浙江大学

报告题目：	小学期论文阅读报告
学　　院：	计算机科学与技术学院
专业与班级：	计科 1902
姓　　名：	
学　　号：	

小学期论文阅读报告

1. Neural Fields in Visual Computing and Beyond

1.1. 导言

这是一篇综述，阅读此综述源于我在阅读 NeRF 时发现有许多自己不清楚的概念。为了全面了解毕业设计要研究的这一方向，我找了两篇综述对 NeRF 和 NeRV 的相关知识进行学习。这篇综述发表于 2022 年，许多最新的研究成果都有被谈到。

在综述的开篇，作者首先明确了神经场（neural field）的定义。首先看场（field），场在物理学上是为所有（连续）空间和/或时间坐标定义的量，如电磁场，通过一个时空坐标得到一个物理量。场被广泛地用于表示一个物体或者一个场景的潜在的物理量，进而可以从这些物理量重建物体或场景，而神经场则是指一个部分或全部被神经网络参数化的场。

用来参数化场的神经网络通常选用 MLP（Multi-Layer Perceptron）。选用 MLP 的理由之一是场可以表示为一个方程，而 MLP 是一个通用逼近器（universal approximator）。MLP 可以在学习的过程中逼近真实的场的方程，即使没有对方程的形式做出任何预设。选用 MLP 的另外一个理由是，传统的表示物体或场景的方法大多基于采样，而采样受到奈奎斯特采样定理、离散性、存储空间等诸多限制，用神经场来表示物体或场景可以克服以上问题。

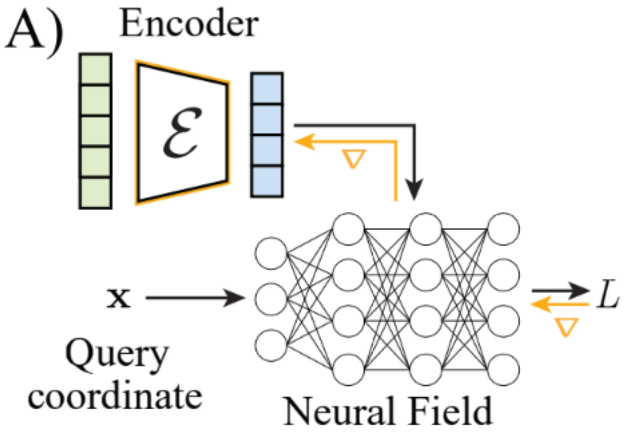
接下来，作者主要介绍了五种方法，这五种方法分别是先验学习与调节（prior learning and conditioning）、混合表示（hybrid representation）、前向图（forward map）、网络架构（network architecture）和控制神经场（manipulate neural field）。这五种方法分别从不同的环节和层面提升神经场的能力。最后作者介绍了神经场在各个应用领域的成果，对神经场未来的发展做出展望。

1.2. 先验学习与调节

先验（prior）反映了模型在观察数据之前所具有的知识，这些知识可以来源于训练集的数据也可以来源于人的主观认知。而调节（conditioning）则是希望先验可以被广泛地应用于不同的条件（varying conditions）。在神经场上实现调节常用的方法是利用隐变量z（latent variables）：隐变量包含了特定场的属性，用隐变量来控制神经场。调节可以按照隐变量的生成分为自动编码器（auto encoder）和自动解码器（auto decoder），也可以按照隐变量的作用范围分为全局调节（global conditioning）和局部调节（local conditioning）。调节可以影响神经场的泛化能力。

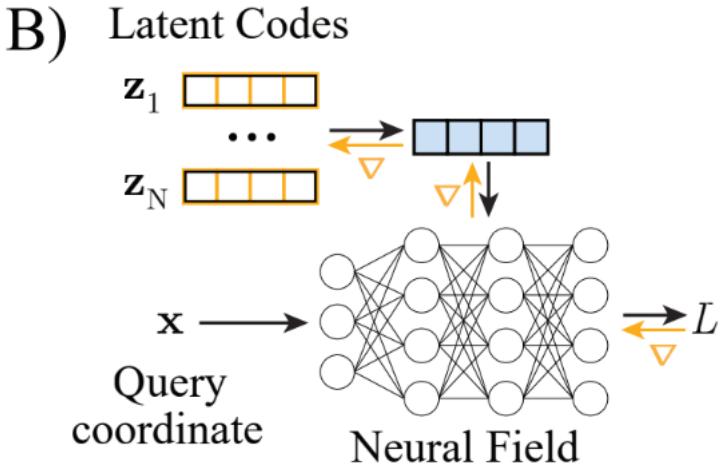
1.2.1. 自动编码器

隐变量由一个常常是神经网络的编码器生成，编码器的输入是一个观察（observation，可以是一个物体或者场景），输出是该观察对应的隐变量。在这一结构中，解码器则是被隐变量控制的神经场。训练时编码器的参数会和神经场的参数一起被优化。这种结构速度快，因为编码和解码都只需要一次前向传递（forward pass）。



1.2.2. 自动解码器

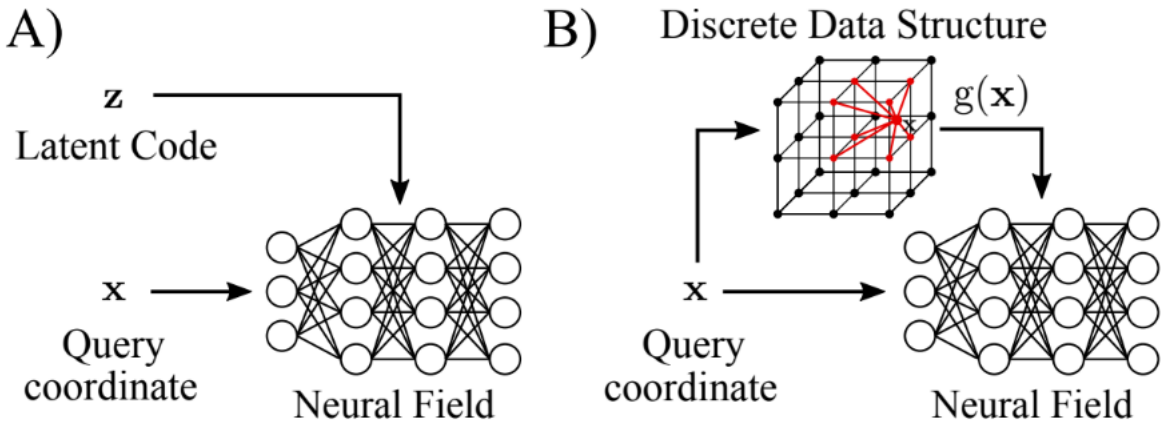
在这一结构中，由优化过程充当编码器，每个观察对应的隐变量被映射为神经场的参数，并通过最小化损失函数得到，而解码器是被隐变量控制的神经场。训练时隐变量和神经场的参数一起被优化。将自动解码器应用于一个新的观察时，神经场的参数被固定，而新观察的隐变量需要通过一个简短的训练得到，这使得自动解码器的速度要显著慢于自动编码器。但是相比于自动编码器，它的优点是它没有对观察作出附加的假设，没有引入多余的参数（编码器神经网络的参数）。比如在自动编码器中，一个处理 2D 图片输入的编码器常常意味着该结构已经假定了输入是一个代表 2D 图片的向量，而一个自动解码器可以处理任意尺寸或无序的输入。



1.2.3. 全局调节与局部调节

全局调节即一个观察只会产生一个隐变量，隐变量代表了整个观察的信息。神经场的参数也只被隐变量影响。

局部调节即一个观察会产生多个隐变量，每个隐变量只代表了观察的局部信息。因为产生了多个隐变量，需要一个离散数据结构g来存储隐变量。使用时，需要先利用时空坐标，从离散数据结构中取得该局部隐变量$z = g(\boldsymbol{x})$，以控制神经场。



1.2.4. 将隐变量映射为参数

隐变量通过映射为神经场参数来实现对神经场的控制。实现映射主要有三种方法：直接将时空坐标和隐变量连接起来作为神经场的输入、定义一个仿射方程将隐变量映射为神经场第一层的偏移向量以及训练一个神经网络进行映射。

1.3. 混合表示

混合表示将神经场与离散数据结构结合起来，通过将神经场的参数保存在离散数据结构中降低计算量，提高神经场的可扩展性。用离散数据结构直接保存参数的方案叫平铺（tiling），保存局部隐变量的方法叫嵌入（embedding）。在传统的方法中，离散数据结构也被用来表示物体或者场景；由于实际的物体或场景是连续的，各种插值方法常与离散数据结构一起使用。常用的离散数据结构可以分为规则网格（regular grids）和不规则网格（irregular grids）。

