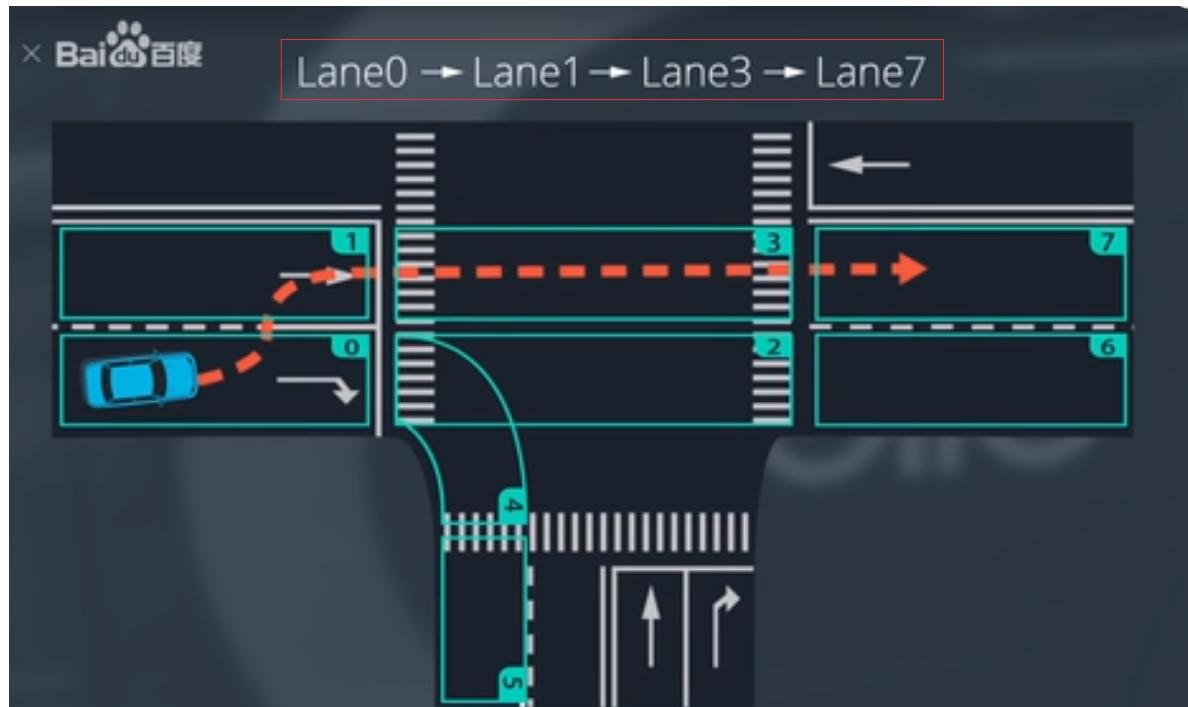
## (2) 基于数据驱动的预测

数据驱动预测使用机器学习的算法，通过观察结果来训练模型，一旦机器模型训练好，就利用此模型去做预测

优点：训练数据越多，模型效果越好。

### 基于车道的预测

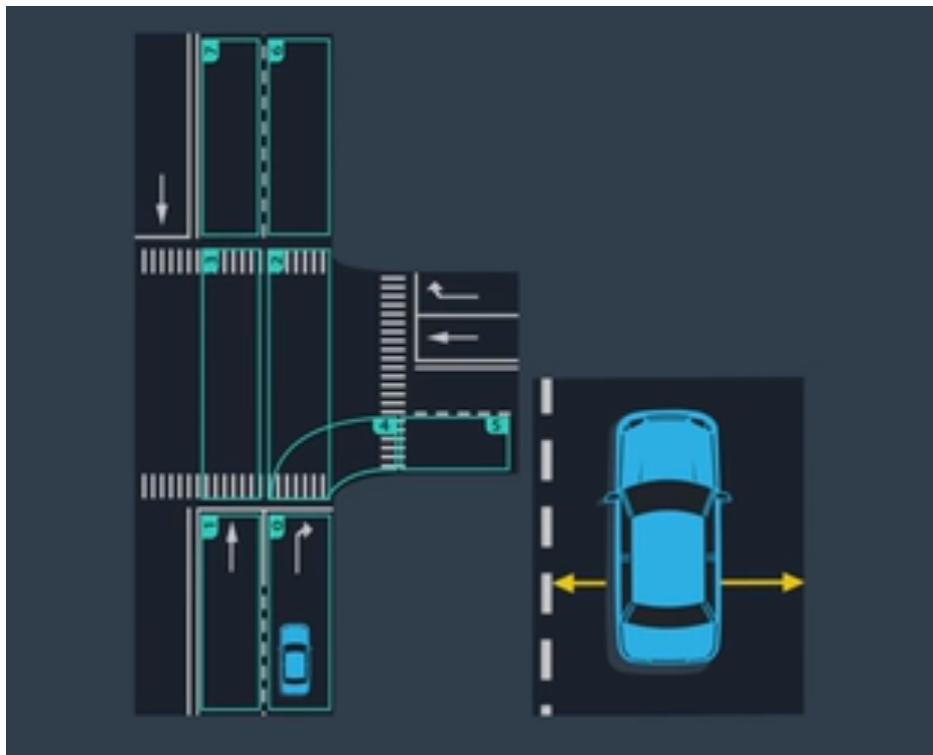
Apollo提供了一种基于车道序列的预测方法。为了建立车道序列，首先将道路分成多个部分，每个部分都覆盖了一个易于描述车辆运动的区域，我们更关心车辆如何在这些区域内转换。



### 障碍物状态

需要知道物体的状态。

当我们行驶时，作为人类，我们通过观察一个物体的朝向、位置、速度、加速度来预测它将要做什么。这同样也是无人驾驶汽车如何观察物体的状态。除了位置、速度、朝向、加速度之外，无人车还需要考虑车道段内物体的位置。例如，预测模块会考虑从物体到车道线段边界的纵向和横向距离。

