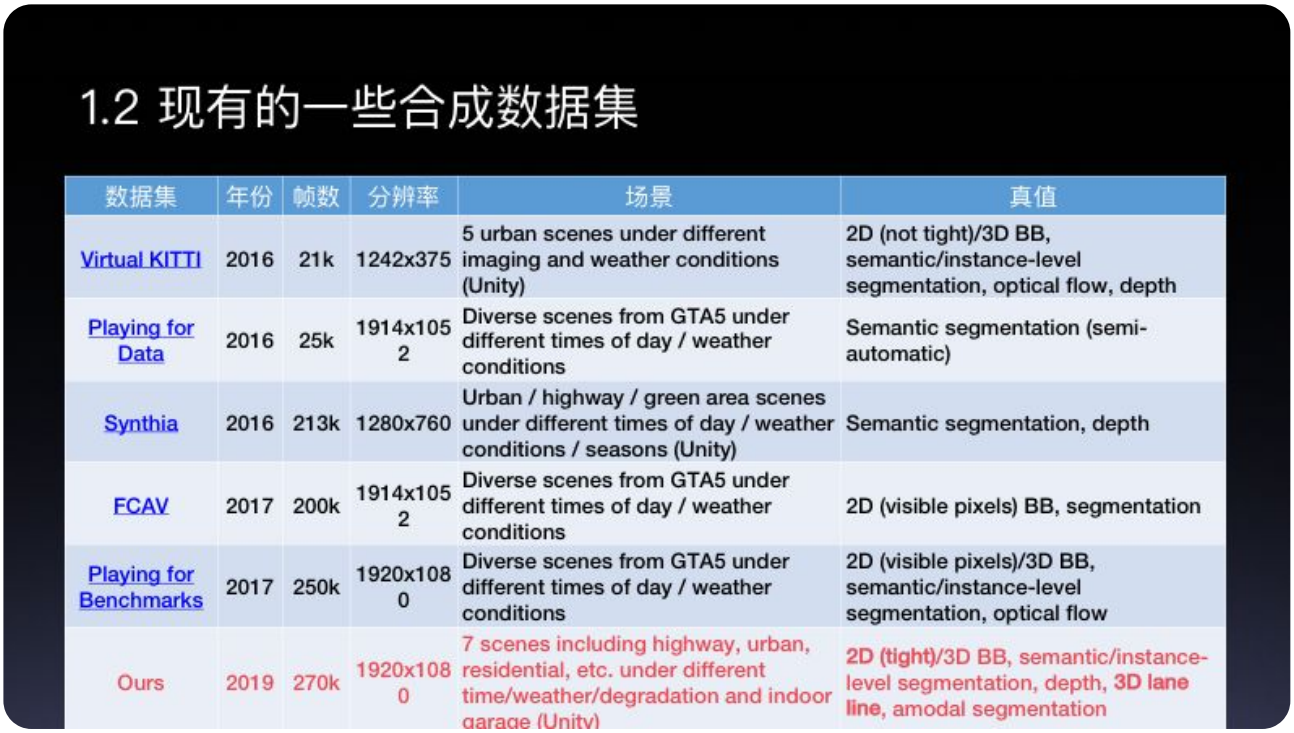
Apollo公开课丨为纯视觉感知生成合成数据集

视觉感知是指使用相机而不使用LiDAR传感器的感知系统，是所谓的低成本解决方案。今天要分享的内容是为纯视觉感知生成合成数据集，具体介绍了为什么需要合成数据集，怎么制作合成数据集，以及如何制作高质量的虚拟世界和如何产生精确的真值。

