# 1 自动驾驶简介

我们正处在自动驾驶未来的黎明。为了探究自动驾驶的未来，让我们一起回顾其历史吧。

信息技术的开端真正开始于20世纪60年代，当时仙童半导体和英特尔通过生产微处理器奠定了信息技术的基础层，作为副产品，他们创造了硅谷。虽然微处理器技术极大地提高了我们的生产力，但普通大众接触这些技术的途径有限。20世纪80年代，微软和苹果通过引入图形用户界面，奠定了信息技术的第二层，实现了“每个家庭都有一台Mac电脑”的愿景。在本世纪头十年，以谷歌为代表的互联网公司通过将人与信息连接起来，奠定了信息技术的第三层。例如，通过谷歌，可以间接连接信息提供者和信息使用者。然后在21世纪10年代，像Facebook和LinkedIn这样的社交网络公司，通过有效地将人类社会转移到互联网，奠定了信息技术的第四层，并允许人们直接相互连接。在互联网人类社会的人口规模达到一定规模后，在2015年前后，Uber和Airbnb的出现，通过在互联网人类社会上提供服务，奠定了信息技术的第五层，形成了互联网商业社会。虽然Uber和Airbnb为我们提供了通过互联网高效接入服务提供商的手段，但这些服务仍然是由人类提供的。

## 自动驾驶技术综述

如图1.1所示，自主驾驶不是单一的技术，而是由多个子系统组成的高度复杂的系统。主要分为三个部分:算法，包括传感、感知和决策(这需要对复杂的情况进行推理);客户端系统，包括操作系统和硬件平台;以及云平台，包括高精度地图、深度学习模型训练、仿真、数据存储等。

算法子系统从传感器原始数据中提取有意义的信息，以了解其所处的环境，并对其未来的行动做出决策。客户端系统将这些算法集成在一起，以满足实时性和可靠性的要求。例如，如果摄像机以60Hz的频率生成数据，客户端系统需要确保处理管道的最长阶段需要少于16毫秒才能完成。云平台为自动驾驶汽车提供了离线计算和数据存储。通过云平台，我们可以测试新的算法，更新高清地图，训练更好的识别、跟踪和决策模型。

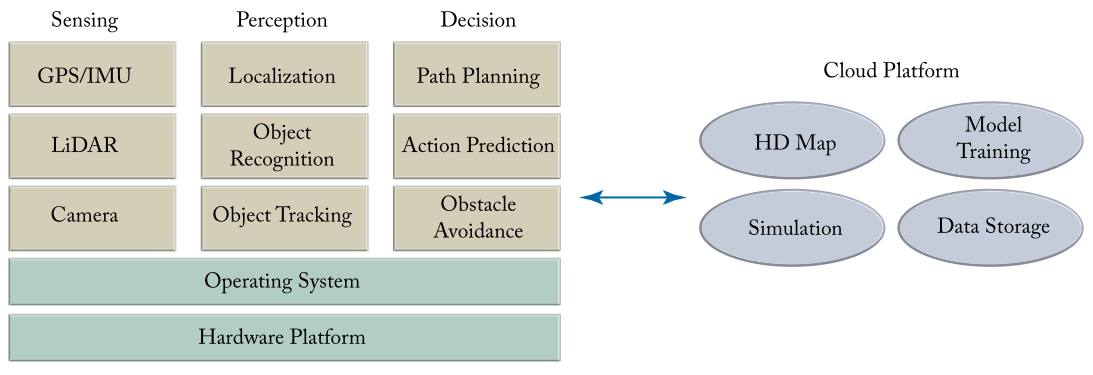


图1.1 自动驾驶系统技术概览

## 自动驾驶算法

算法部分包括：传感，从传感器或原始数据中提取有意义的信息;感知，定位车辆和了解当前环境;决策，换句话说，采取行动，以可靠和安全地到达目标目的地。

### 传感

通常，自动驾驶汽车由几个主要的传感器组成。事实上，由于每种类型的传感器都有各自的优点和缺点，在自动驾驶汽车中，为了提高可靠性和安全性，必须将来自多个传感器的数据结合起来。如下传感器：

1）GPS/IMU: GPS/IMU系统通过报告惯性更新和高速率(如200Hz)的全球位置估计，帮助自动驾驶车辆定位。GPS是一种相当精确的定位传感器，但是它的更新速度很慢，大约只有10Hz，因此不能提供实时更新。然而，IMU误差随时间累积，导致位置估计的相应降低。尽管如此，IMU可以更频繁地提供更新，频率在200Hz或更高。这应该满足实时需求。将GPS和IMU相结合，可以为车辆定位提供准确、实时的更新。