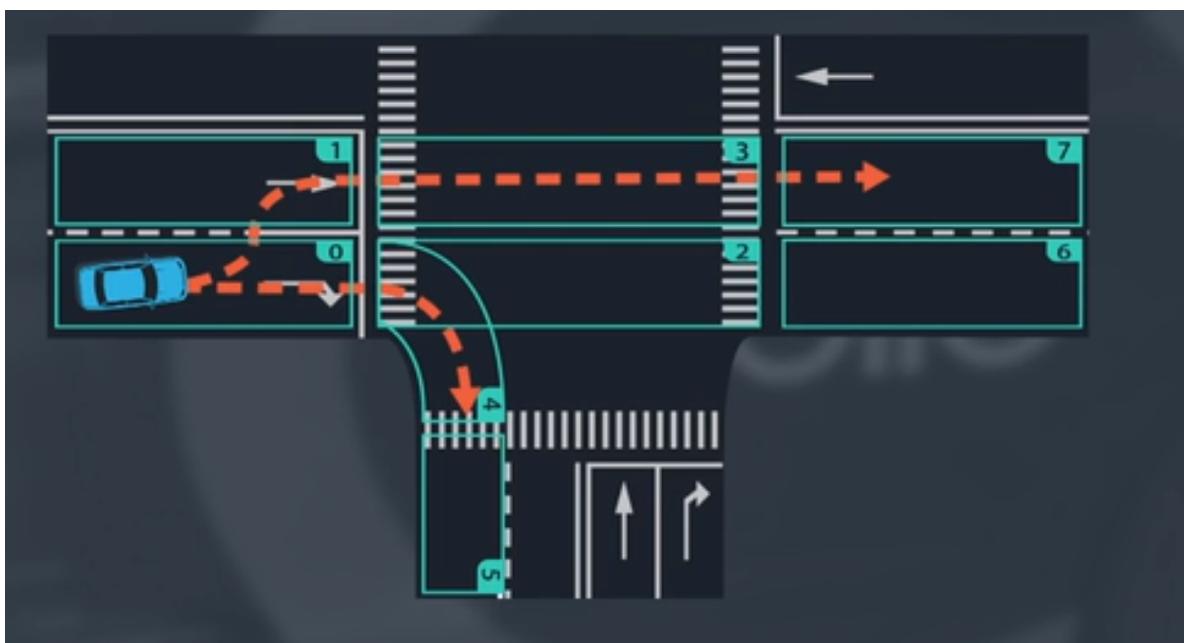


预测模块还包括之前时间间隔的状态信息，以便做出更准确的预测。

### 预测目标车道

使用车道序列框架的目标是：为道路上的物体生成轨迹。

我们会预测车道线段之前的过渡。



将预测简化成选择问题。

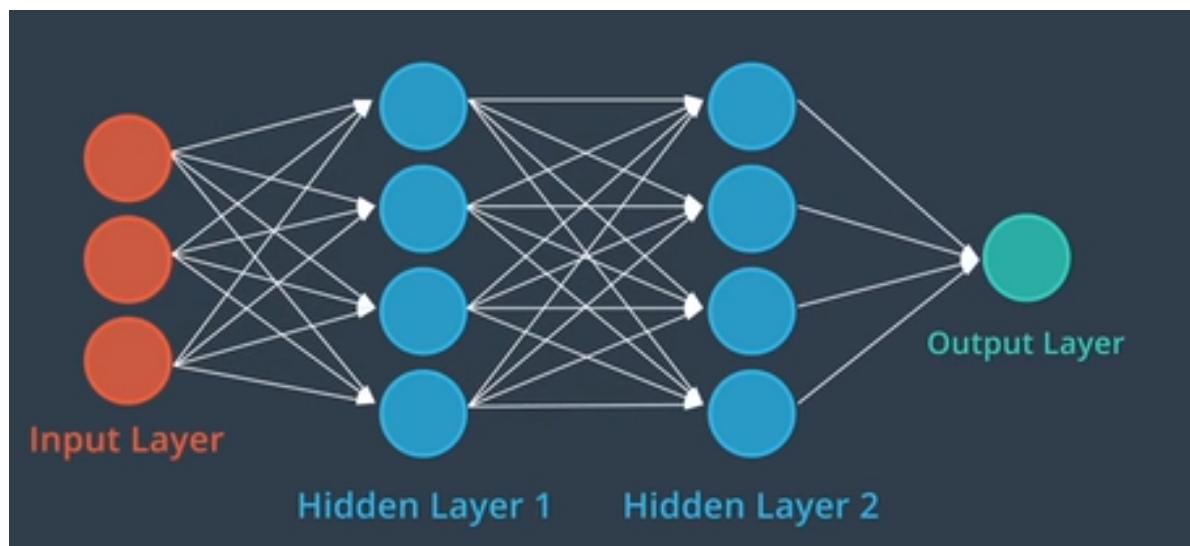
需要一个模型，将车辆状态和车道段作为输入，这个模型用于提供车辆可能采取每个车道序列的概率。同时希望模型能够学习新的行为，应该使用观测数据对模型进行经验型训练，在训练中，我们将真实的车辆行为提供给模型，不仅包括车道段和对象状态，还包括对象最终选择哪条车道序列，随着时间的增加，模型可以自我迭代更新，精度不断提升。每个记录将由观察对象跟随的车道段序列和对象的相关状态组成。

在每个时间点，对象占用并具有特定的状态，整个记录由一系列车道段和对象相关状态组成。

## 递归神经网络

递归神经网络或RNN是一种利用时间序列数据特征的一种预测方法。

神经网络是可训练的多层模型，神经网络从输入提取高级特征，并使用这些特征来计算得到输出。



例如有一个神经网络来分类图像是否包含汽车，网络的中间层将提取特征，如轮胎和窗户。

神经网络从数据中学习的方式叫做后向传播。

首先，神经网络得到输入并产生输出；然后，计算机比较输出和真值之前的误差，这种误差通过后向回到整个网络，中间的隐藏层根据观察到的这种差别调整其中的中间值或者叫权重，这可以提高神经网络的准确率。

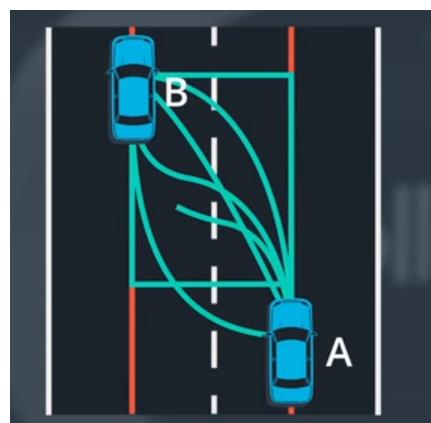
## 递归神经网络在目标车道预测的应用

Apollo使用RNN建立一个模型来预测车辆的目标车道。Apollo为车道序列提供一个RNN模型，为相关对象状态提供了另一个RNN模型，Apollo连接这两个RNN的输出并将它们馈送到另一个神经网络，该神经网络会估计每个车道序列的概率，具有最高概率的车道序列是我们预测车辆将遵循的序列。

## 轨迹生成

这是最后一步了，一旦预测到物体的车道序列，就可以预测到物体的轨迹。

在任何A、B两点，物体进行轨迹有无限的可能。



如何预测最有可能的轨迹呢？

先通过设置约束条件来去除大部分候选轨迹。首先假定汽车将与目标车道的中心对齐，继续去除车辆无法实际执行的轨迹；通过考虑车辆的当前速度和加速度从剩余的轨迹中进行选择。

## 第六课：规划

了解 Apollo 应用于无人驾驶车路径规划的几种不同方式。

在规划中，通过结合高精度地图定位和预测来构建车辆轨迹。

规划的第一步是路线导航，侧重于如何从地图上的A前往B，在进行路线规划时，将地图数据作为输入，并输出可行驶路径。在Apollo中，通过路线规划模块处理该任务。

轨迹规划的目标是生成避免碰撞和舒适的可执行轨迹，该轨迹由一些列点定义，每个点都有一个关联速度和一个指示如何抵达那个点的时间戳。

### 路由

路线规划的目标是从地图上的A点前往B点的最佳路径。

路线规划使用三个输入。

第一个输入为地图。Apollo提供的地图数据包括公路网和实时的交通信息；

第二个输入为我们当前在地图上的位置；

第三个输入为我们的目的地

有了这三个输入，路线规划模块就为寻找前往目的地的路径做好了准备。

### 世界到图

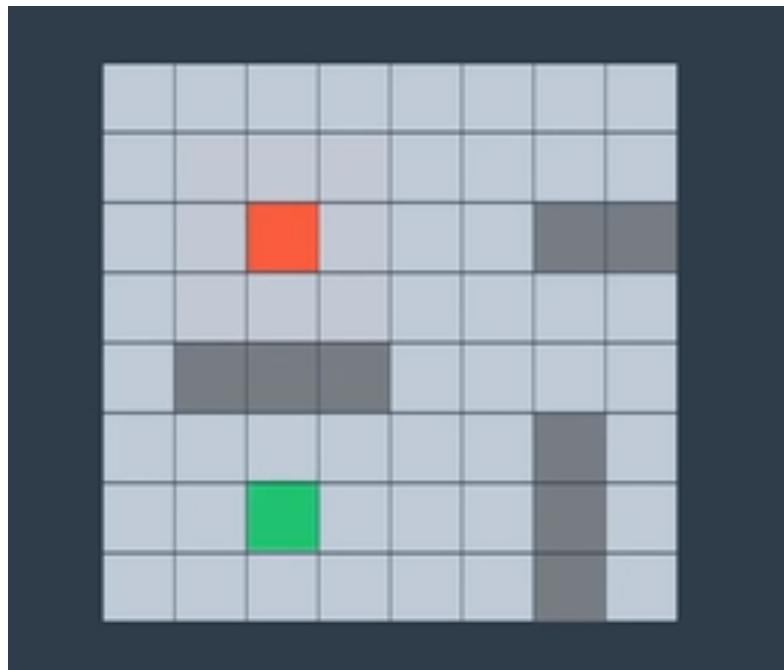
Apollo使用更智能的搜索算法。在开始搜索之前，它将地图数据重新格式化为“图形”的数据结构，该图形由“节点”和边缘组成；节点代表路段，边缘代表这些路段之间的选择。

在现实生活中，拐弯比直走要费劲。将地图转换为图形的好处在于，在计算机领域，人们已经发现许多用于在图形中查找路径的快速算法，一旦我们在图形中找到一条好的路径，就可以轻松地将图形中的路径重新转换为地图上的路径。

### 网格世界

A\*算法是经典的路径查找处理算法。

演示A\*如何通过网格工作。将网格中的每个单元当作一个节点，我们可以从任何一个节点移动到其任意相邻节点；网格中包含一些阻挡潜在路径的墙壁。



人类可以通过查看图形轻易的找到最佳路径，但是对于计算机，这并不明显；计算机必须检查是否存在通往目的地的任何路径，计算机尽可能竭尽所能尝试所有的可能路径，以找到最短的路径，但是，这需要耗费大量时间。

从初始节点开始，我们需要确定8个相邻的节点中哪个是最有希望的候选节点，对于每个候选节点，考虑两件事：一是计算从开始节点到候选节点的成本；二是估计从候选节点前往目的地的成本。计算前往候选节点的成本很容易，因为它与我们相邻。计算从候选节点到目的地的成本很困难。