合成数据集的一个优势是**容易快速生成精确的真值，**不再需要耗时、耗力、耗钱，以及容易出错的人工标注。我们在真实世界中采集的所有原始数据是不能直接训练的，必须要经过人工标注出图片里面的障碍物、车道线等等，这一过程需要耗时、耗力。另外如果只用相机采集真实数据集，有些三维真值没有其他传感器（比如LiDAR）的帮助，是无法得到的。而用合成数据集以游戏引擎制造整个虚拟世界，可以获得所有真实信息。合成数据集的另外一个优势是**可以动态生成多样性数据集，其中包括不同的时间、不同的天气、不同的路况、不同的车辆颜色以及随机产生的路障等。**

下表是一些现有合成数据集的对比。合成数据集大概有五个，这第一个和第三个是**基于游戏引擎合成的，**其它三个是**用《侠盗猎车手》游戏合成的，**里面的场景是游戏公司花了好几年的时间和上亿美元打造的逼真仿真环境。通过一些手段再现游戏场景，根据里面的真实数据生成一些合成数据。表格最下面是Apollo的合成数据集，其分辨率是**1920X1080**，有超过**27万**张图片，这些图片不是视频连续图片。我们目前打造了七个场景，包括高速公路、城市、居民区还有室内停车场等。



▲合成数据集的对比