在数学上，一个离散数据结构可以表示成迪拉克 δ 函数的和：$g(\boldsymbol{x}) = \alpha_0\delta(\boldsymbol{x}_0 - \boldsymbol{x}) + \alpha_1\delta(\boldsymbol{x}_1 - \boldsymbol{x}) + \cdots + \alpha_n\delta(\boldsymbol{x}_n - \boldsymbol{x})$

1.3.1. 规则网格与不规则网格

常用的规则网格包括二维的像素（pixel）和三维的体素（voxel）。优点是可以通过简单的坐标进行索引，也可以运用标准的处理方法对数据进行处理。缺点是存储空间随着采样密度增大快速增大，同时因为奈奎斯特采样定理，采样高频信号时需要设置非常高的采样密度。

常用的不规则网格包括点云（point cloud）和网格（mesh）。优点是可以避免奈奎斯特采

样定理的限制，可以进行变形以自适应地增加复杂数据区域的容量。缺点是不规则的采样模式带来的一系列问题，比如索引问题和难以进行数据处理。

综述里解释点云时提到的维诺差值，和解释多边形网格时提到的调和坐标较为复杂，可以在学习更多数学知识后再次了解。

1.3.2. 常用的三维表示

阅读到这里的时候，我又去了解了常用的表示三维物体或三维场景的方法。这些方法可以分为显式表示和隐式表示。显式表示包括体素、点云和网格，它们把物体上的点的位置都表示出来；隐式表示包括 Occupancy Function 和 SDF（Signed Distance Function），它们不表达点的具体位置，而表示点与点的关系。

体素用规则的立方体表示三维物体，一个体素是数据在三维空间中的最小分割单位，类似于二维图像中的像素。这是一种规则的数据表示法，容易送入网络进行学习。

点云将物体表示为三维空间中点的集合，一般用激光雷达或深度相机扫描物体即可获得。这种数据的获取成本低，但是没有记录点之间的连接关系。

网格将多面体表示为顶点和面片集合，包含了物体表面的拓扑信息。渲染时常将多边形网格转化成三角形网格。网格是一种性价比较高的数据表示法，但是神经网络难以利用网格进行学习。

Occupancy Function 将物体表示为一个占有函数，输入点的坐标，函数输出该点是否在物体的表面上。SDF 将物体表示为符号距离函数，即空间中每个点到表面的距离。对于物体表面上的点，SDF 输出 0；对于物体外的点，SDF 输出正数；对于物体内的点，SDF 输出负数。理论上 Occupancy Function 和 SDF 都具有无限精度且内存占用少，但是实际上网络很难拟合出一个连续的函数，一般会通过采样将函数保存为离散的形式；都需要后处理才能得到显式几何结构。

1.4. 前向图

作者在这一节首先定义了重建域（reconstruction domain）和传感域（sensor domain）。重建域即如何表示世界，包括体素、SDF、辐射场等表示方法。传感域即如何观察世界，包括视觉（摄像机）等感知世界的手段。前向图即为一个从重建域映射到传感域的模型。这一个概念我觉得不是很重要，因为前向图似乎只是作者在综述里为了区分神经场的各个环节所提出的一个概念，实际这个概念并不广泛流行。前向图在图形学和可视计算里最重要的形式就是渲染。

1.4.1. 渲染

在这篇综述的语境下，渲染即是一个把神经场转换为图像的前向图。

如果一个神经场代表了一个几何形状的表面，渲染时常常需要计算光线和表面的交点，这

其实就是一个求根的过程。最常用的方法是光线步进（ray marching）及其变种，包括球面追踪（sphere tracing）和分段追踪（segment tracing）。光线步进代表沿着光线按照一定的步长采样，直到光线与物体表面相交。球面追踪是一种与 SDF 结合、动态调整步长的方法，每次步进根据 SDF 的值来调整步长。分段追踪则是根据求解安全距离调整步长。求解出光线和表面的交点后，则可以通过 BSDF 等方法，求解卡吉雅渲染方程（Kajiya’s rendering function）来进行渲染。

这里的分段追踪的涉及数学原理较为复杂，可以在学习相关的数学知识之后再次了解。

1.4.2. 内嵌物理知识神经网络

另外一种比较特殊的前向图是内嵌物理知识神经网络（PINN，Physics Informed Neural Network）。传统的神经网络方法大多是纯数据驱动的，可能会导致一个不现实也就是不可信的结果。PINN 通常用于求解偏微分方程，可以与 SDF 结合得到更加真实的结果。

1.4.3. 恒等映射函数

在某些领域，传感域就是重建域。在这种情况下，只需要让神经场对训练数据过拟合即可，此时的前向图为一个恒等映射函数（identity mapping function）。

1.5. 网络架构

作者在这一节讨论的主要是神经网络本身会带来的问题，比如频谱偏差（spectral bias）和激活函数（activation function）的选择。

频谱偏差指的是神经网络本身倾向于拟合一个低频的函数，这会导致现实世界中高频信号的丢失。克服这一问题的主要方法是位置编码（positional encoding）和选取激活函数。位置编码指的是将时空坐标从低维映射为高维，这一映射常常利用三角函数，可以表示为$\gamma(\boldsymbol{x}) = [\gamma_1(\boldsymbol{x}), \gamma_2(\boldsymbol{x}), \ldots, \gamma_m(\boldsymbol{x})]$。其中采用$\gamma_{(2i)}(x) = \sin\left(2^{i-1}\pi x\right), \gamma_{(2i+1)}(x) = \cos\left(2^{i-1}\pi x\right)$的位置编码被称为具有傅里叶特征（Fourier feature）的位置编码。另一种克服频谱偏差的方法是选用周期非线性函数作为激活函数。可以证明，使用具有傅里叶特征的位置编码和使用周期非线性函数作为神经网络的首层具有同样的效果。

此外，激活函数的选取也会对神经网络的求导带来影响，比如求解偏微分方程对激活函数的导数性质提出了一定的要求。

1.6. 控制神经场

这一节讲述的方法主要是如何控制神经场，也就是针对神经场的可编辑性。相比于传统的用离散数据结构代表物体或场景而言，目前神经场的编辑工具非常少，难以编辑一个用神经场表示的物体或者场景。一般控制神经场主要有两种方式：通过转换坐标输入或者直接编辑神经

场的参数。

1.6.1. 转换坐标输入

第一种方法是通过显式几何进行空间变换。对于物体建模问题，结构上的先验可以通过包围盒或者粗糙的显式几何实现。对于关节式物体，运动链可以提供对几何形状的显式控制，将多个局部的神经场组合起来表现整个关节式物体。

第二种方法是通过神经场进行空间变换。使用神经场进行空间变换是一个非常欠缺监督和约束的问题，即很容易变换出一个不真实的结果。因此，通过神经场进行空间变换需要一系列基于物理直接的正则损失项，这其中包括平滑度（smoothness）、稀疏性（sparsity）、循环一致性（cycle-consistency loss）和图像空间信息（image-space loss）。

第三种方法是进行时间变换。通过对时间坐标进行变换实现这一效果，如加速减速进行变换$\Phi(a \cdot t)$。

1.6.2. 直接编辑参数

直接控制隐变量或者控制神经场的参数都可以实现对神经场的控制，主要有以下三种方法：

隐变量插值或隐变量交换，在隐变量空间中使用其他的隐变量可以影响神经场。

微调隐变量或者微调神经场参数，可以在训练时通过微调隐变量或参数来控制神经场的行为。

通过超网络（hypernetwork）编辑神经场参数，神经场的参数众多，人力编辑参数是不现实的，但是可以通过训练一个超网络根据条件编辑神经场的参数。

2. State of the Art on Neural Rendering

2.1. 引言

这篇综述发表于 2020 年 NeRF 发表的半个月前。作者给神经渲染（neural rendering）下的定义是允许对场景属性（scene property）进行控制的使用深度学习的图片或视频生成方法。对于图片来说，通过在重建（reconstruction）和渲染（rendering）使用深度神经网络，神经渲染可以学习到从捕获图片到新图片（novel image）的映射。再结合物理知识和统计学知识，神经渲染可以完成对场景属性的设置和修改。

从神经渲染的定义可以看出，这篇综述的侧重点在于内容创造（content creation），即如何通过深度神经网络生成新的图片或视频。

2.2. 理论基础

该节的内容主要是图形学的内容以及深度生成模型。原文介绍了图形学知识以及生成对抗网络（GAN，Generative Adversarial Networks），并且我又去了解了另外一种常用的生成数据的