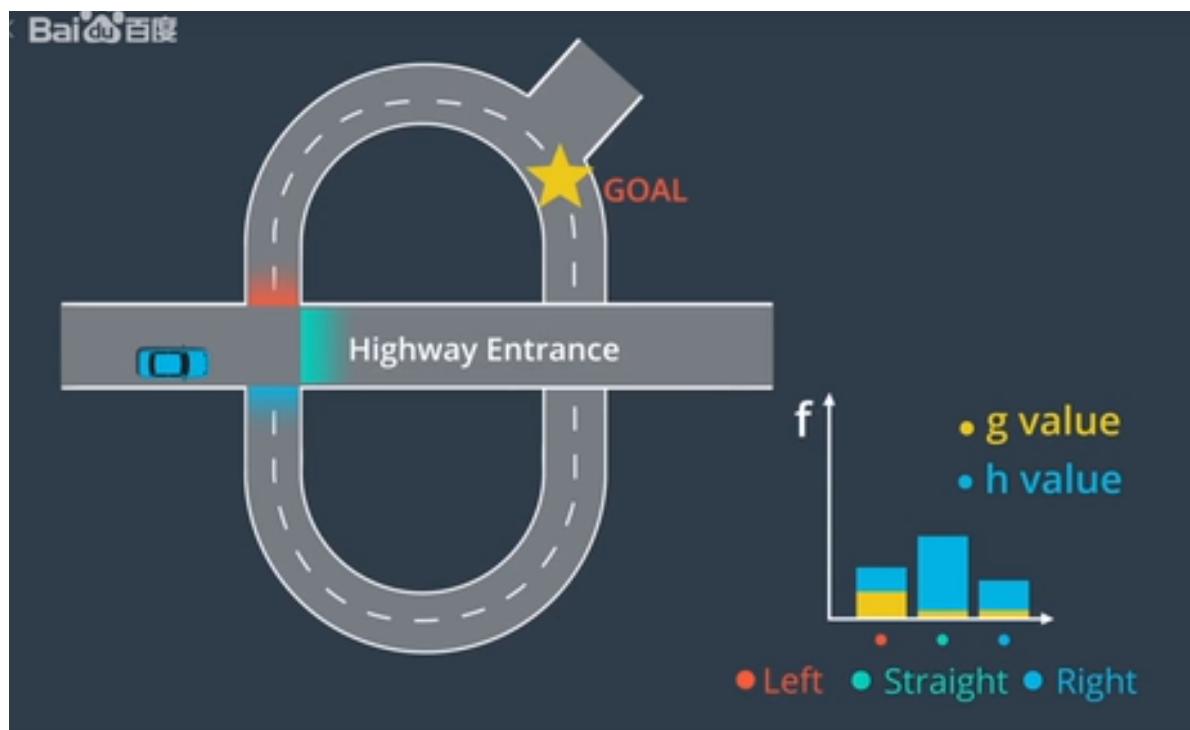
g:表示开始节点到候选节点的成本；

h:表示从候选节点到目的地的成本

通过添加g值和h值来计算总和，即f值。最佳候选节点是f值最小的节点。

每次到达新节点的时候，就重复此过程来选择下一个候选节点，而且总是选择我们尚未访问过的且具有最小值f的点。这就是A\*算法。

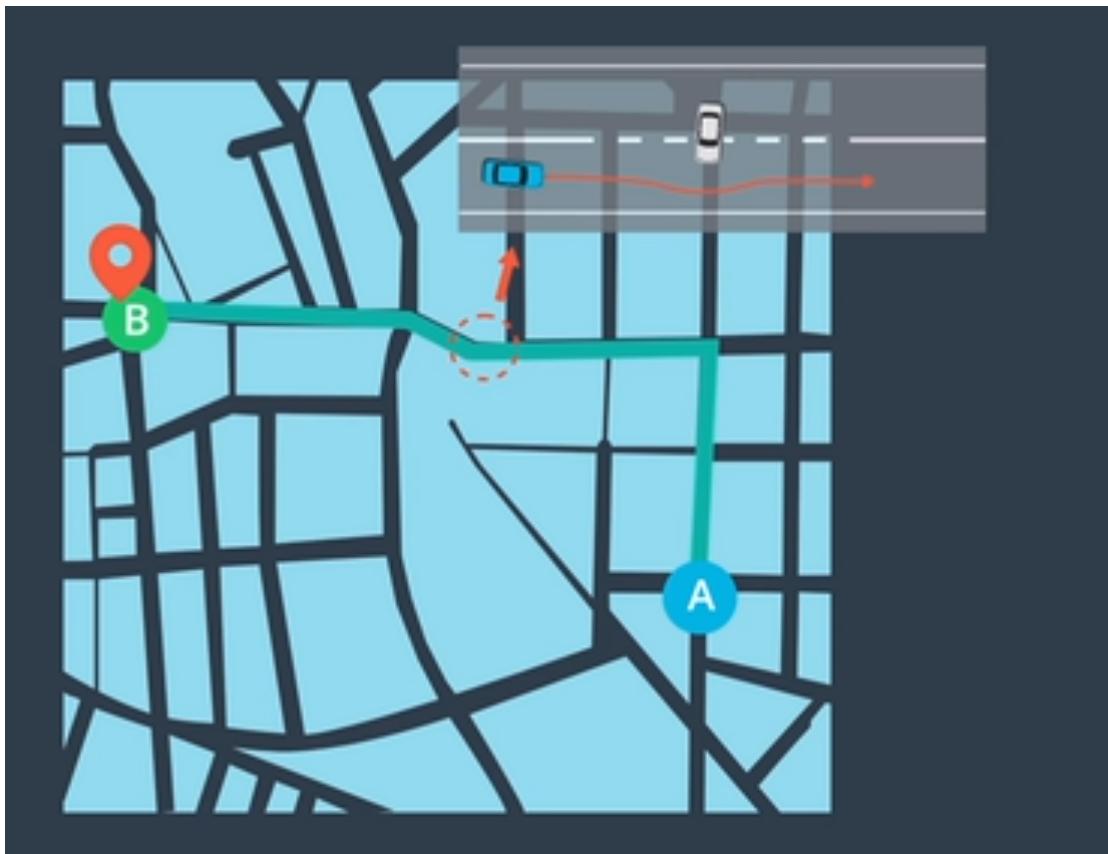
考虑真实世界中的地图。假设我们到达一个交叉路口，我们可以直走、右转或者左转。



首先，把这张图转换成具有三个候选节点的图形，接下来对选项进行评估，在实践中，拐过交叉路口很费劲，所以我们为这个节点分配了更高的g值（**g**是起始点到候选节点的成本）。路途较远的时候，我们要分配更高的h值（**h**值是从候选节点到目的地的估计成本）；接下来通过计算g值和h值相加来计算每个节点的f值。这张图中，最低f值实际对应的是右端候选节点，所以，这就是我们要前往的节点。

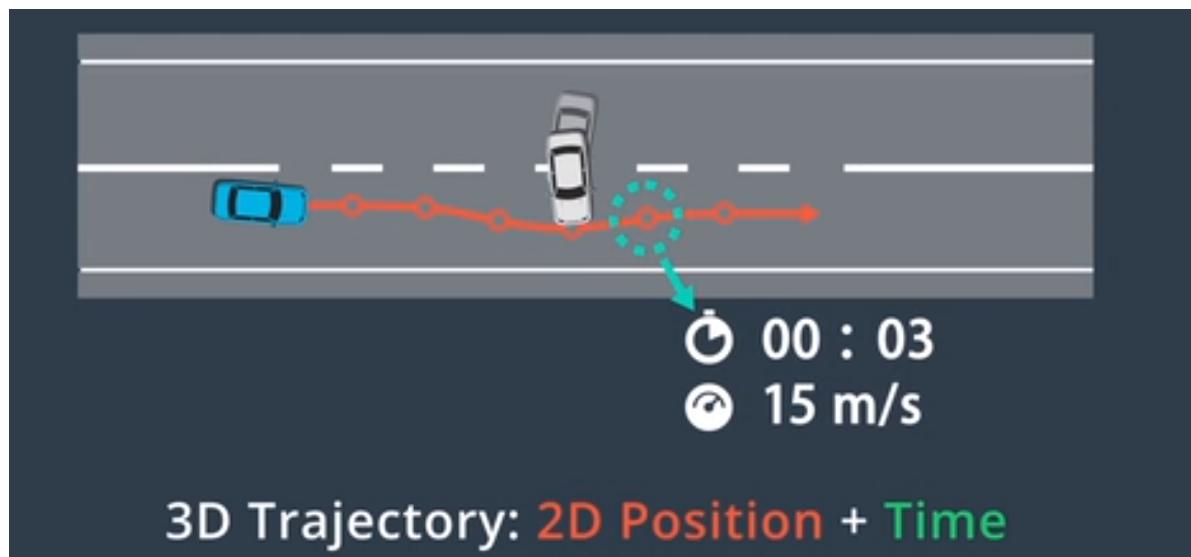
### 从路由到轨迹

高精地图路线只是规划过程的一部分，我们仍需要构建这条路线前进的低等级轨迹。这意味着处理不属于地图的东西（自行车、人等）。比如，我们可能需要与试图在我们前面掉头的汽车互动，这些场景需要更低级别、更高精度的规划。将这一级别的规划称为**轨迹生成**。



