可微渲染：入门综述

Dezeming Family

2023 年 4 月 26 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

目录

一　介绍

本文的撰写主要翻译和参考自 [1]，并适当补充了一些其他论文或网站上的描述。顺便提一句，看到这个大组写的综述，只能感叹该组的资料储备和科研积累之丰富，像我正在读博的这种只有我一个人做渲染方向的组，永远也不可能发表这种内容丰富的综述了。该综述内容比较通俗易懂，是全面了解可微渲染的很好的材料。为了更好地跟原文对照，我们的描述和翻译的章节顺序都是严格和论文一致的。很多内容我不会在综述中扩展讲解，一是里面至少有一半以上的技术我也没有了解过，二是可能会偏离原文太多，因此我会把一些比较有意思的技术放在其他文章中介绍。本文也不会附带论文原文中的参考文献，遇到需要查阅文献的内容可以去原文进行搜索。本文提供了一些可以当做入门资料的学习网站，放在了参考文献列表中。

可微渲染是一个新的领域，它允许计算 3D 对象的梯度并通过图像传播其梯度。它还降低了对 3D 数据收集和注释的要求，同时在各种应用中实现了更高的成功率。本文回顾了现有的文献，讨论了可微渲染的现状、应用和开放的研究问题。

最近几年已经表明神经网络对 2D 和 3D 推理是有效的，然而大多数 3D 估计方法依赖于有监督的训练机制和昂贵的标注 (annotations)，这使得收集 3D 观测的所有属性具有挑战性。因此，最近一直在努力利用更容易获得的 2D 信息和不同级别的监督来理解 3D 场景。

其中一种方法是将图形渲染过程集成到神经网络管线中，这允许将 3D 估计转换并合并到 2D 图像级别的表征中。计算机图形学中的渲染是生成由几何体、材质、场景灯光和相机属性定义的三维场景图像的过程。渲染是一个复杂的过程，其微分并不是唯一定义的（比如该过程包含了随机估计、多维跟踪），这阻碍了直接集成到神经网络中。

可微渲染（DR）构成了一系列技术，通过获得渲染过程的有用梯度来解决端到端 (end-to-end) 优化的这种集成。通过将渲染微分化，DR 弥合了 2D 和 3D 处理方法之间的差距，使神经网络能够在对 2D 投影进行操作的同时优化 3D 实体。

如下图所示，3D 场景参数的优化的实现可以通过相对于渲染输出来反向传播梯度。通过将渲染层集成到预测的场景参数，并通过以各种方式比较渲染图像和输入图像来应用 loss，从而应用普遍使用的 3D 自监督管线。该过程的应用非常广泛，包括基于图像的 3D 对象重建训练、人体姿态估计、手姿态估计和人脸重建。

尽管有其潜力，但使用现有的或开发新的 DR 方法并不简单。这可归因于四个原因：

- 在过去几年中，已经发表了许多基于 DR 的方法。为了了解哪些方法适合解决某些类型的问题，需要彻底了解这些方法的机理以及潜在的属性。
- 为了选择或开发一种新的 DR 方法，应该知道现有方法的评估方法。
- 为了在新任务中使用 DR，有必要调查 DR 在现有应用程序中的使用情况。然而，由于应用程序的多样性，对该领域有一个清晰的了解并非易事。
- 在过去的几年里，出现了几个 DR 库，每个库都专注于可微分渲染过程的不同方面。这使得一些库对于特定类型的应用程序很有用，而对于其他库，可能需要从头开始实现额外的功能。此外，某些应用程序受到计算需求的限制，而基于 DR 的方法的现有实现往往无法满足这些需求。实时应用程序或嵌入式设备尤其如此，因此需要高度优化的神经网络。

为了解决这些缺点，需要对 DR 的现状进行适当的调查。然而，据我们所知，迄今为止还没有为此目的进行全面审查。

在这项工作中，我们在第 2 节中概述了 DR 算法的现状，在第 3 节中提供了评估指标，在第 4 节中使用可微分渲染的应用程序，以及目前用于促进第 5 节研究的库。除了对现有方法进行调查外，我们还讨论了公开的研究问题，并为未来的工作提供了建议。