模型——变分自编码器（VAE，Variational Auto Encoder）。两种生成数据的模型都引入了随机变量，我的理解是引入随机变量可以获得一个连续的分布，生成数据的能力更强，同时随机性的引入增强了模型的鲁棒性，使得模型生成的数据更加丰富。

2.2.1. 图形学理论基础

这部分理论知识与图形学课程所学基本一致。

经典的图形学方法使用基于物理的方法来合成图像。经典图形学方法逼近现实世界中图像形成的物理过程：光源散发出光子，光子与场景中的物体发生互动，最后被照相机记录。即光线传播（light transport）的过程。光线传播的过程一般由卡吉雅渲染方程描述：$L_o(\boldsymbol{p}, \omega_o, \lambda, t) = L_e(\boldsymbol{p}, \omega_o, \lambda, t) + L_r(\boldsymbol{p}, \omega_o, \lambda, t)$，其中$L_o$代表表面上出射的辐射度，$L_e$代表表面自身散发的辐射度，$L_r$代表表面反射的辐射度。

图形学需要一定的方式来表示一个场景，这包括场景中物体的表示、场景中表面的材料和场景中的光源。表示一个物体有显式的表示和隐式的表示，前者包括体素、网格等表示方法，后者包括 SDF、辐射场（radiance field）等方法。材料可以通过反射描述，而材料的反射特性可以通过 BRDF（bidirectional reflectance distribution function）和 BSSRDF（bidirectional subsurface scattering reflectance distribution function）表示，前者描述了光线在材料表面的反射特性，后者描述了光线在材料内部的次表面散射特性。场景中的环境光源通常用环境贴图（environment map）来表示，也可以用纹理贴图（texture map）或者方程来表示一个连续发光表面。

图形学中最常用的相机模型是针孔相机模型。针孔相机模型有焦距等内参和位姿等外参，相机的成像过程也就是把世界坐标系转化为相机坐标系的过程。

图形学上的渲染指的是把包括相机、光源等场景定义转换为一个模拟的相机的图像的过程。渲染中用到最多的方法是栅格化（rasterization）和光线追踪（raytracing）。将从场景定义到图像的过程反过来，由图像估计场景的各个参数的过程被称为逆渲染（inverse rendering）。神经渲染用到的方法就是用神经网络来估计场景的各个参数，而不是使用预定义的物理模型或数据结构。神经网络的引入取得了非常好的效果。

除了经典的渲染方法外，综述还简单介绍了基于图像的渲染（IBR，image-based rendering）。这种方法通过组合、扭曲已有的图像来生成新图像。主要用于静态物体的新视角合成（novel view synthesis）。

2.2.2. 对抗生成网络

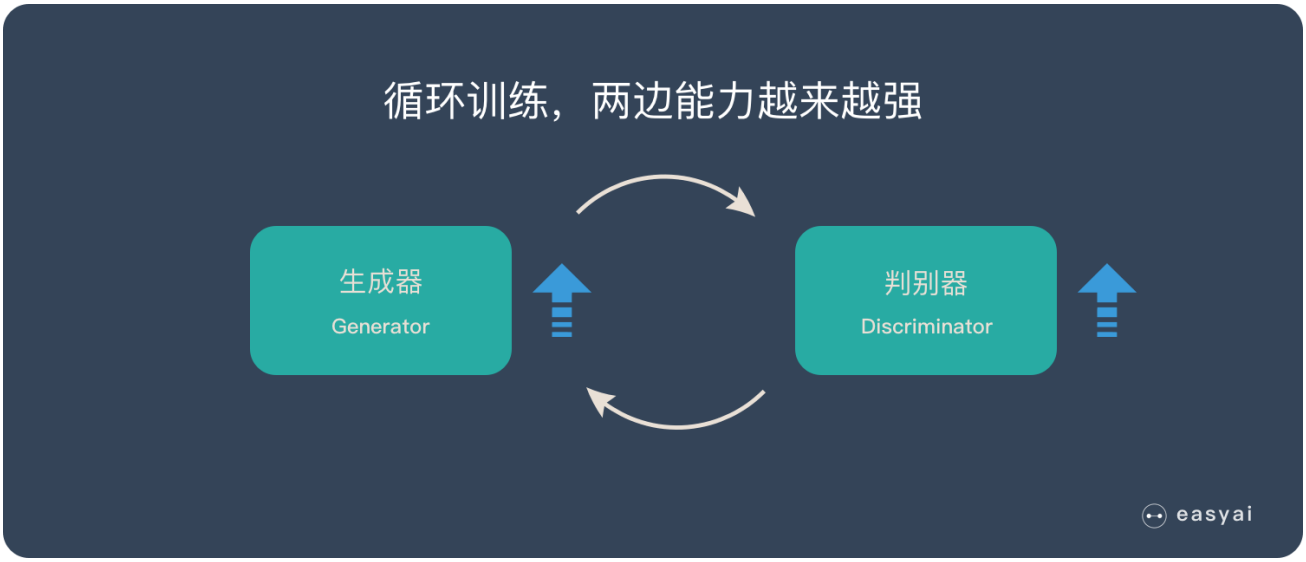
GAN 同时训练一个生成器（generator）和一个判别器（discriminator）。

生成器的输入一个服从某一随机分布（常常是高斯分布）的采样值，即噪声，输出一个图片。生成器的训练目标是欺骗判别器，即让生成器认为自己生成的图片是真实的。训练完成后，给予生成器一个输入即可生成图片。

判别器的输入为生成器的生成图片和来自训练集的真实图片，输出为图片来源（生成或真实）的概率。判别器的训练目标是正确区分出生成图片。训练的过程也就是在判别器的监督下，生成器学习到的分布逐渐向真实的分布逼近的过程。

GAN 使用的损失函数是$\mathcal{L} = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))]$。其中前半部分代表判别器将真实的图片判定为真实的概率，后半部分代表生成器将生成图片判定为生成的概率。这一损失函数的定义与二分类对数似然函数的交叉熵损失函数相似。判别器要最大化这一损失函数，而生成器则是希望最小化这一损失函数。

综述中特地提到为了实现对图像合成的控制，需要引入对模型的调节。其中的代表成果是 conditional GAN（cGAN）。这里的调节通过引入一个标签x实现，输入的随机变量采样值用z表示，生成器生成的图片用y表示。生成器的输入变成了标签和噪声输入，而判别器的输入变成了标签和生成的图片，判别器不仅判断图片的真假，还要判断图片和标签是否一致。这意味生成器不能仅仅生成一个图片，还要根据标签生成图片才能骗过判别器。相应地，损失函数变为$\mathcal{L}_{cGAN} = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,z))]$。