2）激光雷达:激光雷达用于测绘、定位和避障。它的工作原理是将光束反射到物体表面，然后测量反射时间来确定反射度。由于其高精度，激光雷达可以用来制作高清地图，在高精度地图上定位移动的车辆，检测前方障碍物等。通常情况下，激光雷达单元，如Velodyne 64光束激光器，以10Hz的频率旋转，每秒将读取130万个数据。

3）摄像头:摄像头主要用于目标识别和目标跟踪任务，如车道检测、交通灯检测、行人检测等。为了提高自动驾驶汽车的安全性，现有的方案通常在汽车周围安装8个或更多的1080p摄像头，这样我们就可以使用摄像头来检测、识别和跟踪汽车前后和两侧的物体。这些相机通常以60Hz的频率工作，如果两者结合起来，每秒将产生大约1.8 GB的原始数据。

4）雷达和声纳:雷达和声纳系统主要用于避障的最后一道防线。由雷达和声纳产生的数据显示距离车辆路径前方最近的物体的距离和速度。一旦我们发现前方不远处有物体，就可能有碰撞的危险，那么自动驾驶汽车就应该刹车或转弯以避开障碍物。因此，雷达和声纳产生的数据不需要太多的处理，通常直接提供给控制处理器，而不是通过主计算管道，以实现诸如转向、刹车或预紧安全带等“紧急”功能。

### 1.2.2 感知

传感器数据随后被输入感知阶段，以提供对车辆环境的理解。自主驾驶感知的三个主要任务是定位、目标检测和目标跟踪。

GPS/IMU可用于定位，如前所述，GPS提供了相当精确的定位结果，但更新率相对较低，而IMU提供非常快的更新，但代价是结果的准确性较低。因此，我们可以使用卡尔曼滤波技术来结合两者的优点，提供准确和实时的位置更新。如图1.2所示，其工作原理如下:IMU每隔5 ms更新一次车辆的位置，但是错误会随着时间的推移而累积。幸运的是，每100毫秒，一个GPS更新被接收，这有助于纠正IMU的错误，通过运行这个传播和更新模型，GPS/IMU组合可以产生快速和准确的定位结果。然而，我们不能仅仅依靠这种组合来定位，原因有三:(1)精度只有一米左右;(2) GPS信号存在多路径问题，意味着信号可能会从建筑物上反弹，从而带来更多的噪声;(3) GPS一览无余的天空，因此在隧道等环境中无法工作。

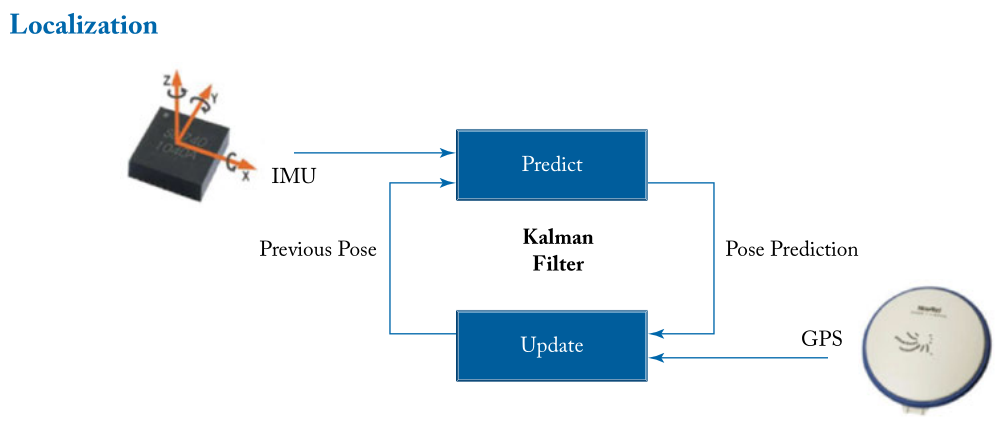


图1.2 GPS/IMU定位

摄像头也可以用于定位。基于视觉的定位可以通过以下简化的管道实现:(1)通过双目立体摄像头，首先得到一个视差图，该视差图可用于提取每个点的深度信息;(2)通过匹配连续立体图像帧间的显著特征，建立不同帧间特征点之间的相关性。然后我们可以估计过去两帧之间的运动;(3)通过对比已知地图的显著特征，我们还可以得出当前车辆的位置。然而，这种基于视觉的定位方法对光照条件非常敏感，因此，这种方法本身并不可靠。