### 3D轨迹

轨迹生成的目标是生成由一系列路径点所定义的轨迹，我们为每个路径点分配了一个时间戳和速度，我们让一条曲线与这些路径点拟合，生成轨迹的几何表征，由于移动的障碍物可能会暂时阻挡部分道路，轨迹中的每个路点都有时间戳，我们可以将时间戳与预测模块的输出相结合，以确保在我们计划通过时，轨迹上的每个路径点均未被占用



这些时间戳创建一个三维轨迹，每个路径点由空间中的两个维度以及时间上的第三个维度来定义。

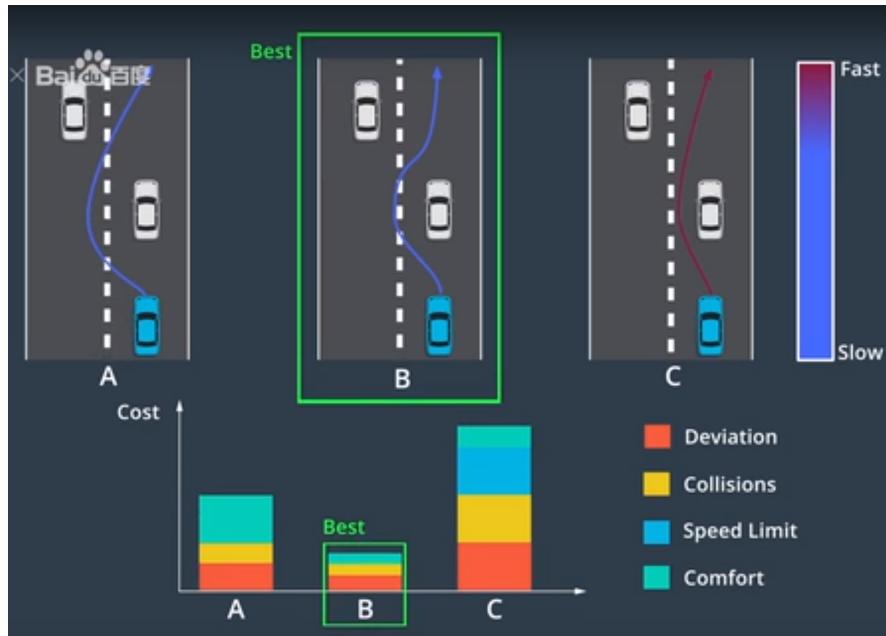
### 评估一条轨迹

首先，轨迹要能够免于碰撞；其次要让乘客感到舒适，所以路径之间的过度及速度的任何变化都必须平滑；

再者，路径点对于车辆应该实际可行；例如高速行驶的汽车不能立即做180°转弯。

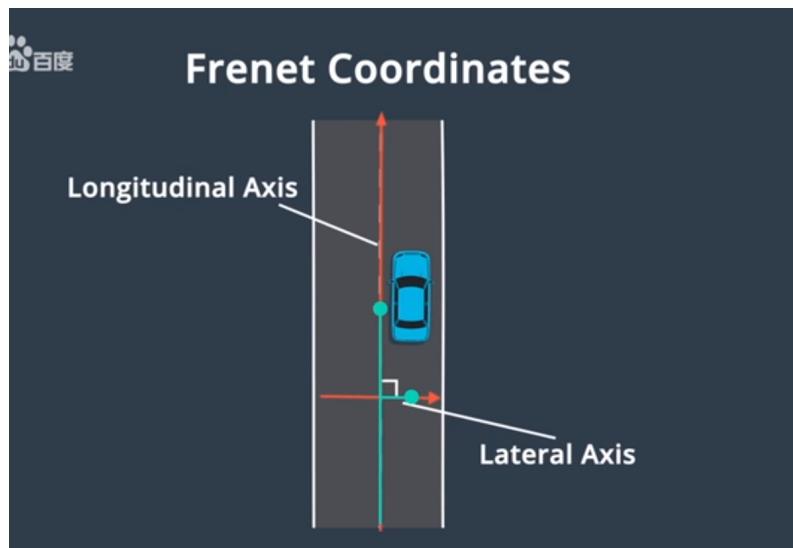
最后，轨迹应该合法，需要了解每个路径点的交通法律，并确保轨迹遵循这些法律法规。

如何选择最佳路径呢？使用“成本函数”



### Frenet坐标

通常使用笛卡尔坐标系描述物体的位置，但是笛卡尔坐标系并不是最佳选择，即使给出了车辆的位置 $(x,y)$ ，如果我们不知道道路在哪里，也很难知道车辆行驶了多远，也难以确定它是否偏离车道中心。用Frenet坐标系代替。Frenet描述了汽车相对于道路的位置。



纵坐标表示在道路中的行驶距离；横坐标表示汽车偏离中心线的距离。

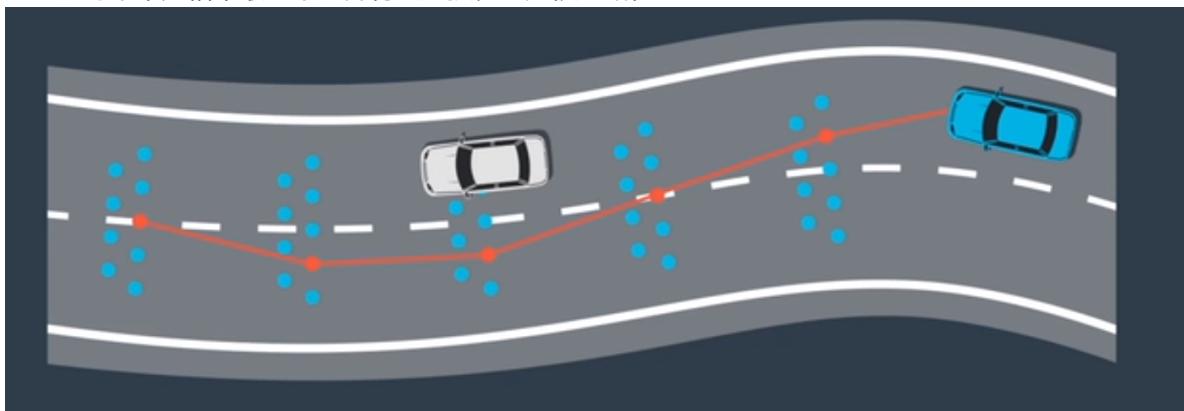
### 路径-速度解耦规划

路径-速度解耦规划将轨迹规划分为两步：路径规划和速度规划。

在路径规划中，生成候选曲线，这是车辆可行驶的路径，使用成本函数对每条路径进行评估，按照成本路径进行排名，选择最低成本的路径；下一步是确定这条路线行进的速度。选择的是曲线速度，而不是单个速度。将路径和曲线速度相结合，可以构建车辆行驶的轨迹。