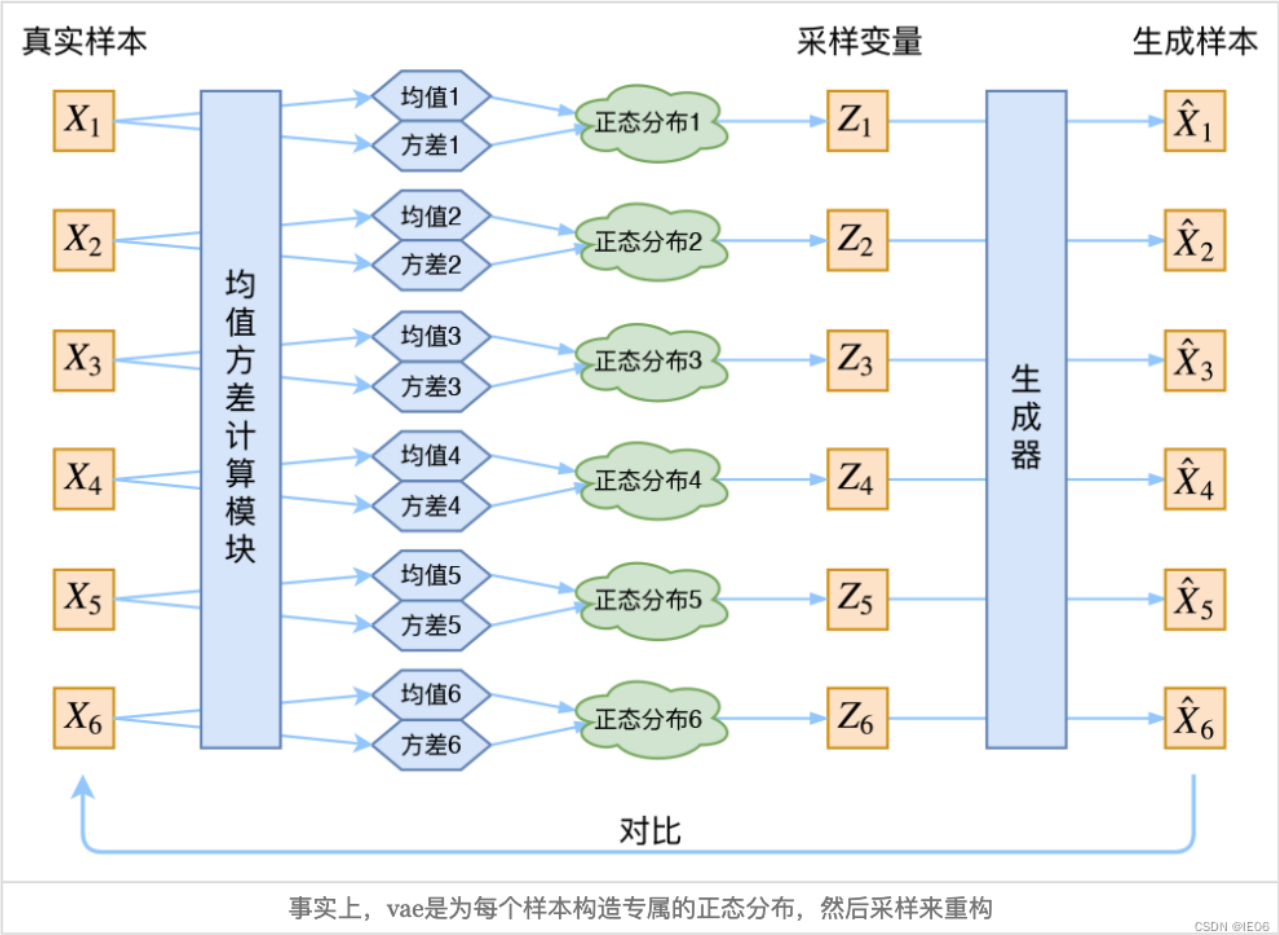
2. 2. 3. 变分自编码器

VAE 的原理稍微复杂，由一个编码器和一个解码器组成。首先需要明确为什么已经有了自动编码器我们还需要 VAE。这是因为自动编码器编码得到的隐变量空间是离散的，一个观察总是会得到同一个确定的隐变量。如果一个输入是在训练集中没有见过的，那么编码的效果很差，进而会导致生成数据的结果很差。而引入了随机变量的 VAE 很好地克服了这一缺点，生成数据的能力更强。

编码器不再认为一个观察x对应一个确定的隐变量z，而是认为一个观察对应着一个随机分布，即隐变量和观察的后验分布$p(z|x)$。编码器得到一个观察作为输入时，计算均值和方差。编码器实现了降维和泛化。

解码器从编码器计算得到的分布中随机采样作为输入，输出生成的数据，它希望从隐变量空间中恢复出输入的数据，也就是试图从$p(z|x)$恢复出$p(x|z)$。这个过程需要用到隐变量空间$p(z)$的分布，在这里一般人为设定为高斯分布，也就是给予他一个先验，希望学习得到的隐变量空间逼近先验。

总的来说，我们希望 VAE 能够把其他方法难以求解的后验分布$p(z|x)$给学出来。将其用 KL 散度表达出来并最小化，经过变换可以得到损失函数$\mathcal{L} = \| x - d(z) \|^2 + KL[N(\mu_x, \sigma_x), N(O, I)]$。其中前半部分表示的是，希望生成的图片要与真实的图片接近，也代表着解码器解码得准，后半部分则代表着隐变量空间与先验接近。



2.2.4. 用无标签数据学习

本节探讨的主要是无监督学习的问题。没有了标签的约束，模型学习到的映射可能是千奇百怪的，因此，在无监督学习中需要给模型添加一些额外的约束。主要有以下三种：

循环一致性损失（cycle consistency loss）。从输入生成输出后，还要从输出重新生成一个输入，确保真实的输入和生成的输入的一致性。其实也就是确保了映射是一种双射。

距离保持损失（distance preserving loss）。输入空间中两个输入间的距离，应该与输出空间中对应的两个输出间的距离一致。

权重共享策略（weight sharing strategy）。在多个域的学习中，如联合分布，使用共享一部分权重的策略来使得多个模型可以在高级抽象层或隐变量空间中保持一致。

2.3. 神经渲染方法的分类

该节的内容主要讲述了如何根据五类标准对神经渲染的方法进行分类。

按照控制方式（control）分类。控制方式指给神经网络提供控制信号的方式，主要有直接传递场景参数作为输入；将场景参数平铺到输入图像的所有像素上或将场景参数连接到网络的激活层；利用图像的空间结构信息和一个图像-图像的转换网络实现从参考图到输出图的映射。在我看来，前两种类似于如何将隐变量映射为网络参数，第三种方法即是用数据驱动实现映射。

按照图形学模块（CG modules）分类。神经渲染不可避免地用到了许多传统图形学的方法和策略，可以根据图形学方法分类。

按照显式控制和隐式控制（explicit vs. implicit control）分类。根据是否可以对合成的内容进行显式的控制，可以将神经渲染方法分类。

按照多模态合成（multi-modal synthesis）分类。多模态合成指模型应该能提供多个结果供选择，其实也就是根据随机性对神经渲染方法做了分类。

按照泛化能力（generality）分类。

3. NeRF

3.1. 引言

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis 是 2020 年 ECCV 上获得最佳论文荣誉提名的工作，其使用二维的图像进行监督，用 MLP（Multi-Layer Perceptron）学习场景的辐射场表达。NeRF 完成的任务主要是新视角合成，与传统的 IBR 方法形成了鲜明对比。总的来说，该工作的主要贡献有如下三项：

一个使用 MLP 来表示一个连续的场景（continuous scene）的方法，这个场景可以具有复杂的几何形状和材料，被表示为一个五维的神经辐射场。

一个基于经典的体渲染（volume rendering）的可微的渲染流程，包括了一个渲染时采用的分层采样策略。

用位置编码把五维的空间坐标映射到了高维空间内，使得神经辐射场有能力学习到高频的场景内容。

3.2. 神经辐射场

NeRF 使用一个 MLP 拟合出一个连续的辐射场来表示一个场景，这个 MLP 代表了场景的辐射场，渲染时即使用这个辐射场来进行渲染。

3.2.1. 输入

MLP 的输入为空间坐标$\boldsymbol{x}$和视角方向$\boldsymbol{d}$。空间坐标$\boldsymbol{x}$为三维，即(x, y, z)，代表相机将要观察的点。视角方向$\boldsymbol{d}$为二维，即(θ, ϕ)；在输入 MLP 之前，视角方向会被转化为笛卡尔坐标系下的三维单位向量，代表着相机从何方向观察点。即 MLP 的输入为一个六维的向量，由一个三维的空间坐标和一个三维的视角方向组成。

3.2.2. 输出

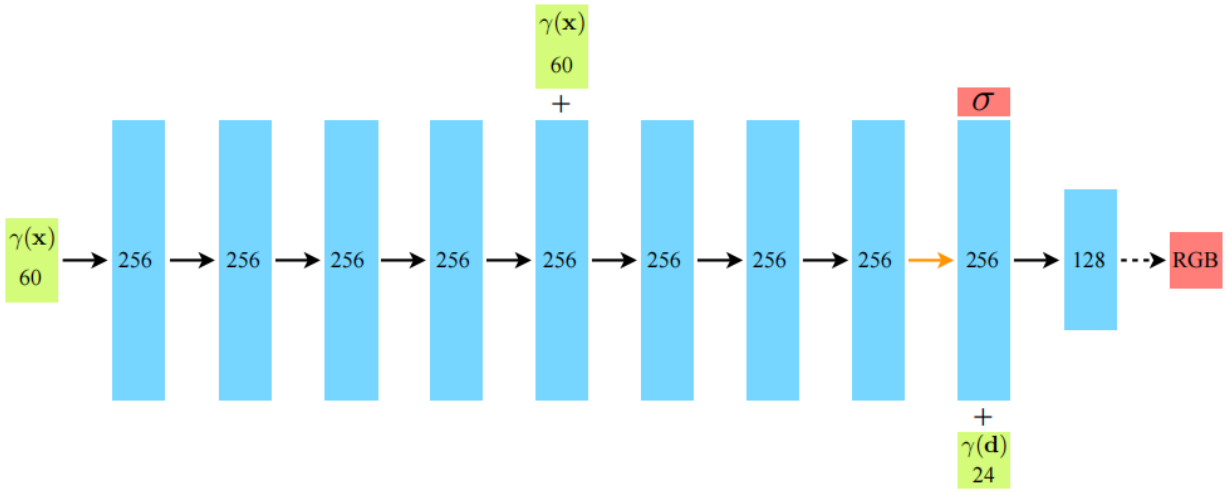
MLP 的输出为点的体密度σ和该点发出的颜色$\boldsymbol{c}$。点的体密度为一个标量σ，其计算由空间坐标决定$\sigma(\boldsymbol{x})$，代表着光线终止在位于该点的微粒的概率。点散发的颜色为一个三维向量(r, g, b)，即使用 RGB 来表示颜色。点散发的颜色与点的位置和观察的视角有关，因此可以被表示成$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{d})$。在 NeRF 中，每个场景都被处理成发光实体，MLP 直接记录点往不同方向散发出的光（颜色），而不是记录表面的反射等物理属性。