另外，我们现在提供的真值种类很多，有二维的包围盒、三维的包围盒。我们的合成数据集是第一个也是唯一一个提供三维车道线真值的。

制作合成数据集的技术难点主要有两条，一是**如何高效的制作新场景、新世界，**是目前最大的难点之一，我们目前只有七个场景，并不是有几十个场景。另外一个技术难点是**真实数据和合成数据之间的Domain Gap问题，**解决这个难点可以通过提高渲染质量，采用**Domain adaptation、Domain randomization、Image-to-image translation的方法。**

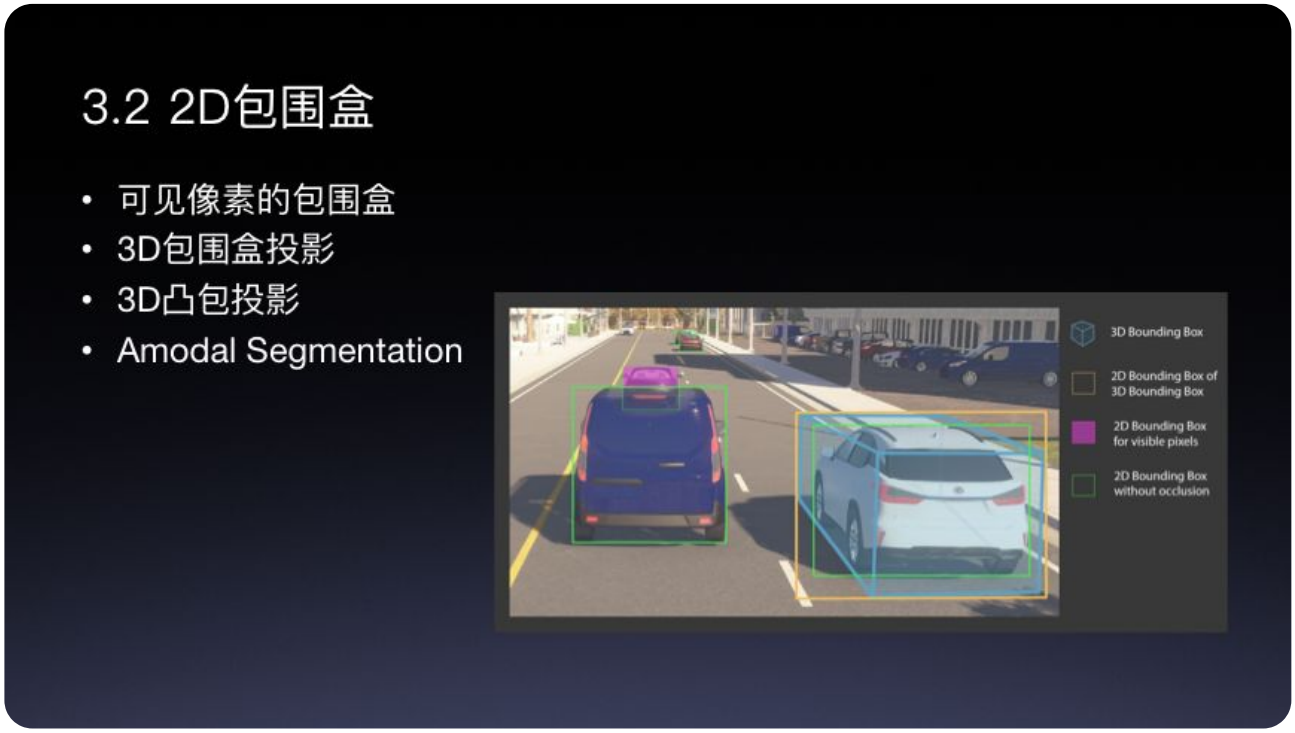
目前，我们采用**虚拟测量法**，通过真实相机拍摄大量的图片，然后对图片重叠部分进行对准，产生点云，生成网格，最后贴上纹理，得到一个比较真实的三维模型。一般来说，得到的三维模型是特别密的，而且是跑不起来的，因此需要大大简化。通过使用向量贴图、高度贴图等纹理，使简化后的网格看起来还是真实的，保留所有的真实性。下面这个视频我们可以看到左边是我们合成的场景，右边是我们实际拍摄的场景，可以看到我们合成的场景达到以假乱真的效果。