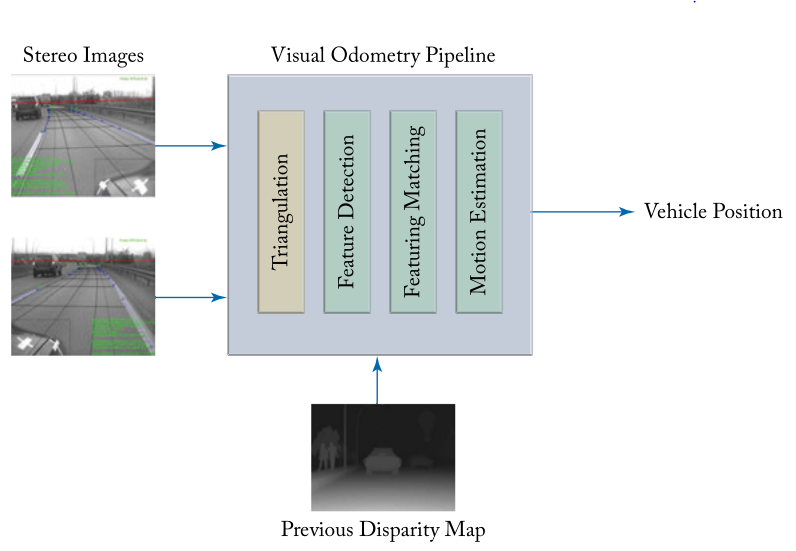


图1.3 立体视觉测距

这就是为什么激光雷达方法通常求助于粒子滤波技术。激光雷达生成的点云提供了对环境的“形状描述”，但很难区分单独的点。通过使用粒子滤波器，系统将特定的观察形状与已知的映射进行比较，以减少不确定性。为了定位与这些地图相关的移动车辆，我们可以应用粒子滤波方法将激光雷达的测量数据与地图关联起来。粒子滤波方法实现了10厘米精度的实时定位，在城市环境中是有效的。然而，激光雷达有它自己的问题:当空气中有许多悬浮粒子时，如雨滴和灰尘，测量结果可能异常糟糕。因此，如图1.4所示，为了实现可靠和准确的定位，我们需要一个传感融合处理来整合所有传感器的优点。

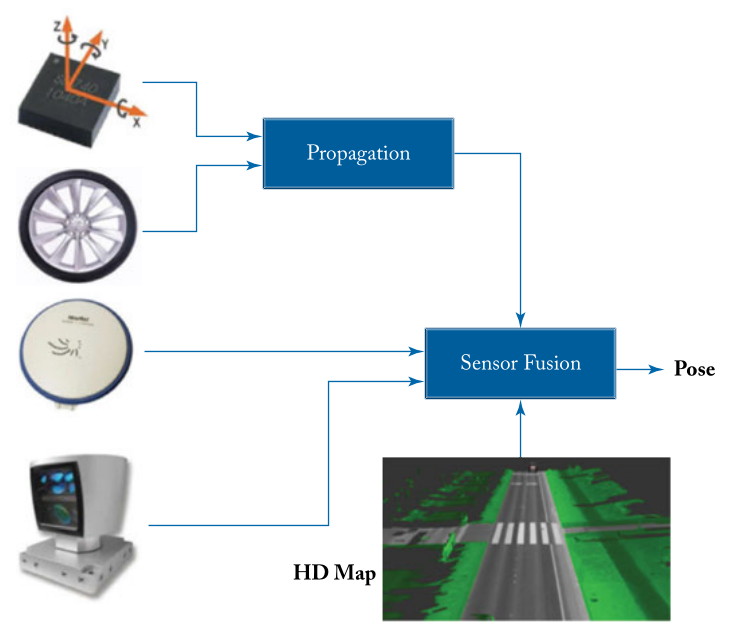


图1.4 传感融合定位管线

### 1.2.3 目标识别和跟踪

由于激光雷达提供了非常准确的深度信息，因此它最初主要用于自动车辆的目标检测和跟踪任务。然而，近年来，我们看到深度学习技术的快速发展，实现了显著的目标检测和跟踪精度。卷积神经网络(Convolution Neural Network, CNN)是一种深度神经网络，广泛应用于物体识别任务中。一个通用的CNN评估管道通常由以下几层组成。(1)卷积层包含不同的滤波器提取与输入图像不同的特征。每个过滤器包含一组“可学习的”参数，这些参数将在训练阶段之后得到。(2)激活层决定是否激活目标神经元。(3)池化层减少了表示的空间大小，从而减少了参数的数量，从而减少了网络中的计算量。(4)全连接层，神经元与前一层的所有激活都有完全连接。