3.2.3. 网络结构

MLP 的网络结构如下图。输入的空间坐标经过位置编码变成了 60 维的向量，视角方向经过位置编码变成了 24 维的向量。

网络的所有层都是全连接层，除了最后一层外每一层都有 256 个频道。黑色箭头代表着使用了 ReLU 激活函数，橙色箭头代表不使用激活函数，而虚线黑色箭头代表使用了 sigmoid 激活函数。

可以看到倒数第二层输出体密度和一个 256 维的特征向量，体密度在输出前经过 ReLU 函数的处理以确保非负性。特征向量和视角方向连接后输入到 128 维的最后一层，得到点$\boldsymbol{x}$由视角$\boldsymbol{d}$观察到的颜色。这里通过连接使用的空间坐标和视角方向即通过连接的形式起到了调节的作用。



3.2.4. 训练

NeRF 采用分层采样，因此需要训练两个 MLP，一个用于粗糙地采样，一个用于精细地采样，分别用下标c和f表示。

NeRF 使用 COLMAP 来从真实图片中估计相机位姿等参数。在每轮优化迭代中，NeRF 从训练集所有的像素中随机采样一批光线（像素），并采样、渲染出这些像素的颜色，将生成的像素颜色与真实的像素颜色进行逐个比较。

因此 NeRF 采用的损失函数需要同时把两个 MLP 的训练考虑在内，NeRF 采用均方误差的形式来定义损失函数，并将两个 MLP 的损失求和，即$\mathcal{L} = \sum_{\boldsymbol{r}\epsilon\mathcal{R}}[\| \hat{C}_c(\boldsymbol{r}) - C(\boldsymbol{r}) \|_2^2 + \| \hat{C}_f(\boldsymbol{r}) - C(\boldsymbol{r}) \|_2^2]$。这里的$\mathcal{R}$即被随机采样到的一批光线，$\boldsymbol{r}$即这批光线中的其中一束，$C(\boldsymbol{r})$即训练集中光线的真实颜色。

3.3. 体渲染

体渲染是一种展示一个三维离散数据结构的二维投影的方法，这个离散的数据结构通常是规则的，如体素。要使用体渲染，通常需要先定义相机的位姿和每个体素的不透明度与颜色（常用 RGBA 表示）。体素的不透明度在 NeRF 中即为体密度（volume density），颜色即为点散发的 RGB 颜色。要从离散数据结构中渲染出图像，体渲染常常使用光线步进的算法。

3.3.1. 体密度与透光率

体渲染中有两个重要概念，一个是体密度σ（volume density），一个是透光率T（transmittance，不知道“透光率”这个翻译是否准确）。

NeRF 当中所使用的渲染模型是以粒子作为研究对象的模型。光线经过一个点时，一部分光会从该点经过，一部分光会与该点的粒子互动而终止。光与粒子的互动包括吸收、反射等行为，也就使得该点产生了可被观察到的颜色。从概率学上说，体密度代表着一束光线终止在某个点的概率，从统计学上说，则是反映了某个点的粒子密度。光线不断向前前进，不断有一部分光被路径上的粒子吸收，最后光强必定越来越小，这就需要引入透光率的概念。

以粒子作为研究对象的模型最后会推导得到光线的指数衰减，即某点处的透光率$T(s) = \exp\left(\int_0^s -\sigma(t)dt\right)$（这里的过程比较复杂，我是参考了一个网上的资料，链接在参考资料处）。这里的透光率表示当光线到达s处时，光强还剩下百分之多少。从统计学上说，也就是有百分之多少的光线没有碰撞到粒子。从概率学上说，也就是一根光线到达s处时，没有碰撞到任何粒子的概率。原文给出的解释即为概率学意义的解释。

3.3.2. 渲染方程

首先，NeRF 使用公式$\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$表示相机光线路径上的一个点。其中$\boldsymbol{o}$表示相机光心，$\boldsymbol{d}$表示视角方向，$t$表示点与光心的距离。也就是用向量的形式来表达空间中的一个点。

对于透光率，我们不可能也没必要在一个无限长的距离上进行积分，结合相机视锥，只需要在近端裁切面t_n和远端裁切面t_f之间进行积分。由此透光率可以表示为：

$$T(t) = \exp\left(\int_{t_n}^{t} -\sigma(\boldsymbol{r}(s))ds\right)$$

而最终这根光线最后返回回来的颜色为路径上碰撞到的粒子颜色的期望，表示出来为：

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\,\sigma\big(\boldsymbol{r}(t)\big)\,\boldsymbol{c}(\boldsymbol{r}(t),\boldsymbol{d})dt$$

从统计学上说，这里的$T(t)\,\sigma\big(\boldsymbol{r}(t)\big)$即为有百分之多少的光在点$\boldsymbol{r}(t)$发生碰撞，而$T(t)\,\sigma\big(\boldsymbol{r}(t)\big)\,\boldsymbol{c}(\boldsymbol{r}(t),\boldsymbol{d})$即为该点的颜色。

3.3.3. 使用数值积分

计算机不可能计算连续积分，把上述渲染方程改写成数值积分的形式就得到了：

$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\boldsymbol{c}_i\,, \qquad T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

采样的点通过类似于光线步进的方式选出，即在一段距离上等间距采样，公式中的$\delta_i = t_{i+1} - t_i$即为采样的间隔。实际上，这把采样渲染变成了$\alpha_i = 1-\exp(-\sigma_i\delta_i)$的经典的阿尔法组合（alpha composition）计算。

最后，决定好相机位姿等参数，渲染时根据上述公式计算图像上每个像素点的颜色即可。

3.4. 位置编码

由前面的综述已经知道，MLP 总是倾向于学习一个低频的函数，导致丢失高频的细节。NeRF 采用位置编码来解决这个问题。NeRF 采用了一个具有傅里叶特征的位置编码函数，空间坐标和视角方向被编码后输入 MLP：

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

其中的L为超参数，人为设定，决定了 MLP 能学习到的信号的最高频率。

对于空间坐标，L取 10，对空间坐标的每一个值计算一次，使得空间坐标由 3D 变成 60D。对于视角方向，L取 4，对视角方向的每一个值计算一次，使得空间坐标由 3D 变成 24D。

3.5. 分层采样

前面提到NeRF采用分层采样策略，因此需要同时训练一个粗糙神经网络（coarse network）和一个精细神经网络（fine network）。提出分层采样的目的是希望渲染时能在采样时能增大有用表面的贡献，减小对空白区域和被遮挡区域的重复采样。分层采样策略进行两次采样，过程如下：

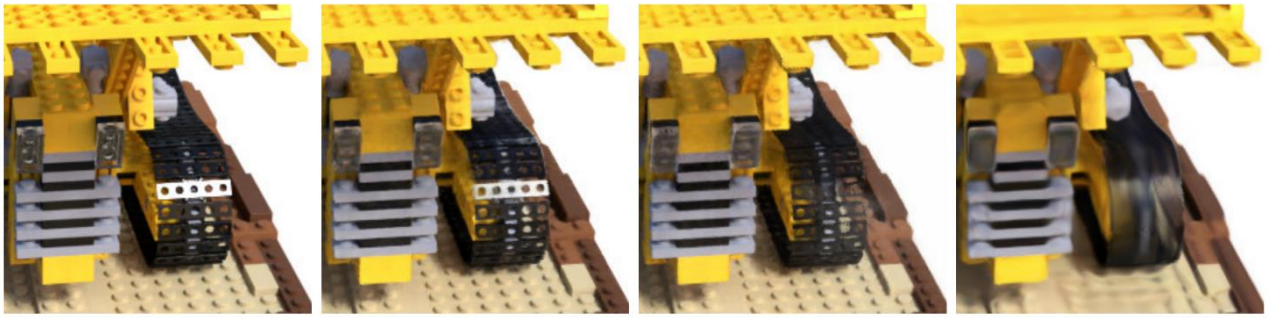
首先在粗糙神经网络中，对光线路径上的近端裁切面和远端裁切面之间的点进行等距采样，共采样N_c个点。根据采样结果，计算每个采样点的权重$w_i = T_i(1 - \exp(-\sigma_i\delta_i))$。这些等距采样点的权重实际上构成了一个概率密度方程，指示着光线上哪一部分对光线颜色的贡献大。