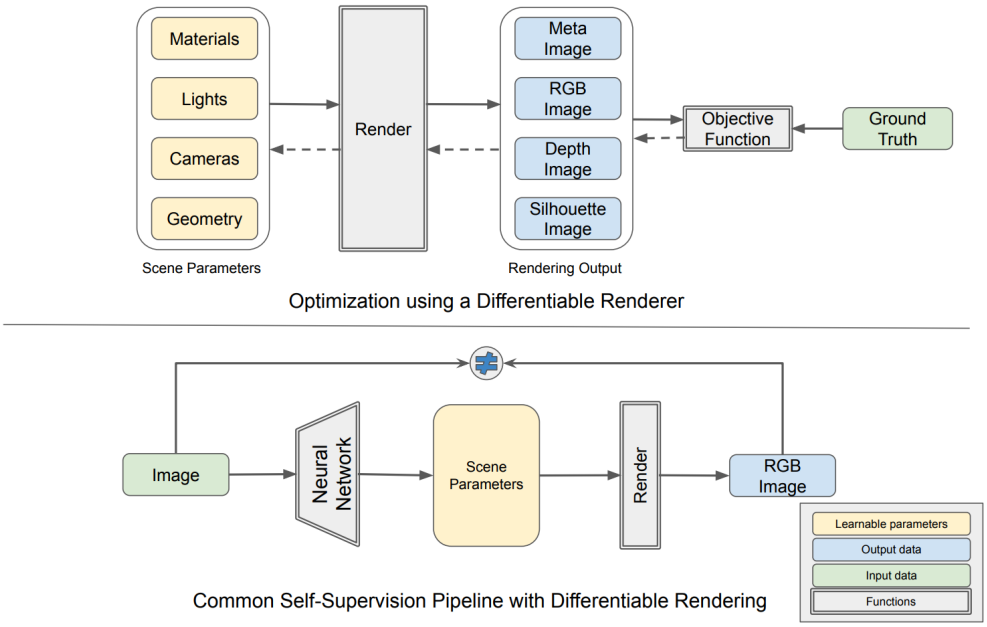

Fig. 1. Schematic overview of differentiable rendering. Best viewed in color. The top part shows a basic optimization pipeline using a differentiable renderer where the gradients of an objective function with respect to the scene parameters and known ground-truth are calculated. The bottom part shows a common self-supervision pipeline based on differentiable rendering. Here, the supervision signal is provided in the form of image evidence and the neural network is updated by backpropagating the error between the image and the rendering output.

上图是可微分渲染的示意图概述。顶部显示了使用可微分渲染器的基本优化管线，其中计算了目标函数相对于场景参数和已知 ground truth 的梯度。底部显示了一个基于可微分渲染的常见自我监督管线。这里，以图像 evidence 的形式提供监督信号，并且通过反向传播图像和渲染输出之间的误差来更新神经网络。

二　算法

首先定义符号，渲染函数 $\mathcal{R}$ 的输入参数有形状参数 Φ_s、相机参数 Φ_c、材质参数 Φ_m 和光源参数 Φ_l；输出参数是 RGB 图像 I_c 或者是一个深度图像 I_d。我们把输入图像标记为 $\Phi = \{\Phi_s, \Phi_c, \Phi_m, \Phi_l\}$，输出标记为 $I = \{I_c, I_d\}$。有些 DR 还可以有其他的输入输出项，但我们这里只关注最一般的项。

可微渲染就是根据计算的输入参数导数 $\partial I/\partial\Phi$，来优化损失函数。这种梯度的计算可以是近似的，但是它需要能够传播有用的信息，以助于最小化目标函数。

输入的几何参数 Φ_s，是可微渲染算法之间的主要区别，每种几何形式对于解决特定问题都有不同的长处。我们把 DR 算法基于底层数据表示来分为四大类：mesh，voxel，point clouds，neural implicit representations：

我们还讨论了神经渲染，因为越来越多的研究使用神经网络模型学习渲染，而不是手动设计渲染及其微分。下表列出了我们在本节中涵盖的文献：

TABLE 1
Overview of the representative differentiable rendering methods. They are classified by the four main underlying data representations.