除了拍摄之外，我们还参考了大量文档，因为所有的文档一般包括一些政府或者某些行业机构制定的标准文档，比如说常见的八角形的每个边要多宽，底色用什么颜色，用什么字号，路灯的高度，车道

线的宽度以及所有的细节都有相关的文档。此外，我们的美术也参考了大量真实文档，保证复制出来的场景跟真实场景尽可能的一致。

如何产生精确的真值

我们可以产生很多种不同的数据真值，包括**二维、三维包围盒，还有截断、遮挡，**此外，还可以产生深度图像的真值，**像素级语义分割和物体分割，**以及我们所知唯一一个三维车道线真值。



▲ 2D 包围盒

物体检测真值

二维包围盒是最基本也是感知团队最需要的真值。二维包围盒有很多方式，最容易的方式是可见像素的包围盒，渲染出一个物体有多少像素，对所有这些像素取最小值或最大值就得出一个二维包围盒，**图中粉红色区域就是二维包围盒，**这个物体被遮挡了一部分。如果仅仅得到二维包围盒是不够的，训练感知算法需要整辆车的包围盒，而且包围盒是绿色的部分，假设这个车没有被遮挡，它的包围盒应该是什么样。

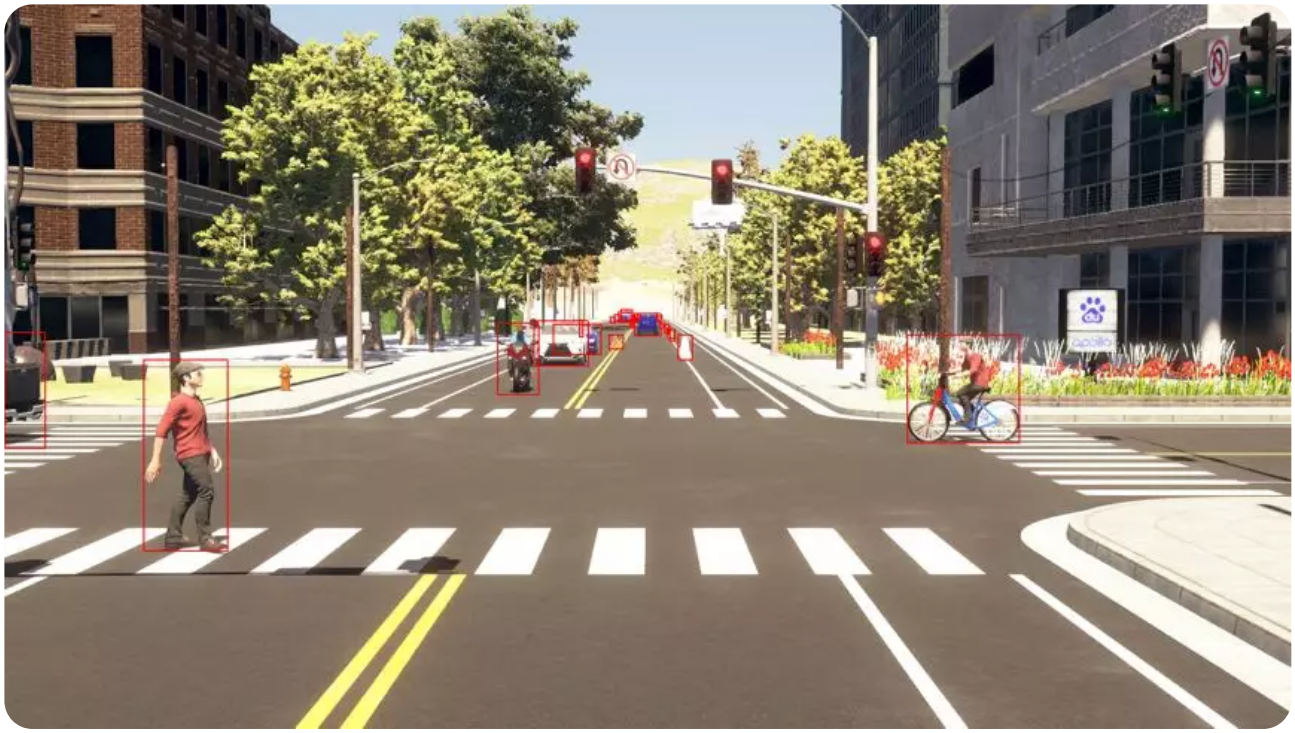
三维包围盒的投影。所有物体的每一个点的三维坐标都知道，每一个点取最小值或最大值很容易得到一个三维包围盒，把它投影到平面上再计算投影的包围盒得到二维包围盒，如图右边所示。**蓝色部分**

就是三维包围盒，投影得到的二维包围盒是黄色的包围盒，它比车要大一些。而感知团队希望得到绿色紧密的包围盒。那么，怎么得到这样的包围盒，有**三维凸包投影和Amodal Segmentation，**这两个方法都可以得到精确的二维包围盒，各有优缺点。

▲ 3D 凸包投影的 2D 凸包

三维凸包的好处是物体有一部分在图像之外也可以得到它的投影。用三维凸包的二维包围盒与图像求一个交，就可以知道物体有多大的比例被截断，这个真值也是非常重要的一个真值。然后利用三维凸包分别计算X/Y方向被截断的比例，进而得到被截断部分的投影包围盒。