### 路径生成与选择

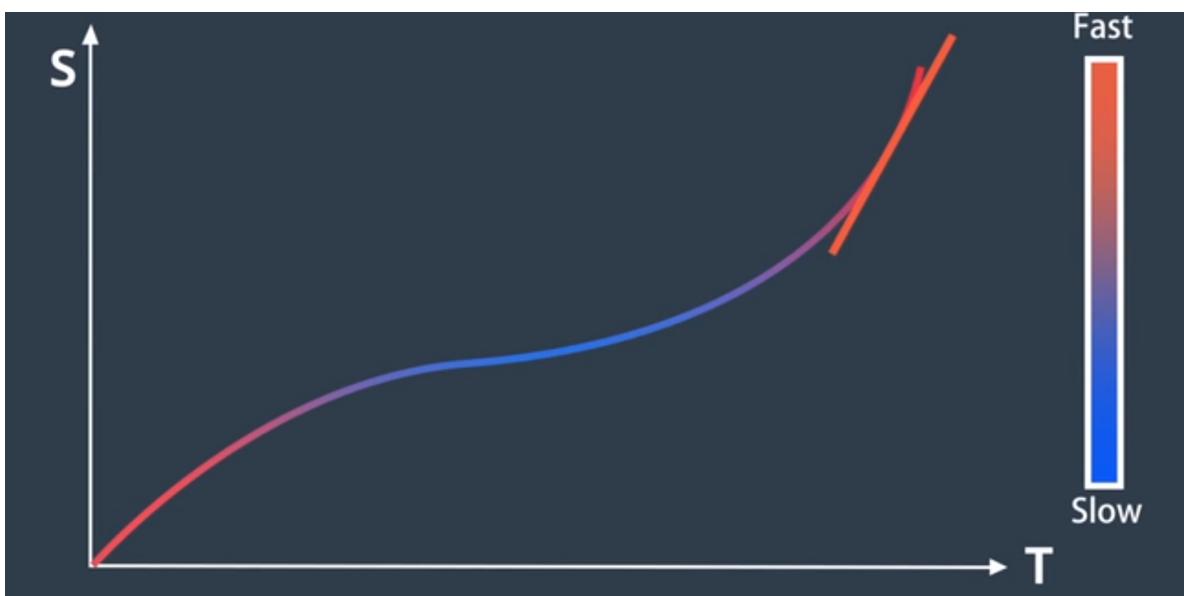
在路径-速度解耦规划中生成候选路径。首先将路径分割成单元格，然后对单元格中的点进行随机采样，通过从每个单元格中取一个点并将点连接，创建候选路径。



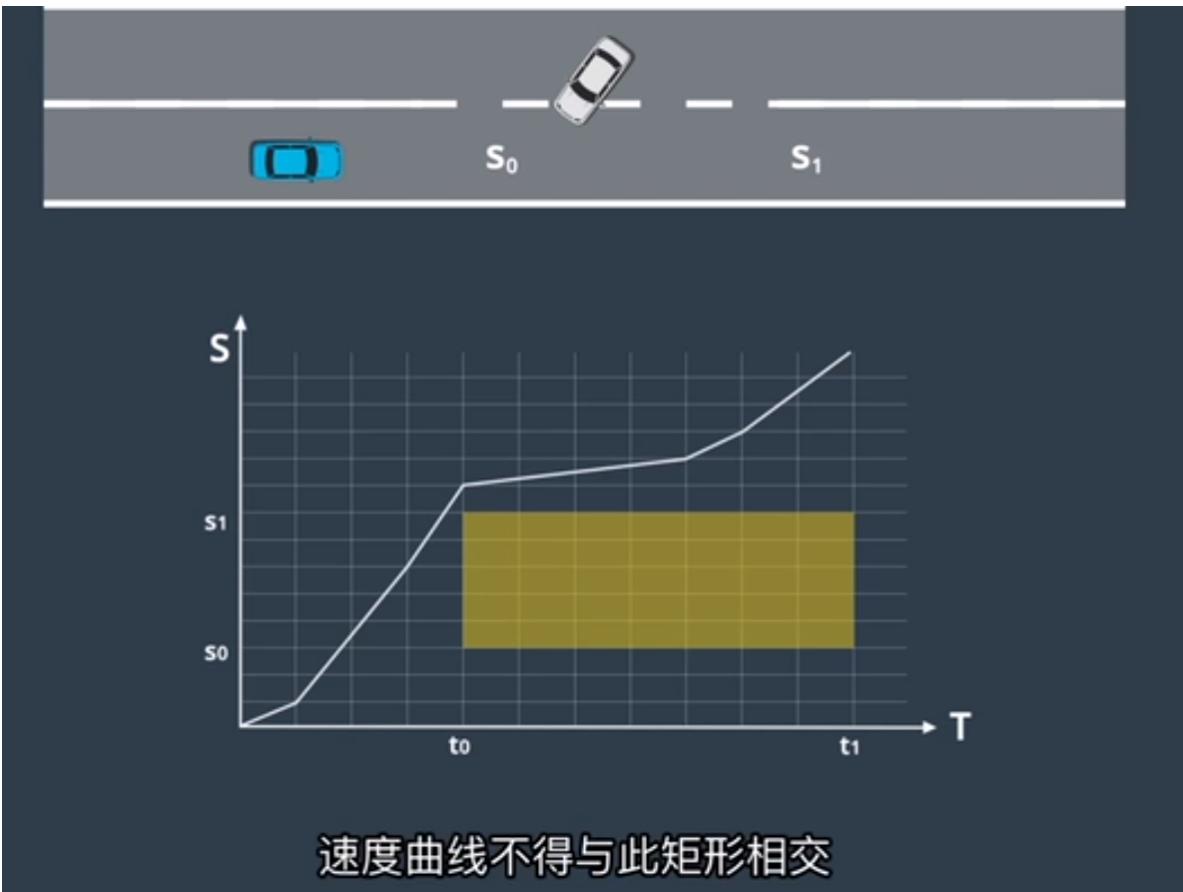
重复此过程，可以构建多个候选路径，使用成本函数对这些路径进行评估，并选择成本最低的路径。

### ST图

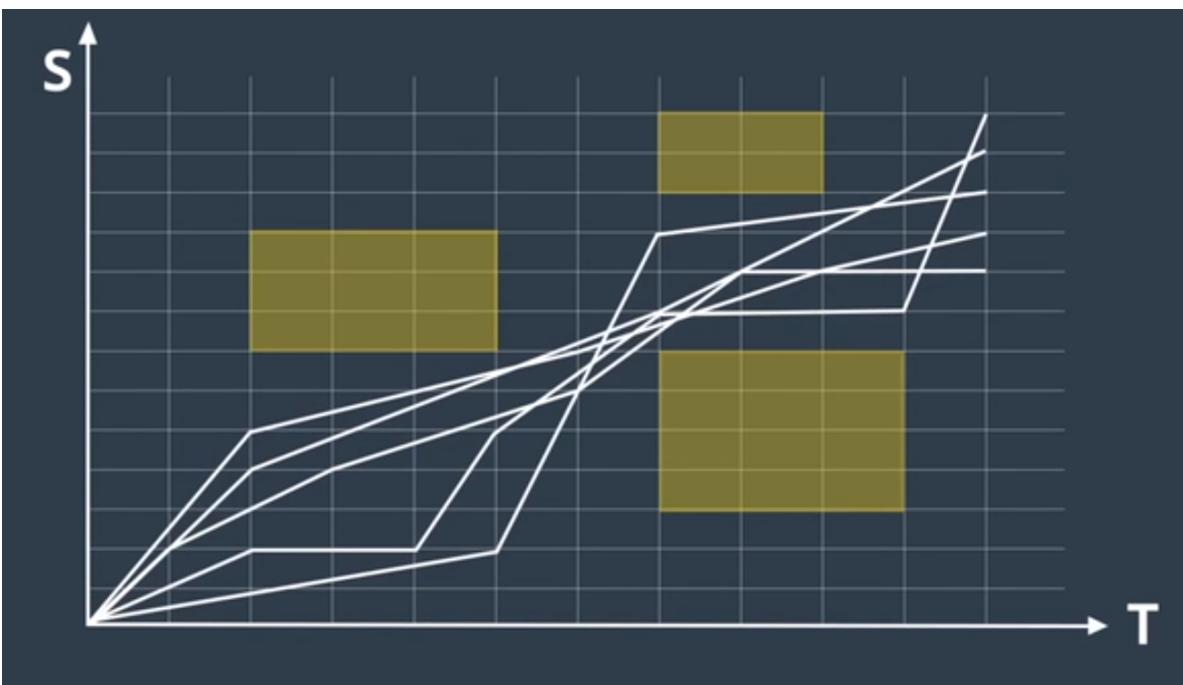
路径选择后的下一步是选择与该路径关联的速度曲线，一个被称为“ST”图的工具可以帮助我们设计和选择速度曲线。