Data Repr	Type	Literature
Mesh	Analytical derivative	[16], [17], [18]
	Approx. gradient	[19], [9], [20], [14]
	Approx. rendering	[21], [22], [23]
	Global illumination	[24], [25], [26], [27], [28]
Voxel	Occupancy	[7], [29], [30], [31]
	SDF	[32]
Point cloud	Point cloud	[33], [34], [35], [36], [37], [38]
	RGBD image	[39], [40]
Implicit	Occupancy	[41], [42]
	Level set	[43], [44], [45], [46]

2 1　Mesh

2 1.1　渲染的解析导数 (Analytical Derivative)

网格 (mesh) 将三维形状表示为一组顶点和连接这些顶点的曲面。它被广泛使用，尤其是在计算机图形学中，因为它可以以紧凑的方式表示复杂的三维形状。

给定渲染的输入和一个像素，确定其颜色的过程可以分为（1）将三角形分配给像素和（2）基于分配的三角形顶点的颜色计算像素的颜色。前者是通过将所有网格三角形从世界空间投影到屏幕空间并确定包围像素的网格三角形来完成的。然后，选择最靠近相机的三角形，由于该选择产生了一个离散的三角形标识符 (identifier)，因此该运算对于所有参数都是不可微分的。

所有进一步的运算都是可微分的（也就是说，如果没有“选择最近的三角形”这个过程，整个渲染都可以是可微的）。图 2 展示了简化的计算流程。首先，将三角形、灯光的位置和方向从世界空间投影到相机空间（图 2(c)、(d)），然后转换到屏幕空间（图 2 中 (b)）。这些运算是可微分的，因为它们是通过简单的矩阵乘积实现的。然后，像素坐标可以表示为三个顶点的加权和。权重的计算（图 2(a)）是通过求解（可微分）线性程序来完成的，而像素深度是通过在相机空间中插值三个顶点的深度来计算的（图 2 的 (d)）。类似地，像素处的材质和法线向量通常表示为三角形顶点处定义的材质和法向向量的加权和。对于局部照明模型，给定材质和照明参数以及像素处的法线向量，可以使用反射模型计算像素颜色（图 2(e)）。流行的反射模型，如 Phong、Lambertian 和 Spherical Harmonics 都是可微的。因此，可以解析计算像素深度和颜色相对于输入参数的导数。

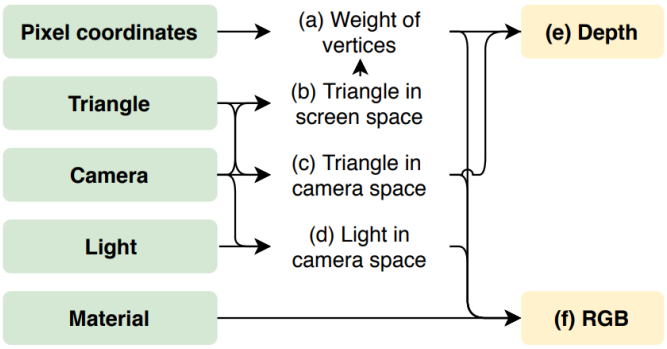

Fig. 2. Several operations that are performed inside a rendering function, given a pixel, its corresponding triangle and material defined on vertices of the triangle, camera parameters, and light configurations. The green boxes represent inputs and the yellow boxes represent outputs. Best viewed in color.

在标准渲染中，通常每个像素只选择一个三角形来计算最终的颜色值，这可能会导致优化问题。例如，让我们考虑一个由一个白色三角形和两个黑色三角形组成的场景，如图 3 所示。顶点颜色 $v_i^w = 1$，$v_i^b = 0$。使用重心坐标 w_i 满足 $v_p = \sum w_i v_i^w$ 且 $\sum w_i = 1$。v_p 的颜色是常量 $c_{v_p} = \sum w_i c_{v_i^w} = 1$。因此，$c_{v_p}$ 的与 v_i^w 和 v_i^b 梯度是 0。

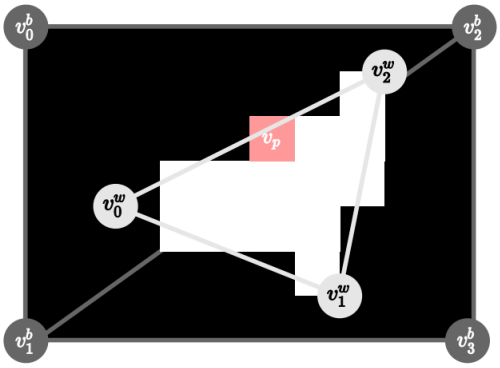

Fig. 3. An image of 10×7 pixels that shows a scene composed of three triangles. The vertex colors of one are white and its vertex positions are denoted by v_i^w. The vertex colors of the other two are black and their vertex positions are denoted by v_i^b.