其次在精细神经网络中，在上一次采样得到的概率密度方程的指导下，利用逆变换采样（inverse transform sampling）采样N_f个点。根据$N_c + N_f$个点的结果计算最终的光线颜色。

在源码中，源码根据命令行参数 `N_importance` 决定是否定义精细网络 `network_fine`，定义时会将两个网络的参数一起放入 `grad_vars`，优化器直接优化 `grad_vars`。

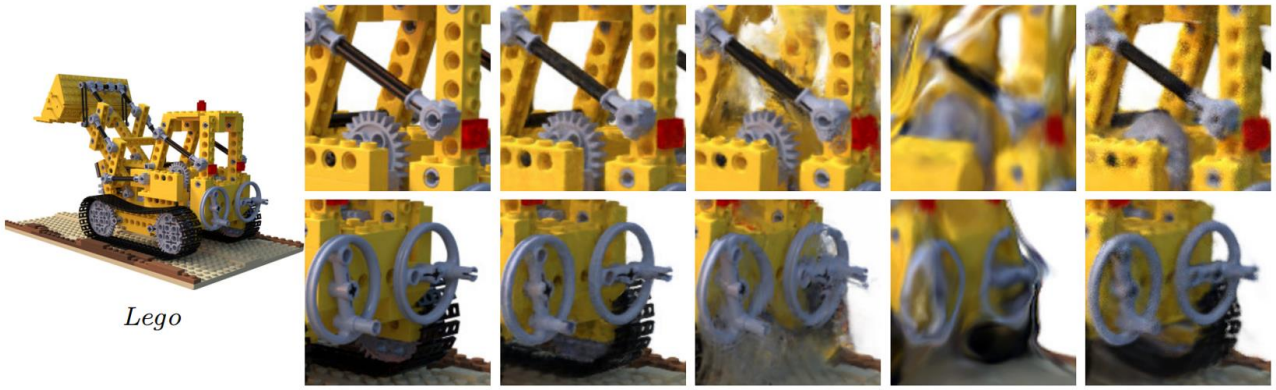
3.6. 结果分析

从下图可以看出，经过位置编码，履带上的细节这一高频信号得以被还原，而没有位置编码的网络对这一细节的处理有严重的涂抹。



Ground Truth　Complete Model　No View Dependence　No Positional Encoding

通过和其他模型的对比可以看出，NeRF 的在新视角合成这一方向取得了巨大的突破。但是与真实图片相比，仍然丢失了一些细节，并且边界显得不那么直。总的来说，可以看出位置编码和分层采样的作用，但是分层采样这一做法时间复杂度太高，速度很慢。



Lego

Ground Truth　NeRF (ours)　LLFF [28]　SRN [42]　NV [24]

4. NeRV

4.1. 引言

NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis 是 2020 年发表

的工作，其在 NeRF 和神经反射场（Neural Reflectance Field）的基础上发展而来。该工作的主要成果有二：结合 NeRF 和神经反射场，实现了一个可以在合成新视角图像时进行重光照的类 NeRF 网络；引入了可见度场（visibility field），提高了渲染速度。

NeRF 展示了使用神经场来合成栩栩如生的图像是可能的，但是 NeRF 把场景处理成发光实体。实体发出的光从何而来？其实是场景中的表面反射的场景中的固定光源的光，也就是在固定光源下的表面的颜色。 NeRV 希望能增强 NeRF 的可编辑性，希望在合成新视角图像的时候可以调整场景光源，即重光照（relighting）。如何进行重光照呢？作者从神经反射场中汲取灵感，用一个神经场记录场景中的表面如何反射光——BRDF。由此实现了一个可以在合成新视角图像时进行重光照的类 NeRF 网络。

在介绍 NeRF 时已经提到，分层采样的策略使得高质量的图像合成成为可能，但也使得渲染、训练的速度大大下降。渲染时，计算可见度（visibility，在 NeRF 中被称为透光率 transmittance）是一个需要采样体密度并计算数值积分的过程，NeRV 直接训练一个可见度场来记录该信息。渲染时不计算而是直接访问可见度场来获得可见度，提高了渲染速度。

4.2. 神经反射场和神经可见度场

NeRV 使用了三个神经网络，分别是形状场（shape field）、反射场（reflectance field）和可见度场（visibility field）。NeRF 本质上是将场景建模为往外发射光线的实体，而 NeRV 则是将场景当做可以反射光线的实体。由于物体颜色的本质就是光照的结果，因此 NeRV 就单独对密度和反射光照建模，而 NeRF 是对密度和颜色联合建模。反射光照的建模基于 BRDF 反射方程，即要求反射场预测出物体表面的镜面粗糙度（specular roughness）和 3D 漫反射率（3D diffuse albedo）。物体表面的法线是可以由密度信息推理出来的。

与 NeRF 一样地，空间坐标和方向在输入网络之前要经过位置编码，在 NeRV 超参数L分别取 7 和 4。

4.2.1. 形状场

NeRV 的形状场$MLP_\theta: \boldsymbol{x} \rightarrow \sigma$记录的是场景物体的密度信息。其输入为一个点的空间坐标$\boldsymbol{x}$，输出为该点的体密度σ。通过密度信息，即形状场本身，可以计算得到发现信息$\boldsymbol{n} = \nabla_{\boldsymbol{x}} MLP_\theta(\boldsymbol{x}(t))$。

该神经网络使用 8 个全连接的 ReLU 层，每个层有 256 个频道。

4.2.2. 反射场

NeRV 的反射场$MLP_\psi: \boldsymbol{x} \rightarrow (\boldsymbol{a}, \gamma)$取自于神经反射场，记录的是场景物体的反射信息。其输入为一个点的空间坐标$\boldsymbol{x}$，输出为该点的 3D 漫反射率 **a** 和镜面粗糙度γ。

该神经网络使用 8 个全连接的 ReLU 层，每个层有 256 个频道。

4.2.3. 可见度场

NeRV 的可见度场$MLP_{\phi}: (\boldsymbol{x}, \omega) \rightarrow (V_{\phi}, D_{\phi})$记录了环境光照可见度（environment lighting visibility）和光线的终止深度（expected termination depth）。其输入为一个点的空间坐标$\boldsymbol{x}$和光线方向ω，输出为环境光照可见度V_{ϕ}和光线的终止深度D_{ϕ}。在 NeRF 中，出于重光照的需要，在表示光源时采用了环境贴图（environment map）的形式。环境光照能见度即环境贴图中的光线，沿着方向ω到达点$\boldsymbol{x}$时的可见度V_{ϕ}。终止深度则指的是，对于点$\boldsymbol{x}$的沿方向ω入射的光线，该光线来自于距离D_{ϕ}的点。二者的意义其实也可以通过计算公式看出来：

$$V_{\phi}(\boldsymbol{x}, \omega) = \exp\left(-\int_{0}^{\infty} \sigma(\boldsymbol{x} + s\omega) ds\right),\ D_{\phi}(\boldsymbol{x}, \omega) = \int_{0}^{\infty} \exp\left(-\int_{0}^{t} \sigma(\boldsymbol{x} + s\omega) ds\right) t\sigma(\boldsymbol{x} + t\omega) dt$$

该神经网络使用 8 个全连接的 256 个频道的 ReLU 层来将位置编码后的$\boldsymbol{x}$映射为一个 8D 的特征向量，该特征向量会与位置编码后的ω连接，再被送入 4 个全连接的 128 个频道的 ReLU 层。

4.2.4. 训练

与 NeRF 的训练类似，在每轮优化迭代中，NeRV 从训练集所有的像素中随机采样一批光线$\mathcal{R}$（像素），并采样、渲染出这些像素的颜色，将生成的像素颜色与真实的像素颜色进行逐个比较。这一部分损失可以表示为$\sum_{r \epsilon \mathcal{R}} \| \tau(\tilde{L}(\boldsymbol{r})) - \tau(L(\boldsymbol{r})) \|_2^2$，其中$\tau(x) = x/(1+x)$为色调映射算子（tone-mapping operator）。