目标跟踪是指对目标运动轨迹的自动估计。利用目标识别技术对目标跟踪对象进行识别后，目标跟踪的目标是对目标的轨迹进行后续的自动跟踪。这项技术可以用来跟踪附近移动的车辆以及过路的人，以确保当前的车辆不会与这些移动的物体相撞。近年来，与传统的计算机视觉技术相比，深度学习技术在目标跟踪方面具有明显的优势。通过使用辅助的自然图像，可以离线训练一个堆叠的自动编码器来学习一般的图像特征，这些特征对于视点和车辆位置的变化具有更强的鲁棒性。然后利用离线训练的模型进行在线跟踪。

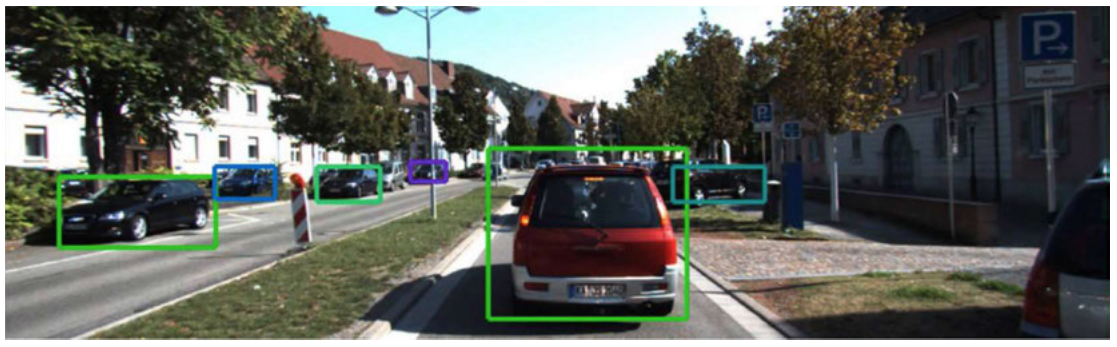


图1.5 目标识别和跟踪

### 1.2.4 执行

基于对车辆环境的了解，决策阶段可以实时生成安全高效的行动规划。

行动的预测

人类驾驶员在交通中面临的主要挑战之一是如何应对其他驾驶员可能采取的直接影响其自身驾驶策略的行为。当道路上有多条车道或车辆处于交通转换点时尤其如此。为了确保车辆在这些环境中安全行驶，决策单元生成对附近车辆的预测，并根据这些预测确定行动规划。为了预测其他车辆的行动，可以生成其他交通参与者的可达位置集的随机模型，并将这些可达集与概率分布联系起来。

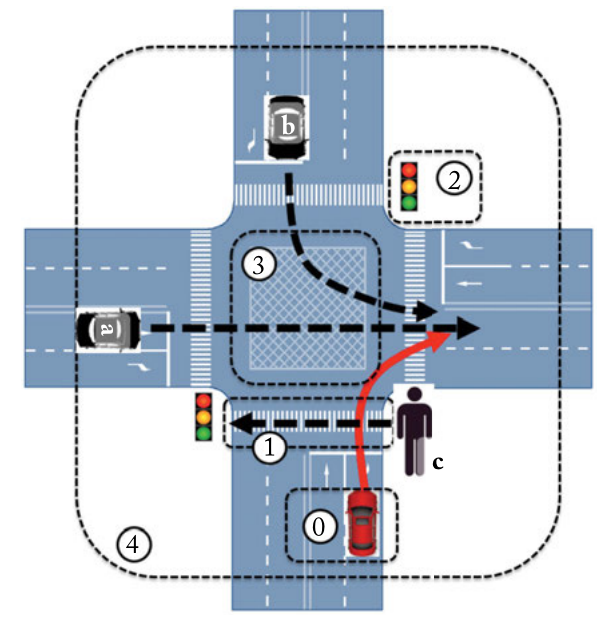


图1.6 行动预测

路径规划

在动态环境中规划自主敏捷车辆的路径是一个非常复杂的问题，特别是当车辆需要充分利用其机动能力时。蛮力方法是搜索所有可能的路径，并利用成本函数来确定最佳路径。然而，蛮力方法将需要大量的计算资源，并且可能无法实时交付导航计划。为了克服确定性完整算法的计算复杂性，利用概率规划器提供了有效的实时路径规划。

避障

由于安全是自动驾驶最重要的考虑因素，我们通常会采用至少两层的避障机制，以确保车辆不会与障碍物相撞。第一层是主动的，基于流量预测。在运行时，交通预测机制产生诸如碰撞时间或预测最小距离等措施，并基于此信息，触发避障机制，进行局部路径重新规划。如果主动机制失败，第二层，即使用雷达数据的反应机制将接管。一旦雷达探测到前方有障碍物，它就会越过当前的控制来避开障碍物。