### 速度规划



在实际中更复杂



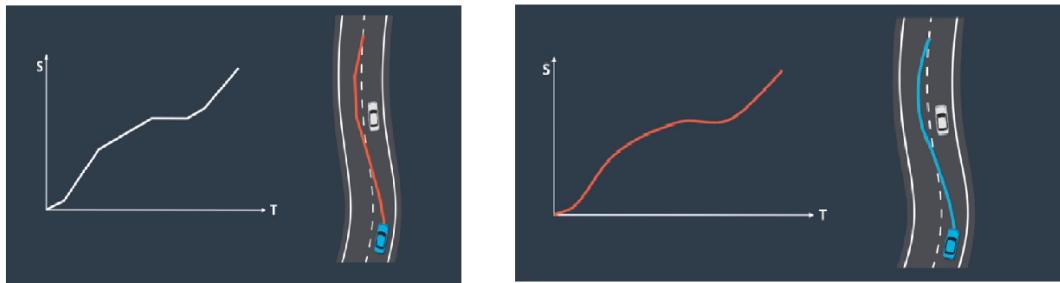
使用优化引擎为该图选择最佳的速度曲线。

### 优化

路径选择涉及将道路划分为单元格，速度曲线构建涉及将ST图划分为单元格。

该解决方案生成的轨迹并不平滑，为了将离散解决方案转换成平滑轨迹，我们可以使用“二次规划”技术。

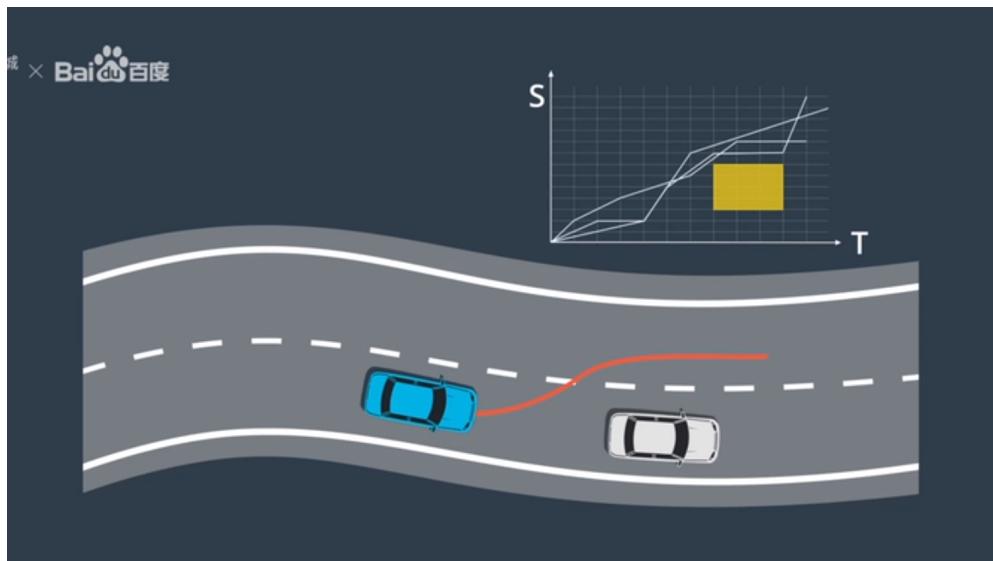
二次规划将平滑的非线性曲线与这些分段式线性段拟合。



一旦路径和速度曲线就绪，可以使用其构建三维轨迹。

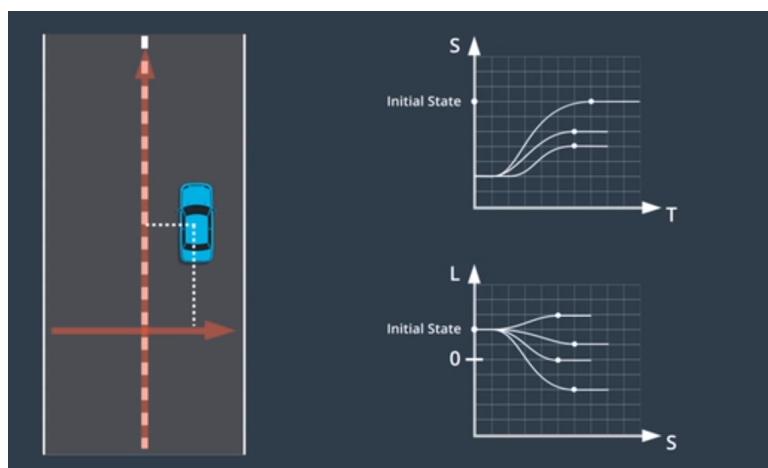
### 路径-速度规划的轨迹生成

假设我们正在道路上行驶，感知系统观察到，一辆缓慢行驶的车辆离我们越来越近。首先，我们在这辆车的周围生成多条候选路径，使用成本函数对这些候选路径进行评估，并选择成本最低的路径；然后使用ST图来进行速度规划，根据其他车辆随时间变化的位置阻挡了ST图的部分区域；优化引擎可以帮助我们确定该图的最佳速度曲线。该曲线受制于约束和成本函数。我们可以使用二次规划让路径和速度曲线变平滑；最后将路径和速度曲线合并构建轨迹。



### Lattice规划

通过使用Frenet坐标，我们可以将环境投影到纵轴和横轴上。我们的目标是生成三维轨迹-纵向维度、横向维度、时间维度。我们可以将三维问题分解成两个单独的二维问题，这是通过分离轨迹的纵向和横向分量来解决的。其中一个二维轨迹是具有时间戳的纵向轨迹，称之为ST轨迹；另一个二维轨迹是相对于纵向的横向偏移，称为SL轨迹。