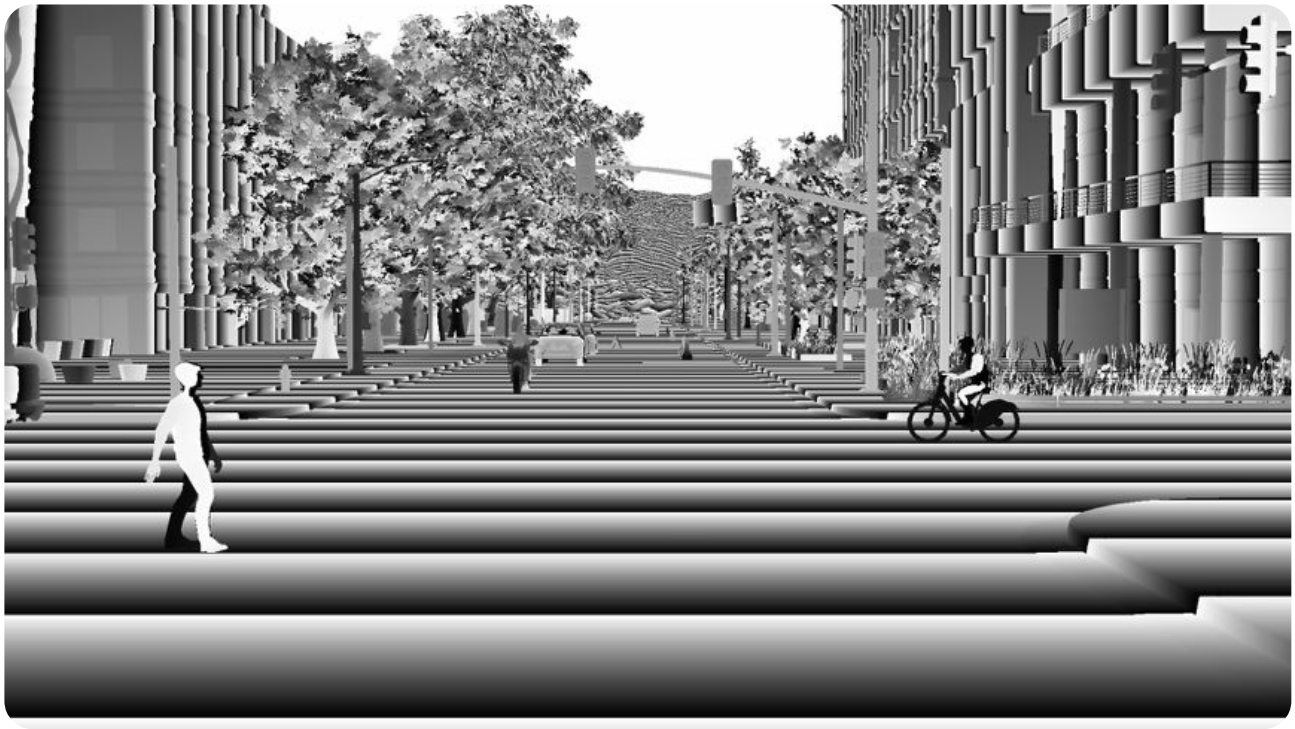
另外一个就是对于被遮挡的物体，人工标注是不可能看到这个被遮挡物体有多少像素，只是大概估计一下物体被遮挡多少，不会产生一个精确的数值，而是产生四个离散值：**完全可见、大部分可见、少部分可见、完全不可见。**而使用**Amodal Segmentation**（每个物体独立渲染）可以产生得到精确的遮挡比例，可以进一步提供感知算法精度。我们也可以一次性渲染多个物体，然后用位操作和MRT对Amodal Segmentation进行加速。例如，如果有32位的MRT可以同时渲染32个物体，这样可以大大的加速。下面就是一些实际的例子，这个例子有一些人、车、障碍物的场景，可以看到蓝色的车的大部分都被前面白色的车遮挡了，如果用可见像素做成包围盒只有一点点。但是用Amodal Segmentation分别渲染两个车，可以得到精确的包围盒。