## 1.3自动驾驶客户端系统

客户端系统将上述算法集成在一起，以满足实时性和可靠性要求。其中的一些挑战如下:系统需要确保处理管道足够快，以处理生成的大量传感器数据;如果系统的一部分发生故障，它必须足够健壮从故障中恢复;此外，它还需要在严格的能源和资源约束下执行所有的计算。

1.3.1机器人操作系统（ROS）

ROS是一个为机器人应用量身定制的强大的分布式计算框架，并得到了广泛的应用。如图1.7所示，每个机器人任务(例如定位)都驻留在一个ROS节点中。ROS节点可以通过主题和服务相互通信。它是一个非常棒的自动驾驶操作系统，除了它由如下几个问题：(1)可靠性:ROS只有一个主系统，没有监视器来恢复故障节点;(2)性能:发送广播消息时，消息重复多次，导致性能下降;(3)安全性:没有认证和加密机制。虽然ROS 2.0承诺会修复这些问题，但ROS 2.0本身还没有经过广泛的测试，而且许多特性还不可用。因此，为了在自动驾驶中使用ROS，我们需要首先解决这些问题。

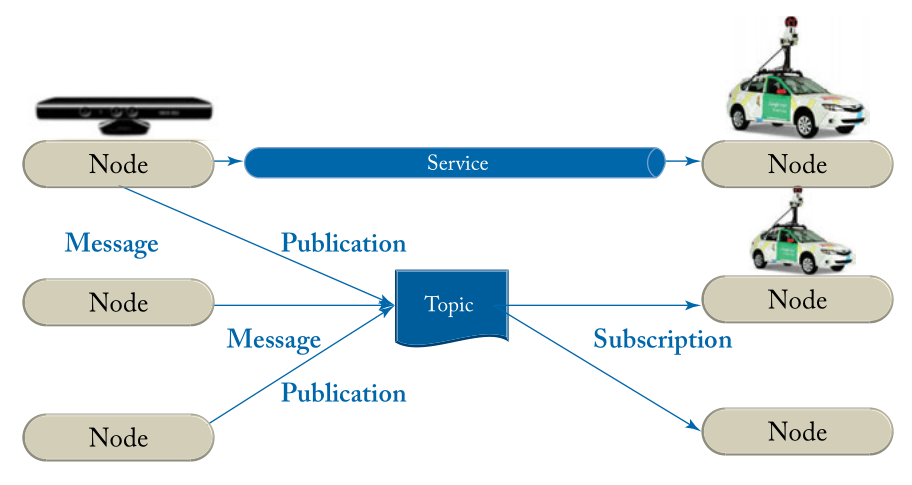


图1.7 机器人操作系统（ROS）

可靠性

当前ROS实现只有一个主节点，当主节点崩溃时，整个系统将崩溃。这并不符合自动驾驶的安全要求。为了解决这个问题，我们在ROS中实现了一个类似zookeeper的机制。如图1.8所示，我们有一个主节点和一个备份主节点。在主节点故障的情况下，备份节点将接管，确保系统仍然运行没有中断。此外，这个ZooKeeper机制监视并重新启动任何故障节点，确保整个ROS系统是可靠的。

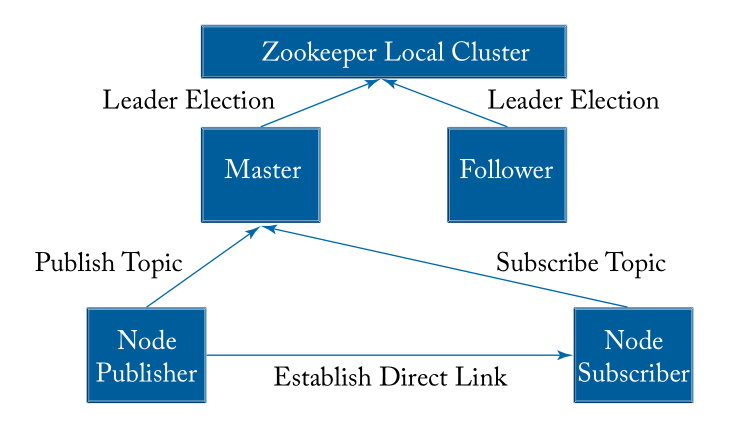


图1.8 ROS的ZooKeeper

性能

性能是当前ROS实现的另一个问题。ROS节点之间经常通信，保证节点间的有效通信至关重要。首先，当本地节点彼此通信时，我通过环回机制。每次通过环回管道时，都会引入一个20ms的开销。为了消除这种开销，对于本地节点通信开销，我们使用了共享内存机制，使得消息不必通过TCPIP堆栈才能到达目标节点。其次，当ROS节点广播消息时，消息会被多次复制，这会消耗大量的系统带宽。如图1.9所示，通过切换到多播机制，大大提高了系统的吞吐量。