类似地，对于图 3 所有像素和顶点，像素颜色相对于顶点位置的导数始终为零。因此，在这种情况下，分析导数无助于优化几何图形。然而，在实践中，顶点的位置会影响像素颜色。例如，当 v_2^w 移动到右边，c_{v_p} 会变为 0（此时 v_p 点不在白色三角形区域内了）。因此，我们可以通过允许不相关像素影响临近区域的三角形来解决导数为 0 的情况。有几种渲染方法提供提供近似的梯度，而其他方法通过近似光栅器过程来克服这个问题。

2 1.2 近似的梯度 (Approximated Gradients)

Loper 和 Black 在第一个通用可微渲染器 *OpenDR* 中应用了近似的空间梯度。$v_p = \sum w_i v_i^w$ 且 $\sum w_i = 1$，与像素导数有关的 v_p 可以用可微滤波器（比如 Sobel filter）来计算。换句话说，$\{\frac{\partial c_{v_p}}{\partial x}, \frac{\partial c_{v_p}}{\partial y}\} = \frac{\partial c_{v_p}}{\partial v_p}$，由于计算梯度时位于 v_p 左右两边的像素也都被考虑到了，所以可以得到非 0 值。

Kato 等人提出了 OpenDR 的两个问题，并提出了一种名为神经 3D 网格渲染器（NMR）的渲染器。第一个问题是梯度计算的局部性。由于 OpenDR 中微分滤波器的局部性，只有边界像素上的梯度可以流向顶点，而不能使用其他像素的梯度，基于此属性的优化可能会导致较差的局部极小值。第二个问题是导数没有利用目标应用程序的损失梯度，例如图像重建。举例说明，比如还是图 3，如果目标是减小 v_p 的强度 (intensity)（这里的强度可以理解为是颜色值），我们应该转移 v_2^w。然而，如果目标是增加强度，我们就不能移动它。因此，梯度应具有客观意识，以便更好地进行优化。由于 OpenDR 的目的不是提供准确的梯度，而是提供用于优化的有用梯度，因此为此需要损失感知的梯度流 (loss-aware gradient flow)（也就是说，这个梯度流要跟我们优化的内容有关，但是由于局部梯度范围太小，可能梯度并不能有效地优化

损失函数。）。为了克服这些问题，作者提出了非局部近似梯度，该梯度也使用从损失函数反向传播的像素的梯度。作者后来用类似于 OpenDR 的局部梯度取代了非局部梯度，以降低计算复杂度。

Genova 等人使用每个三角形相对于每个像素的重心坐标计算光栅化导数。他们为位于三角形边界之外的像素的重心坐标引入负值，以克服遮挡的不连续性。通过省略三角形标识符 (triangle identifiers) 并采用负重心坐标，可以将形状视为局部平面，以近似遮挡边界。然而，在优化平移或遮挡时，这种近似可能会带来问题。

2 1.3 近似的渲染 (Approximated Rendering)

其他方法不是近似反向过程（即近似梯度），而是近似渲染的光栅化（或前向过程），以便能够计算有用的梯度。

Rhodin 等人重新解释场景参数以确保可微分性。为了防止硬对象边界处 (hard object boundaries) 的不连续性，每个对象都由密度参数定义，该参数在对象中心具有最大的不透明性，并在边界处变得透明。因此，渲染结果向边缘模糊且平滑，而从场景参数中移除锐利的角 (sharp corners) 可确保可微分性。

Liu 等人采用了类似的方法，并提出了一种名为 Soft Rasterizer 的渲染器。除了空间模糊之外，它还用概率方法取代了 vanilla rasterization 过程中基于 z-buffer 的三角形选择，在概率方法中，投影到像素 p_i 上的每个三角形都以一定的概率贡献其颜色。在实践中，聚合函数融合了每个像素的所有颜色概率。结果，每个像素颜色被计算为与相关三角形相对应的值的加权和，并且该运算是可微分的。权重基于 2D 屏幕空间中像素和三角形之间的距离，以及相机和三角形之间沿着观看方向的距离。因此，梯度以概率的方式在整个图像上累积信息，而 OpenDR 仅从相邻像素反向传播到顶点，NMR 仅反向传播位于 $[\min(obj_x), \max(obj_x)]$ 和 $[\min(obj_y), \max(obj_y)]$ 内的像素的梯度。请注意，上一节中的所有方法都不提供对前向传播的控制，因为它们的目的只是近似反向梯度。