▲ Amodal Segmentation 分别渲染两个车的实例

深度图像

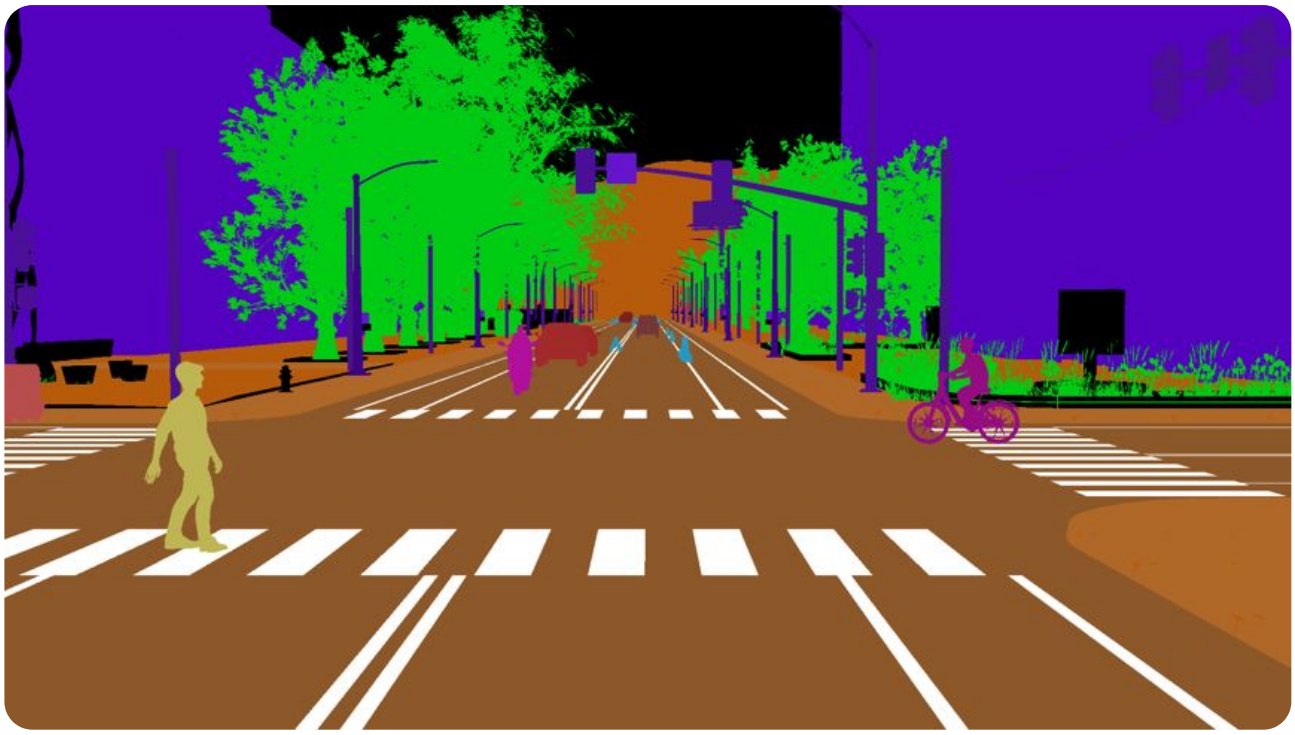
因为我们有整个三维场景的信息，可以把三维场景中每一个像素的深度都存储下来，得到深度图像，可以给感知算法提供另外一种真值。目前我们存储的深度是像素坐标中的Z值，而不是到相机中心的距离。假设最远是655.35米，16位存储可以达到一厘米的精度，我们用**R、G两个通道**，每个通道**八位**来存储，一个存储高八位一个低八位。