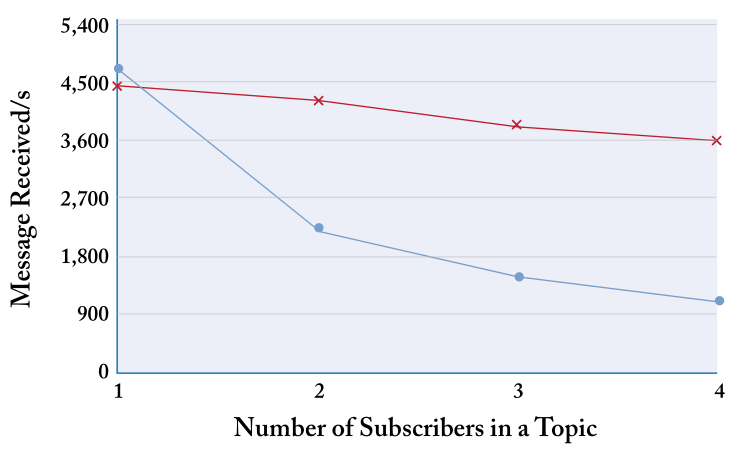


图1.9 ROS中重复传播与广播对比

安全

我们知道，安全是ROS最关心的问题。现在设想两个场景:在第一个场景中，一个ROS节点被绑架并持续分配内存，直到系统耗尽内存并开始杀死其他ROS节点。在这个场景中，黑客成功地使系统崩溃。在第二种情况下，由于默认情况下ROS消息是不加密的，黑客可以很容易地窃听节点之间的消息并应用中间人攻击。要解决第一个问题，我们可以使用Linux Container (LXC)来限制每个节点使用的资源数量，并提供一种沙箱机制来保护节点不受其他节点的影响，从而有效地防止资源泄漏。为了解决第二个问题，我们可以对通信中的消息进行加密，防止事件消息被窃听。

1.3.2 硬件平台

为了理解自主驾驶硬件平台设计的挑战，让我们来看看一家领先的自主驾驶公司的计算平台实现。它由两个计算盒组成，每个计算盒配备一个Intel Xeon E5处理器和4到8个Nvidia K80 GPU加速器。第二个计算盒执行完全相同的任务，用于可靠性:如果第一个计算盒失败，第二个计算盒可以立即接管。在最坏的情况下，当两个盒子运行在他们的峰值，这将意味着超过5000瓦的电力消耗，从而产生大量的热量。此外，每个盒子的价格在2万到3万美元之间，使得普通消费者无法负担整个解决方案。

该设计的功率、散热和成本要求阻止了自动驾驶汽车进入大众市场。为了探索包络的边缘并了解自主驾驶系统在ARM移动SoC上的表现如何，我们在基于ARM的移动SoC上实现了一个简化的、基于视觉的自主驾驶系统，峰值功率消耗为15w。结果表明，性能接近我们的要求:本地化管道能够每秒处理25张图像，几乎与每秒生成30张图像的速度相当。深度融合管道每秒可以执行2-3个目标识别任务。规划和控制管道可以在6毫秒内规划路径。有了这个系统，我们能够以每小时5英里的速度驾驶汽车，而不丢失任何定位功能。

1.4 自动驾驶云平台

自动驾驶汽车是移动系统，因此它们需要一个云平台来提供支持。云平台提供的两个主要功能包括分布式计算和分布式存储。它有几个应用，包括用于验证新算法的仿真;高清地图生产;和深度模型的训练。为了构建这样一个平台，我们使用Spark进行分布式计算，使用OpenCL进行异构计算，使用Alluxio进行内存存储。通过集成Spark、OpenCL和Alluxio，我们成功地交付了一个可靠的、低延迟的、高吞吐量的自主驱动云平台。