Chen 等人提出了 DIB-R，它独立地聚焦于图像的两个不同区域：前景像素和背景像素，前景像素被至少一个人脸覆盖，背景像素不具有任何人脸覆盖。为了避免 Soft Rasterizer 的模糊输出，DIB-R 建议对前景像素使用解析导数，使用面部顶点属性的重心插值计算。它还通过以类似于 Soft Rasterizer 的方式使用基于距离的全局人脸信息聚合（distance-based aggregation of global face information）来防止背景像素消失梯度。

2 1.4 全局光照 (Global Illumination)

这些技术解决的基本也都是渲染中的不连续性问题（无法求导的问题）。

图 2 中的管线不适用于全局光照模型，因为这种简化的技术中一个像素的照明不受其他表面点反射光的影响。尽管这种简化减少了渲染时间，但无法生成包含光线、几何体和材质的复杂交互的照片级真实感图像。使用全局照明模型中渲染方程的蒙特卡洛估计来计算像素的颜色。可微分真实感渲染的主要挑战是，当蒙特卡洛估计的渲染方程的积分包含由于对象轮廓引起的不连续性时，估计与该积分相对应的导数。

Li 等人是第一项计算基于物理的渲染图像上标量函数相对于相机、光材质和几何体等任意输入参数的导数的工作。它使用了一种基于蒙特卡洛光线跟踪的随机方法，该方法估计像素滤波器积分的积分和梯度。由于边缘和遮挡本质上是不连续的，因此积分计算分为平滑和不连续区域。对于被积函数的光滑部分，采用了传统的自动微分区域采样 (area sampling)。对于不连续的组件，引入了一种新的边缘采样方法来捕捉边界处的变化。他们的方法做出了某些假设：网格没有穿透，没有点光源，没有完全镜面反射的曲面，场景是静态的。Zhang 等人提出了一种非常相似的方法，与 Li 等人的方法不同，他们的方法除了支持三角形网格外，还支持对体 (volume) 的微分。这些方法的两个主要缺点是渲染速度和估计梯度的大方差，这是由于寻找所有物体边缘并对其进行采样的任务具有挑战性，这需要许多样本。

Loubet 等人建议重新参数化所有相关积分，包括球面域上的像素积分，而不是依赖于边缘采样。不连续性发生在那些依赖于场景参数的点上，因此它们从积分中重新参数化变量以消除这种依赖性。这种重新参数化允许在场景参数改变时不发生不连续性的空间上进行积分，并且等效于对不连续性之后的积分进行重要采样。尽管这种方法在计算上是有效的，但它不支持完全镜面材质、包含 Dirac delta 函数的简并光源以及被积函数支持下的多个不连续性。此外，由于梯度是近似的，所以它们可能并不总是准确的。

Zhang 等人提出了一种估计路径积分公式导数的方法，而上述所有方法都解决了渲染方程中的不连续性问题。作者证明了路径积分的微分可以分为内部项和边界项，并提出了估计这两个分量的蒙特卡洛方法。所提出的方法是无偏的，并且计算效率高，因为它不需要显式地找到对象的轮廓边缘。然而，由于单个渲染图像的梯度计算需要几秒到几十秒之间的时间，因此训练神经网络是不切实际的。

与材质和光参数有关的梯度可以通过自动微分来计算。然而，考虑到其巨大的内存占用，应用程序仅限于简单的场景。为了解决这个问题，NimierDavid 等人提出了一种有效的梯度计算方法。在他们的方法中，渲染过程不存储计算图。相反，在反向传播过程中，具有梯度的光线从相机投射，并且梯度在遮挡时传播到曲面。然而，使用这种方法优化形状是具有挑战性的，因为对象可见性的变化是不可微分的，并且在反向传播过程中不考虑可见性。