▲深度图像

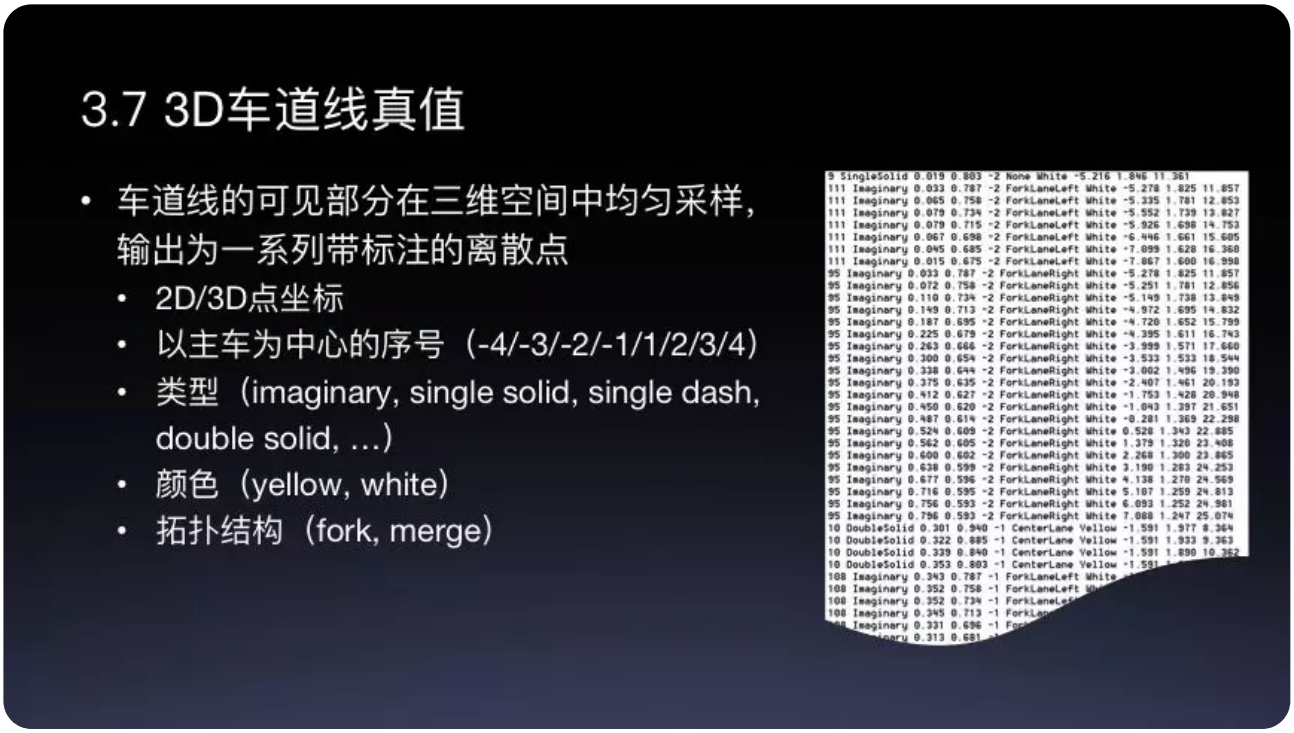
图像分割

我们可以提供像素级的图像分割，同时提供**语义分割**，所有的车是一个颜色，所有的人是一个颜色，路面是一个颜色，建筑物是一个颜色。此外，我们还可以提供**物体分割**，每一辆车都是不同的颜色，把这两种是存储在同一幅图像中，可以节约存储空间，叫做**全光分割**。因为自动驾驶对车和行人较为关注，我们对车和行人做了物体分割，对于其它的路面和建筑物只做了语义分割，三维场景中的语义包括**Sedan、Coupe、SUV、Hatchback、Van、PickupTruck、Truck、Bus、Cyclist、Motorcyclist、Pedestrian、TrafficCone、Barricade、Road、LaneMarking、TrafficSign、TrafficLight、Sidewalk、GuardRail、Sky、Terrain、Pole、StreetLight、Building、Vegetation** 等二十几类。为了方便用户，我们让每一类内部的物体共享色相值（HSV中的H），使得同一类的物体颜色彼此类似而又不同。



▲图像分割

三维车道线



▲ 3D 车道线真值

我们是唯一一个提供三维车道线的数据库。对车道线的可见部分在三维空间进行均匀采样，输出一系列带标注的离散点。目前，**每20厘米采样一个点，得到一堆的三维离散点，**这些点的真值保存在一个文件中，如图右边所示，每一行记录了二维坐标点和三维坐标点，还有一个序号，**序号表示相对主车所在的位置。**在车主左边的车道线以此是负1、负2、负3、负4，右边的是正1、正2、正3、正4，以及车道的类型和颜色，如实线还是虚线，黄线还是白线，车道线是分叉还是交汇，都是我们提供车道线的真值。

▲提供车道线的真值

以上就是本次关于为纯视觉感知生成合成数据集课程的全部内容。欢迎大家提出问题，进入社群进行交流。更多相关技术干货也可以继续关注后续的课程。