1.4.1 仿真

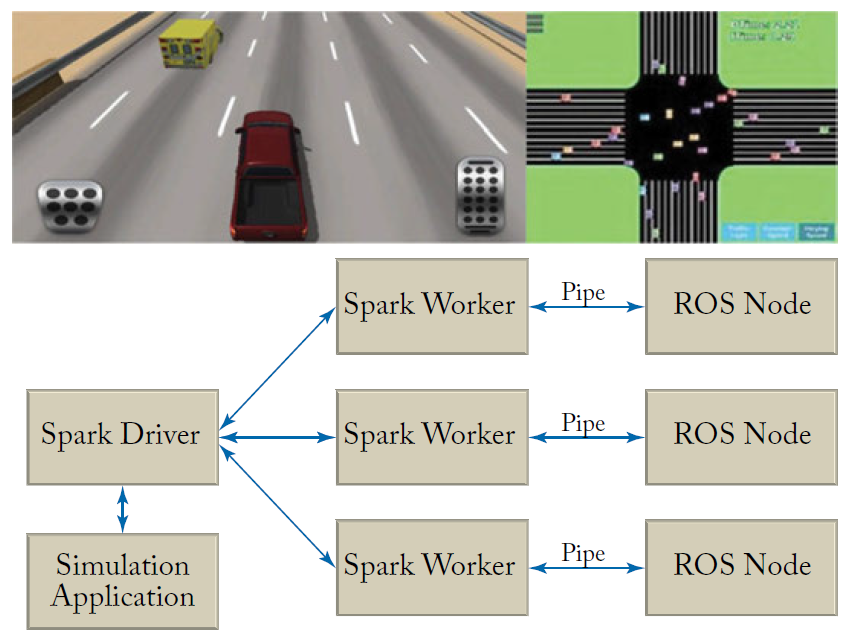


图1.10 基于Spark和ROS的仿真平台

这种系统的第一个应用是仿真。当我们开发一种新的算法时，我们需要在将其应用到汽车上之前对其进行彻底的测试。如果我们要在真正的汽车上测试它，成本将是巨大的，测试周期将很长。因此，我们通常在仿真器上进行测试，例如通过ROS节点回放数据。然而，如果我们要在一台机器上测试新算法，要么它将花费太长时间，要么我们没有足够的测试覆盖率。如图1.10所示，为了解决这个问题，我们开发了一个分布式仿真平台。因此，我们使用Spark来管理分布式计算节点，并在每个节点上运行一个ROS重放实例。在我们使用的自主驾驶目标识别框架，若在一台服务器上运行需要3个小时;使用我们开发的分布式系统，当我们扩展到8台机器时，测试在25分钟内完成。

1.4.2 高精度地图

如图1.11所示，高精度地图制作是一个复杂的过程，包括原始数据处理、点云制作、点云对齐、2D反射地图生成、高精度地图标注以及最终的地图生成等多个阶段。使用Spark，我们把所有这些阶段连接到一个Spark任务中。更重要的是，Spark提供了一种内存处理机制，使得我们不必将中间数据存储在硬盘上，从而大大降低了地图生成过程的性能。