Lattice规划有两个步骤：即分别建立ST和SL轨迹，然后将它们合并。

为生成纵向和横向二维轨迹，先将车辆的初始状态投射到ST坐标系和SL坐标系中，通过对预选模式中的多个候选最终状态进行采样来选择车辆最终状态。

对于每个候选状态，构建一组轨迹，将我们的车辆从其初始状态转换成最终状态，使用成本函数对这些轨迹进行评估，选择成本最低的轨迹。

### ST轨迹的终止状态

根据情况，可以将状态分成3组：巡航、跟随、停止。

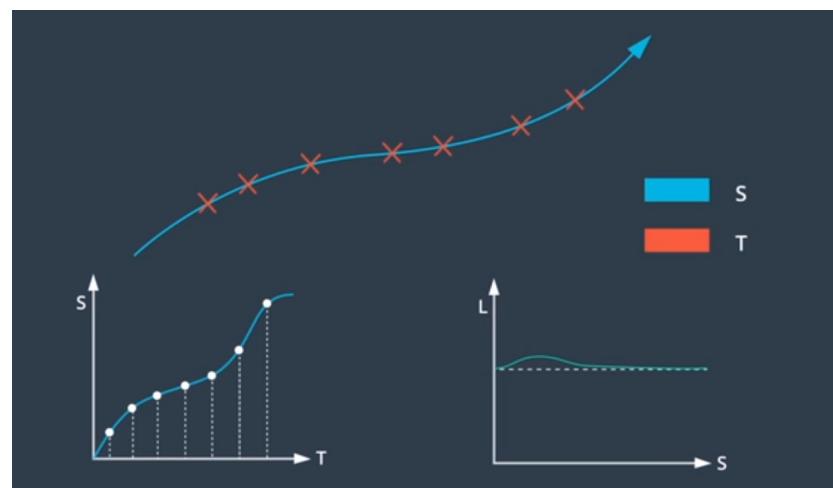
**巡航** 意味着车辆将在完成规划步骤后定速行驶

**跟随车辆** 对位置和时间状态进行采样，并尝试在时间t出现在某辆车后面，在跟随车辆时，我们需要与前方的车保持安全距离，这时速度、加速度将取决于我们要跟随的车辆。

最后一种模式是停止。只需对汽车何时何地停止进行抽样。这里速度、加速度被修正为0。

### Lattice规划的轨迹生成

一旦我们有了ST和SL轨迹，我们需要将它们重新转换成笛卡尔坐标系。然后，可以将它们相结合，构建由二维路径点和一位时间组成的三维轨迹。



## 第七课：控制

了解无人驾驶是如何使用方向盘、油门和刹车来执行我们规划好的轨迹，并掌握 Apollo 中不同类型的控制器。

学习如何使用控制来运行轨迹。控制是驱使车辆前行的策略，对汽车而言，最基本的控制为转向、加速和制动。

通常，控制器使用一系列路径点来接受轨迹。控制器的任务是使用控制输入，让车辆通过这些路径点。

首先，控制器必须准确，避免偏离目标轨迹；其次，控制策略对汽车应该具备可行性；最后，考虑平稳度。

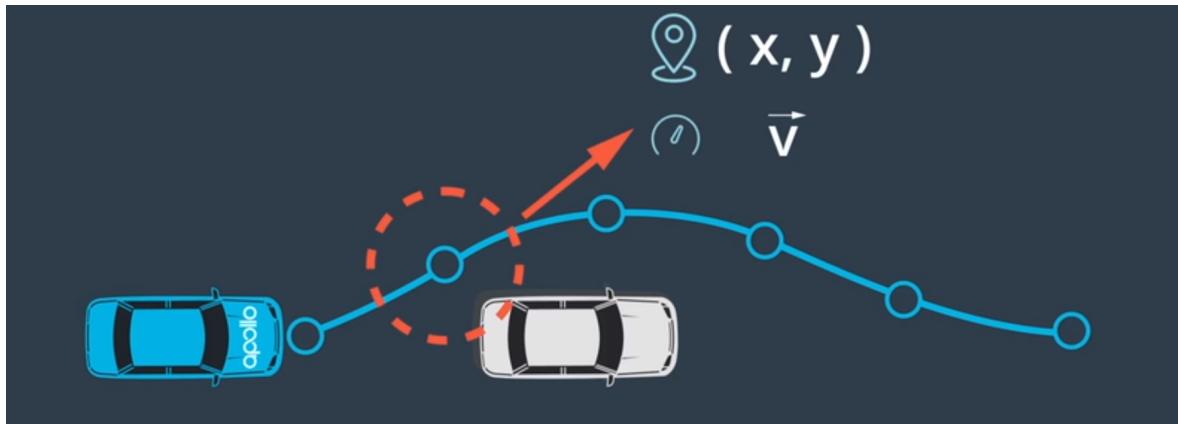
可实现这些目标的三种控制策略：

- (1) 比例积分微分控制 (PID)；
- (2) 线性二次调节器 (LQR)；
- (3) 模型预测控制 (MPC)。

## 控制流程

控制器预计有两种输入：目标轨迹和车辆状态

目标轨迹来自规划模块，在每个规划点处，规划模块指定一个位置和参考速度。每个时间步都对轨迹进行更新



还需要了解车辆状态，其中包括车辆位置（通过本地化模块来计算）、加速度、转向、速度、

使用这两个输入来计算目标轨迹与实际行进轨迹之间的偏差。

控制器输出的是控制输入（转向、加速和制动）的值。当偏离目标轨迹时，希望采取行动来纠正这种偏差。

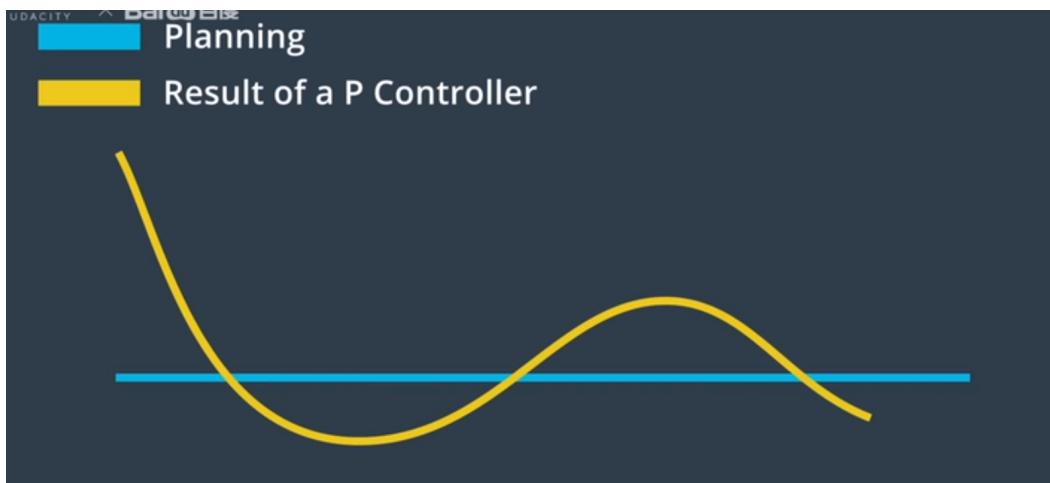
### PID控制

比例积分微分控制（PID）

优点在于它非常简单。只需要知道与目标轨迹有多大偏离。

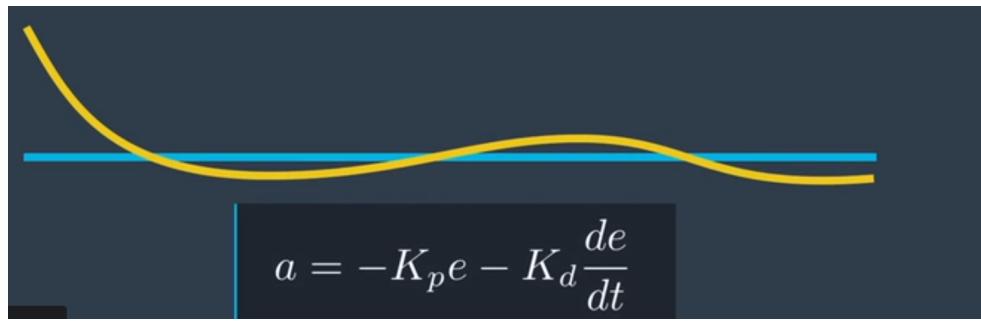
PID的第一个组件为P，代表比例（proportional），P控制器在车辆开始偏离时立即将其拉回目标轨迹，比例控制意味着车辆偏离越远，控制器就越难将其拉回目标轨迹。

在实践中P控制器的一个问题在于，它很容易超出参考轨迹。



当车辆越来越接近目标轨迹时，需要控制器的更加稳定，PID中的D致力于使运动状态处于稳定状态

D代表微分（derivative），PD控制器类似于P控制器，它增加了一个阻尼项，可最大限度地减少控制输出器输出的变化速度。



PID控制器中的最后一项I表示积分 (Integral) , 该项目负责纠正车辆的任何系统性偏差。例如, 转向可能失准, 这可能造成恒定的转向偏移, 在这中情况下, 需要稍微向一侧转向以保持直行; 为了解决这一问题, 控制器会对系统累积误差进行惩罚。我们可以将P、I、D组合, 构成PID。

### PID优劣对比

PID是一种线性算法, 对于复杂的系统而言, 这是不够的。对于无人车而言, 需要应用不同的PID控制器来控制转向、加速, 这意味着很难将横向和纵向控制结合起来; 另一个问题在于PID控制器依赖于实时误差测量, 这意味着受到测量延时限制时可能会失效。

### 线性二次调节器

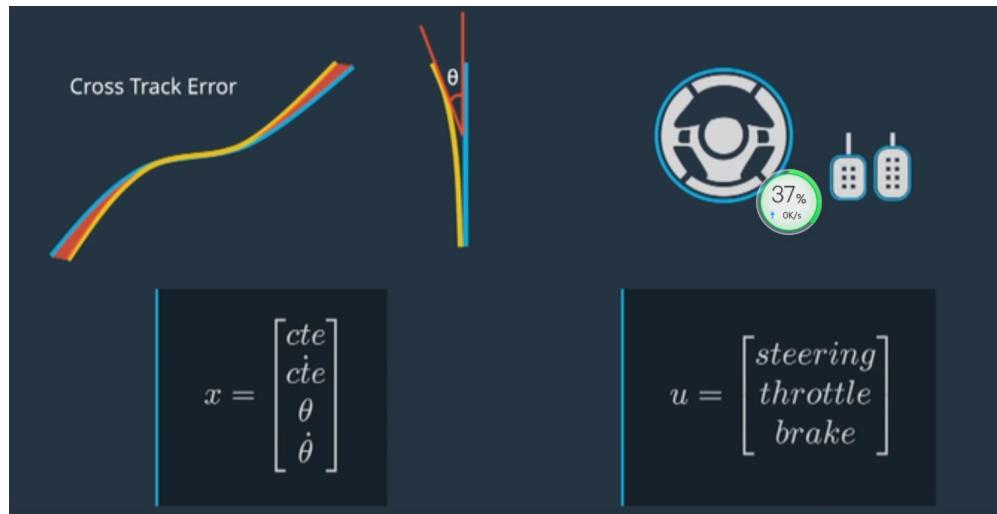
线性二次调节器 (LQR) 是基于模型的控制器, 它使用车辆的状态来使误差最小化。

Apollo使用LQR进行横向控制。

横向控制包括四个组件: 横向误差、横向误差的变化率、朝向误差、朝向误差的变化率。

【变化率与导数相同】用变量名上加一点来表示。

称这四个组件的集合为x。这个集合x捕获车辆的状态, 除了状态之外, 该车还有三个控制输入: 转向、加速、制动。将这个控制输入集合称为u。



LQR处理线性控制, 这种类型的模型可以用等式来表示:

$$\dot{x} = Ax + Bu$$

x (上方带点) 向量是导数或者x向量的变化率, 所以x点的每个分量只是x相应分量的导数。

$$\begin{bmatrix} \dot{cte} \\ \ddot{cte} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = A \begin{bmatrix} cte \\ \dot{cte} \\ \theta \\ \dot{\theta} \end{bmatrix} + B \begin{bmatrix} steering \\ throttle \\ brake \end{bmatrix}$$