上述损失同时约束了形状场和反射场，但没有约束新引入的可见度场。考虑使用的三个场，能反映场景几何信息，反映环境光照可见度和终止深度的是形状场和可见度场。如果让形状场和可见度场都向真实场景逼近，在训练过程中两个场不断变化，可能会有不一致的问题。NeRV 采用的是让可见度场逼近形状场的方法，用形状场来约束可见度场。在每轮优化迭代中，额外采样一批光线$\mathcal{R}'$（像素），这部分光线需要与场景中的物体有交集，即可以用来计算真实的环境光照可见度和终止深度。这部分损失可以表示为$\sum_{\boldsymbol{r}' \epsilon \mathcal{R} \cup \mathcal{R}', t} (\| \tilde{V}_{\phi}(\boldsymbol{r}'(t)) - V_{\theta}(\boldsymbol{r}'(t)) \|_2^2 + \| \tilde{D}_{\phi}(\boldsymbol{r}'(t)) - D_{\theta}(\boldsymbol{r}'(t)) \|_2^2)$。

通过一个系数$\lambda = 20$将以上两个系数组合起来则得到了最终的损失函数$\mathcal{L} = \sum_{\boldsymbol{r} \epsilon \mathcal{R}} \| \tau(\tilde{L}(\boldsymbol{r})) - \tau(L(\boldsymbol{r})) \|_2^2 + \lambda \sum_{\boldsymbol{r}' \epsilon \mathcal{R} \cup \mathcal{R}', t} (\| \tilde{V}_{\phi}(\boldsymbol{r}'(t)) - V_{\theta}(\boldsymbol{r}'(t)) \|_2^2 + \| \tilde{D}_{\phi}(\boldsymbol{r}'(t)) - D_{\theta}(\boldsymbol{r}'(t)) \|_2^2)$。实际上，作者还向形状场计算出的环境光照可见度和终止深度添加了一个停止梯度（stop gradient），用以避免形状场反过来向着可见度场逼近。这个停止梯度如何实现，原文没有细说，需要到源码中寻找答案。

4.3. 使用神经可见度场的体渲染

NeRV 使用的渲染方程在 NeRF 和神经反射场的基础上演变而来，并且在实现体渲染时使

用了可见度场来加快渲染。

4.3.1. NeRF

NeRF 使用的渲染方程已在前文叙述过，这里仅给出 NeRV 改写过的渲染方程形式。即相机$\boldsymbol{c}$接收到的来自方向ω_o的光为：

$$L(\boldsymbol{c}, \omega_o) = \int_0^{\infty} V(\boldsymbol{x}(t), \boldsymbol{c})\, \sigma(\boldsymbol{x}(t))\, L_{\boldsymbol{e}}(\boldsymbol{x}(t), \omega_o) dt$$

其中，

$$V(\boldsymbol{x}(t), \boldsymbol{c}) = \exp\left(- \int_0^{t} \sigma\big(\boldsymbol{x}(s)\big) ds \right)$$

4.3.2. 神经反射场

神经反射场将场景记录为一个粒子场（a field of particles），记录粒子如何反射入射光，如此使得神经场可以在合成新视角图像时调整点光源的位置。NeRV 给出的相机$\boldsymbol{c}$接收到的来自方向ω_o的光为：

$$L(\boldsymbol{c}, \omega_o) = \int_0^{\infty} V(\boldsymbol{x}(t), \boldsymbol{c})\, \sigma(\boldsymbol{x}(t))\, L_r(\boldsymbol{x}(t), \omega_o) dt$$

其中，

$$L_r(\boldsymbol{x}, \omega_o) = \int_{\mathcal{S}} L_i(\boldsymbol{x}, \omega_i) R(\boldsymbol{x}, \omega_i, \omega_o) d\omega_i$$

上述反射光的计算是在计算一个球面积分，即从方向ω_o观察，计算每一个沿方向ω_i入射的光线L_i经过点$\boldsymbol{x}$反射到方向ω_o的光线和，R即为神经反射场拟合出的 BRDF 方程。但是在神经反射场的假设中，场景只有一个可移动的点光源，因此该反射光计算方程实际上可被简化为

$$L_r(\boldsymbol{x}, \omega_o) = L_{\boldsymbol{l}}(\boldsymbol{x}) \tau_{\boldsymbol{l}}(\boldsymbol{x}) f_r(\boldsymbol{x}, \omega_i, \omega_o, \boldsymbol{n}(\boldsymbol{x}), \boldsymbol{R}(\boldsymbol{x}))$$

其中$L_{\boldsymbol{l}}$为光源，$\tau_{\boldsymbol{l}}$为光源光线到点$\boldsymbol{x}$时的可见度，f_r为一个参数为$\boldsymbol{R}(\boldsymbol{x})$的可微反射模型，$\boldsymbol{n}(\boldsymbol{x})$为点$\boldsymbol{x}$处的发现法线。

4.3.3. 渲染方程

前面已经提到，NeRV 渲染时考虑了直射光（direct light）和间接光，其中间接光是一次反射间接光（one-bounce indirect light）。考虑直射光和间接光，采用神经反射场的计算形式，点$\boldsymbol{x}$沿着方向ω_o出射的光线可表示为一个球面积分：

$$L_r(\boldsymbol{x},\omega_o)=\int_S \big(L_e(\boldsymbol{x},\omega_i)+L(\boldsymbol{x},-\omega_i)\big)R(\boldsymbol{x},\omega_i,\omega_o)d\omega_i$$

$$=\int_S L_e(\boldsymbol{x},\omega_i)R(\boldsymbol{x},\omega_i,\omega_o)d\omega_i+\int_S L(\boldsymbol{x},-\omega_i)R(\boldsymbol{x},\omega_i,\omega_o)d\omega_i$$

其中$L_e(\boldsymbol{x},\omega_i)R(\boldsymbol{x},\omega_i,\omega_o)$即为直射光部分，$L(\boldsymbol{x},-\omega_i)R(\boldsymbol{x},\omega_i,\omega_o)$即为间接光部分。

考虑直射光部分，前面已经提到 NeRV 中的光照被表达为环境贴图的形式，因此L_e可以被表示为：

$$L_e(\boldsymbol{x},\omega_i)=V(\boldsymbol{x},\omega_i)E(\boldsymbol{x},-\omega_i)$$

利用可见度场估计的可见度（这里简化了一次数值积分），点$\boldsymbol{x}$沿着方向ω_o出射到相机的直射光部分可以被表示为：

$$\int_S \tilde{V}_\phi(\boldsymbol{x}(t),\omega_i)\,E(\boldsymbol{x}(t),-\omega_i)\,R(\boldsymbol{x}(t),\omega_i,\omega_o)d\omega_i$$

因此对于相机$\boldsymbol{c}$沿方向ω_o的相机光线的直射光部分可以被表示为：

$$\int_0^\infty V(\boldsymbol{x}(t),\boldsymbol{c})\,\sigma(\boldsymbol{x}(t))\int_S \tilde{V}_\phi(\boldsymbol{x}(t),\omega_i)\,E(\boldsymbol{x}(t),-\omega_i)\,R(\boldsymbol{x}(t),\omega_i,\omega_o)d\omega_i\,dt$$

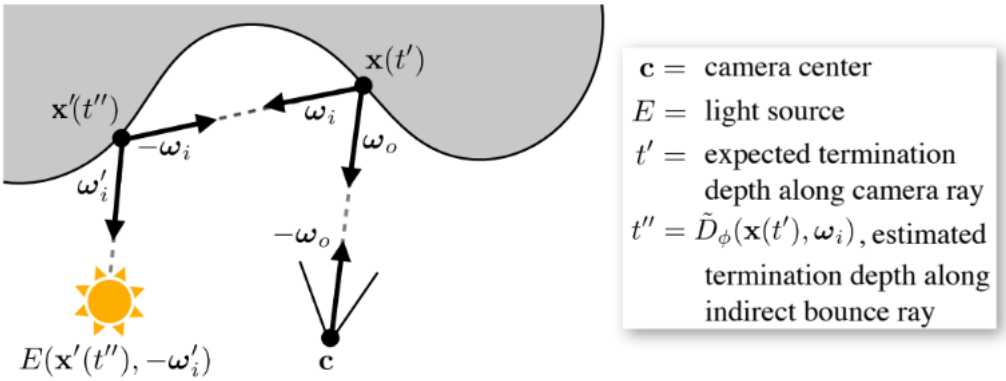


Figure 4: The geometry of an indirect illumination path from camera to light source, and a visualization of our notation.