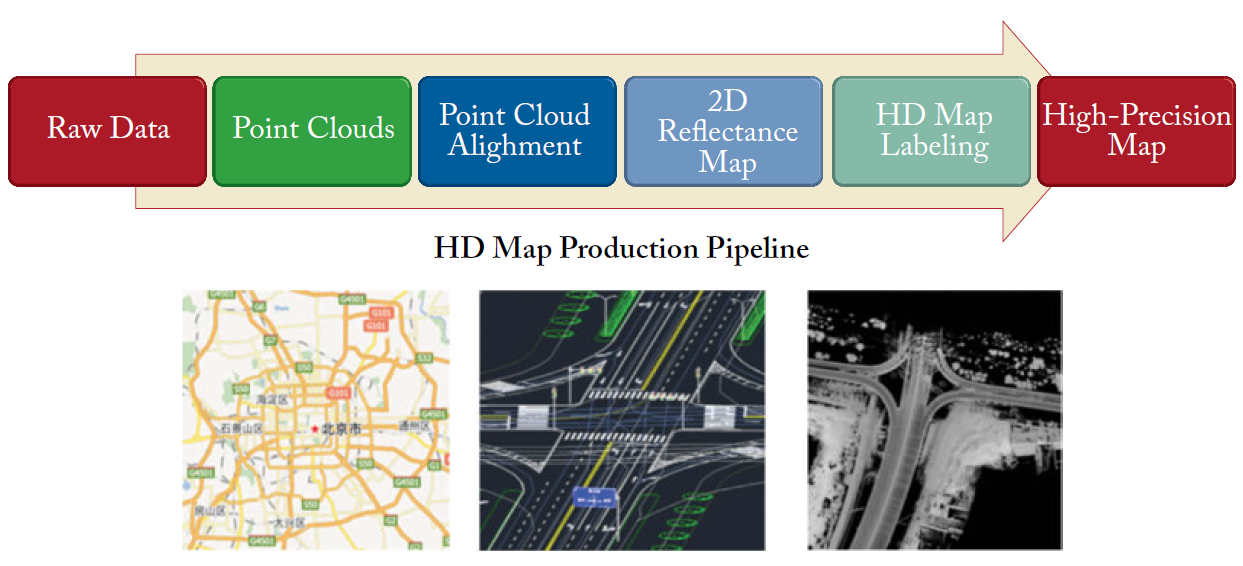


图1.11 基于点云的高精度地图制作过程

1.4.3 深度学习模型训练

当我们在自主驾驶中使用不同的深度学习模型时，必须提供最新的数据来不断提高这些模型的有效性和效率。但是，由于生成的原始数据量非常大，我们无法使用单个服务器实现快速的模型训练。为了解决这个问题，我们开发了一个高度可扩展的分布式深度学习系统，它使用了Spark和Paddle(一个最近由百度开源的深度学习平台)。如图1.12所示，在Spark驱动程序中，我们管理一个Spark上下文和一个Paddler上下文，在每个节点中，Spark executor承载一个Paddler trainer实例。此外，我们使用Alluxio作为这个系统的参数服务器。通过使用这个系统，当我们添加更多的资源时，我们已经实现了线性的性能扩展，这证明了这个系统是高度可伸缩的。

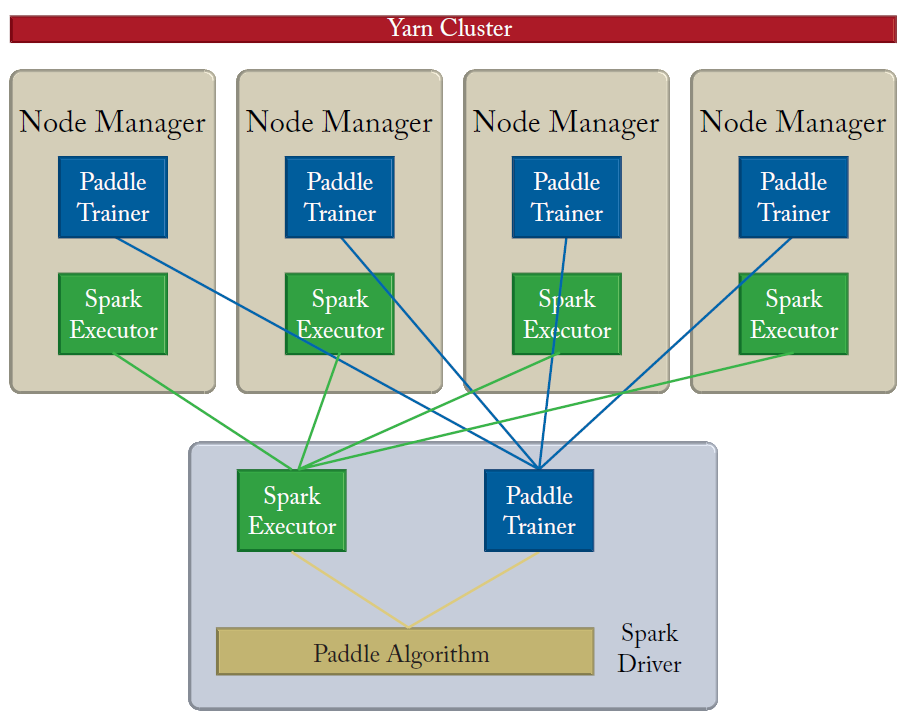


图1.12 分布式深度学习模型训练系统

1.5 这仅仅是个开始

再次声明，自动驾驶，或者一般的人工智能，不是一项单一的技术，它是多种技术的集成。它需要在算法、系统集成和云平台方面进行创新。这只是一个开始，在这个时代有大量的机会，我预计在2020年，我们将正式开始AI时代，并开始看到越来越多的基于AI的产品进入市场。

书的其余部分组织如下。定位是自动驾驶中最重要的任务之一，将在第二章介绍。至于检测，即第三章描述了基于感知数据的环境“理解”，并探讨了不同算法的应用，包括场景理解、图像流、跟踪等。大数据集、图像分类、目标检测、语义分段等所需要的高度复杂的计算，最好由第四章章所提倡的感知深度学习方法来处理。 一旦自动驾驶汽车理解了周围的环境，它就必须以某种方式预测未来的事件(例如，在它附近的另一辆汽车的运动)并规划自己的路线。这就是第五章的目的。第六章包括了更详细的决策、计划和控制。第7章是基于强化学习的规划和控制的设计，在自主系统开发中完全集成情景场景。在这一切的背后，车载计算平台是第八章的主题. 最后，第9章介绍了用于“将所有内容连接在一起”的云平台的基础结构。