这个等式是线性的。因为当我们用 $\Delta x$ 来改变 $x$ 时，并用 $\Delta u$ 来改变 $u$ ，等式依然成立。

$$\dot{x} + \Delta \dot{x} = A(x + \Delta x) + B(u + \Delta u)$$

$$\Delta \dot{x} = A\Delta x + B\Delta u$$

这是对对于LQR中的L的理解。

那么对于Q呢？这里的目标就像控制目标一样典型，是为了让误差最小化，但我们也希望尽可能少地使用控制输入，由于使用这些会有成本；为了减少这些因素，我们可以保持误差的运行总和和控制输入的运行总和；例如：当汽车往右偏转得特别厉害之际，添加到误差总和中；当控制输入将汽车往左侧转时，从控制输入总和中减去一点。当然，这种方法会有问题，因为右转的正误差不一定消除左转的负误差。可以让 $x$ 乘 $u$ ，这样负值也会产生正平方，称这些为二次项。

为这些项分配权重，并将它们加到一起，最优的 $u$ 应该加倍减总和，在数学中，我们将这个值称为成本函数。

$$cost = \int_0^{\infty} (x^T Q x + u^T R u) dt$$

这里 $Q$ 和 $R$ 代表 $x$ 和 $u$ 的权重集合， $X_t$ 和 $U_t$ 是转置矩阵，这意味着它们几乎与 $x$ 和 $u$ 相同，只是重新排列，以便矩阵乘法。

在LQR中，控制方法被描述成 $u = -Kx$ ， $K$ 是一个复杂的skeme 【安全密钥交换机制】，代表如何从 $x$ 计算出 $u$ ，所以找到一个最优的 $u$ 就是找到一个最优的 $K$ ，许多工具都可以轻松解决 $K$ .

### 模型预测控制

#### 模型预测控制 (MPC)

是一种更复杂的控制器，它非常依赖于数学优化，基本上可以将MPC归结为三个步骤：

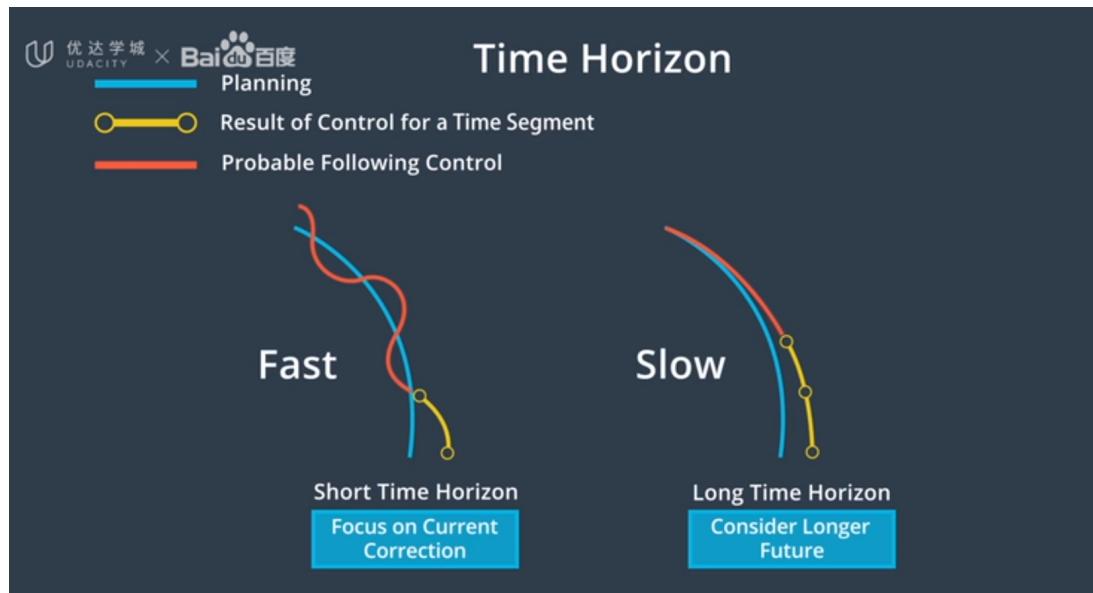
- (1) 建立车辆模型；
- (2) 使用优化引擎计算有限时间范围内的控制输入；
- (3) 然后执行第一组控制输入。

MPC是一个重复的过程，计算一系列控制输入，并优化该序列，但是控制器实际上只实现了序列中的第一组控制输入，然后控制器再次重复该循环。那么为什么不执行整个控制输入序列呢？因为只是采用了近似测量与计算。

## 时间范围与车辆模型

MPC的第一步为定义车辆模型，该模型近似于汽车的物理特性，特别是，该模型估计了假如控制输入应用于车辆时会发生什么。

接下来，我们决定MPC预测未来的能力，预测越深入，控制器就越精确，不过需要的时间也越长。



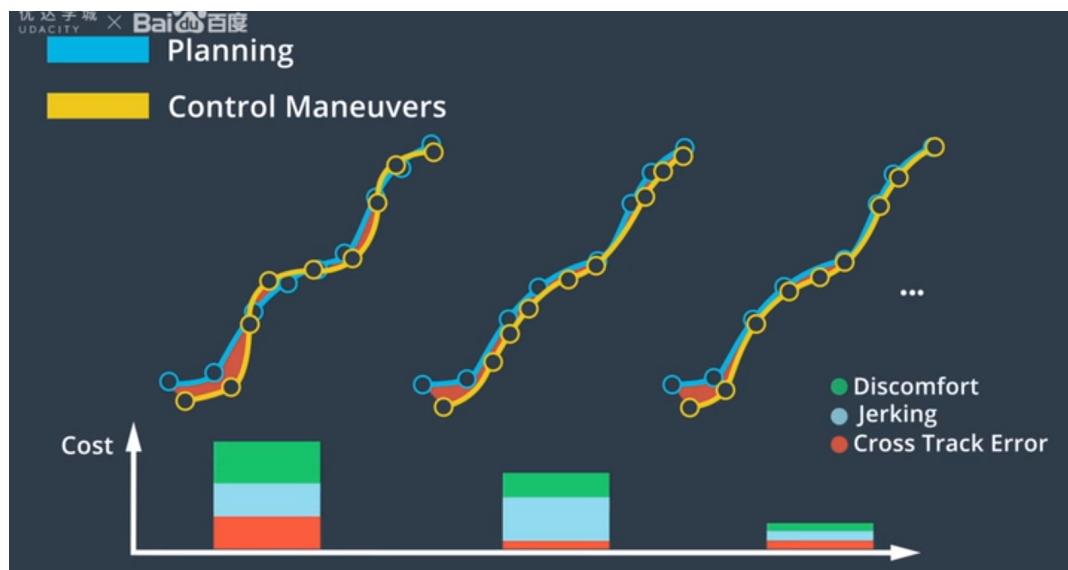
所以，我们需要在准确度与快速获取结果之间做出取舍。

获取结果的速度越快，越能快速地将控制输入应用到实际车辆中。

下一步是将模型发送到搜索最佳控制输入的优化引擎。该优化引擎的工作原理是通过搜索密集数学空间来寻找最佳解决方案，为缩小搜索范围，优化引擎依赖于车辆模型的约束条件。

## MPC优化

优化引擎可间接评估控制输入。可根据成本函数对轨迹进行评估。



成本函数主要基于与目标轨迹的偏差，其次基于因素：加速度、提升乘客舒适度的措施。

为使乘客感到更舒适，对控制输入的调整应该很小

## MPC优劣对比

模型预测控制考虑了车辆模型，因此比PID控制更精确，它也适用于不同的成本函数

另一方面，与PID相比，模型预测控制相对更加复杂、更缓慢、更加难以实现。

在实践中，无人驾驶车的控制可扩展性的重要程度通常意味着值得为MPC投入现实成本，所以MPC称为一个非常重要的无人驾驶车控制器。

## 总结

自动驾驶汽车依靠**人工智能、视觉计算、雷达、监控装置和全球定位系统**协同合作，让计算机可以在没有任何人类主动的操作下，自动安全地操作机动车辆。

核心点：

- (1) 高精度地图：几乎支持软件栈的所有其他模块。
- (2) 定位：讨论汽车如何确定它所处的位置，利用激光和雷达数据，将这些传感器感知内容与高分辨率地图进行对比，这种对比使汽车自定定位。
- (3) 感知：了解无人驾驶车如何感受这个世界（周围的物体是什么）；深度学习；卷积神经网络。
- (4) 预测：几种不同的方式用于预测其它车辆或行人可能如何移动；神经递归网络等。
- (5) 规划：如何将预测和规划相结合以生成车辆轨迹。
- (6) 控制：如何使用转向、油门和制动来执行规划轨迹。