考虑间接光部分，因为间接光计算的复杂性，作者对间接光的计算做了一些有条件的简化。间接光计算公式的推导过程是沿着光的传播路径逆向推导的。参考上图，首先把物体当作硬表面（hard surface），利用计算直射光时已经求取过的终止深度t'求得点$\boldsymbol{x}(t')$，计算该点沿方向ω_o出射到相机的光线（这里用点的出射光替代了光线计算的数值积分）：

$$\int_S L(\boldsymbol{x}(t'),-\omega_i)\,R(\boldsymbol{x}(t'),\omega_i,\omega_o)d\omega_i$$

这里需要计算入射光$L(\boldsymbol{x}(t'),-\omega_i)$来自于何处，记来自于点$\boldsymbol{x}'(t'')$。$\boldsymbol{x}'(t'')$距离$\boldsymbol{x}(t')$的深度

$t'' = \widetilde{D}_{\phi}(\boldsymbol{x}(t'), \omega_i)$可以由可见度场估计的终止深度求得（这里简化了一次数值积分）。为了避免间接光计算的无限循环，作者将间接光的计算限制为只计算一次反射间接光。再次使用硬表面假设，可以得到点$\boldsymbol{x}'(t'')$沿方向$-\omega_i$射入点$\boldsymbol{x}(t')$的入射光（这里用点的出射光替代了光线计算的数值积分）：

$$L(\boldsymbol{x}(t'), -\omega_i) \approx \int_{S} L_e(\boldsymbol{x}'(t''), \omega_i')\, R(\boldsymbol{x}'(t''), \omega_i', -\omega_i) d\omega_i'$$

结合可以得到点$\boldsymbol{x}(t')$沿方向ω_o出射到相机的一次反射间接光：

$$\iint_{S} \tilde{V}_{\phi}(\boldsymbol{x}'(t''), \omega_i')\, E(\boldsymbol{x}'(t''), -\omega_i') R(\boldsymbol{x}'(t''), \omega_i', -\omega_i) d\omega_i' R(\boldsymbol{x}(t'), \omega_i, \omega_o) d\omega_i$$

把直射光和一次反射间接光结合起来，相机$\boldsymbol{c}$沿方向ω_o的相机光线可以被表示为：

$$L(\boldsymbol{c}, \omega_o) = \int_0^{\infty} V(\boldsymbol{x}(t), \boldsymbol{c})\, \sigma(\boldsymbol{x}(t)) \int_{S} \tilde{V}_{\phi}(\boldsymbol{x}(t), \omega_i)\, E(\boldsymbol{x}(t), -\omega_i)\, R(\boldsymbol{x}(t), \omega_i, \omega_o) d\omega_i\, dt$$

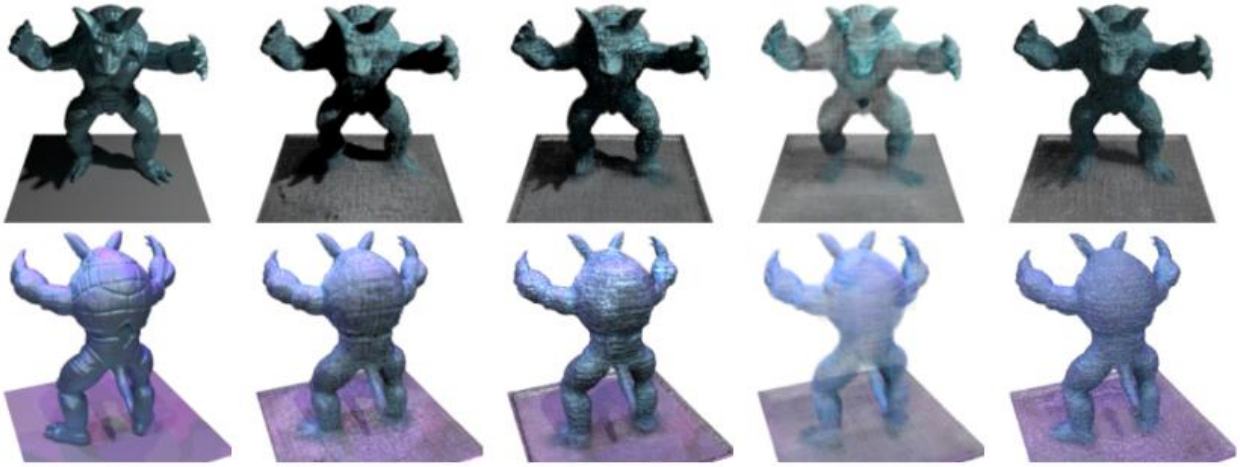
$$+ \iint_{S} \tilde{V}_{\phi}(\boldsymbol{x}'(t''), \omega_i')\, E(\boldsymbol{x}'(t''), -\omega_i') R(\boldsymbol{x}'(t''), \omega_i', -\omega_i) d\omega_i' R(\boldsymbol{x}(t'), \omega_i, \omega_o) d\omega_i$$

有了渲染方程，根据计算机的能力将连续积分化为数值积分，用蒙特卡洛法处理球面积分，结合光线步进算法和可见度场进行渲染即可。

关于硬表面的内涵还不是很清楚，网络上也没有太多介绍资料，还需要再详细了解。

4.4. 结果分析

从下图可以看出，在类似的成果中 NeRV 的成果非常突出。可以看到，即使调整过光源的种类、位置和颜色，NeRV 依然能渲染出一个乍一看还是非常真实的结果。但是仔细看的话，可以看到渲染出来的足部比较虚，地面不平整（或者说不干净），身体纹理有比较多的涂抹。目前我还不清楚哪些部分会导致这些问题，与 NeRF 对比来看，可能是反射的计算和一次反射间接光的估算导致了这一些问题。



(a) Ground Truth (b) Bi et al. Point (c) NeRV (Ours) Point (d) Bi et al. Ambient+Point (e) NeRV (Ours) Ambient+Point

虽然 NeRV 已经使用可见度场加快了渲染速度，但是训练一个场景还是需要一天的时间拟合，这应该是因为引入了反射的计算，加入了几个球面积分。

最后稍微将 NeRV、NeRF 和神经反射场做对比可以得到：

属性	NeRF	神经反射场	NeRV
光源类型	点光源	点光源	环境光
光源数量	单光源	单光源	多光源
光照	固定	动态	动态
渲染方法	物体发光	直射光	直射光+一次反射间接光

5. 讨论

从综述来看，在新视角合成这一应用方向上，在 NeRF 出现之前，最为流行的应该是基于图像的渲染和使用生成模型的方法。即使是基于图像的渲染，也已经引入了许多神经网络来对图片进行组合，用以渲染新视角图像。

NeRF 出现后，人们看到了利用神经网络渲染出高质量的图像的可能，在 NeRF 的基础上发展出了许多研究成果，比如增加重光照能力的 NeRV、显著提高了显示精细细节能力的 Mip-NeRF 和用于室外场景图像的 NeRF in the Wild。但尽管如此，NeRF 类的成果依然还是有着许多问题，其中比较突出的是：

1. 训练速度慢（NeRF 和 NeRV 都需要以天计的训练时间）
2. 只针对静态场景
3. 泛化性差
4. 需要大量视角的训练数据
5. 锯齿

6. 虚影

已经有不少成果在往这些方向改进 NeRF，但依然存在着不小的提升空间。

参考资料

[1]. https://en.wikipedia.org/wiki/Alpha_compositing

[2]. https://autonomousvision.github.io/differentiable-volumetric-rendering/

[3]. https://zhuanlan.zhihu.com/p/512538748

[4]. https://iquilezles.org/

[5]. https://graphics.stanford.edu/courses/cs348b-20-spring-content/uploads/hart.pdf

[6]. https://blog.csdn.net/ranran125/article/details/100076072

[7]. https://zhuanlan.zhihu.com/p/102904841

[8]. https://www.jianshu.com/p/fc12d54d6fed

[9]. https://www.youtube.com/watch?v=KIOSbWNu-Ms

[10]. https://stackoverflow.com/questions/2764238/image-processing-what-are-occlusions

[11]. https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

[12]. https://blog.csdn.net/yunlong_G/article/details/116375556

[13]. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[14]. https://zhuanlan.zhihu.com/p/490440968

[15]. https://zhuanlan.zhihu.com/p/532937226