2 2 Voxel

在本节中，我们将介绍使用体素表示数据的可微分渲染算法。体素是三维空间的单位立方体表示。它可以被参数化为 N 维向量，该向量包含关于在 3D 空间中占据的体的信息以及附加信息。通常的做法是使用二进制值或使用非二进制值对体素中的占用 (occupancy) 信息（简单理解为哪些位置体素值不为 0）进行编码。

对于预测占用 (occupancy) 的应用（预测哪些位置体素值不为 0），通常存储非二进制占用概率 $P_o \in [p_{min}\ p_{max}]$。即使占用概率与透明度值不同，也可以以相同的方式解释它们，以便在光线行进操作期间保持可微分性。在这种情况下，概率 P_o 表示射线在某一点的吸收（透明度）。材质信息也经常被存储。

形状可以表示为从每个体素的中心到对象表面的最短距离，而不是存储 occupancy。在该表示中，每个体素单元被分配一个距离函数（DF）。距离函数可以用有符号值来扩充，该值表示体素是包含在对象内部还是外部，以形成有符号距离函数（SDF）。或者，截断 (truncation) 可以应用于 SDF，以形成截断有符号距离函数（TSDF），用于那些只有到对象表面的距离信息很重要的应用。

在渲染像素时，将考虑沿投影到像素的光线定位的所有体素。目前已经有几种方法来决定结果像素颜色。由于体素在 3D 空间中的位置是固定的，因此在渲染体素时不会出现第 2.1 节中描述的由形状基元的位移引起的梯度问题。收集沿射线定位的体素是渲染过程的第一步。Tulsiani 等人和 Jiang 等人在世界空间中进行这种操作，而 Yan 等人和 Henzler 等人采取了不同的方法。他们将体素从世界空间投影到屏幕空间（使用相机参数），并执行类似于空间变换器 (Spatial Transformer) 的双线性采样，这在计算上更高效。Lombardi 等人的方法截然不同。他们引入了扭曲场的概念（即模板体 (template volume) 和输出体 (output volume) 之间的映射），以提高表观分辨率，减少随着锯齿状运动 (jagged motion) 的网格状伪影 (grid-like artifacts)。估计反向 warp 值，并在 warped 点对模板体 (template volume) 进行采样。由于所有描述的操作都基于体素的子集，因此与它们有关的输出都是可微分的。

沿着采样光线聚集体素是渲染过程的第二步。Yan 等人将占用概率与每个像素相关联。该值是通过从像素投射光线、对所有相关的体素进行采样并选择占用概率最高的体素来计算的。Tulsiani 等人计算射线在一定距离处停止的概率，而不是取最大值。这种方法的优点是除了渲染对象的前景蒙版之外，还能够渲染深度图和彩色图像。Henzler 等人进一步引入了发射材质 (emitting material) 的双向发射-吸收（EA）光辐射模型 (light radiation model)，以及类似于 Tulsiani 等人的视觉外壳 (visual hull, VH) 和仅吸收 (absorption-only, AO) 模型。Lombardi 等人通过从每个像素投射光线，然后沿其迭代累积颜色和不透明度概率，采用可微光线行进 (differentiable ray marching)。当沿着射线的不透明度值的累积和达到最大值时，将丢弃剩余的体素，以防止对其颜色的进一步影响。

与上述方法不同，Jiang 等人处理表示符号距离函数的体素。他们从像素投射光线，并沿着光线定位最靠近相机的表面体素。最终像素颜色被计算为表面体素及其相邻体素的加权和。它们还根据表面点周围的体素值计算表面点处的法线向量，以计算着色值。尽管定位表面点不是可微分的操作，但像素的颜色相对于表面点周围的体素是可微分的。

一些工作在渲染过程中也会考虑材质。Tulsiani 等人、Henzler 等人和 Lombardi 等人简单地使用体素颜色来表示材质。与网格渲染类似，渲染图像相对于材质参数是可微的，因为大多数着色模型都是可微的。

2 3 Point Cloud

点云是表示三维空间中形状的点的集合。它们在 3D 视觉中无处不在，因为它们能够以相对较低的存储成本表示各种拓扑结构。此外，目前市场上可用的大多数 3D 传感器也依赖于点云来编码数据。近年来的研究表明，点云可以成功地集成到深度神经网络中，以解决各种实际的 3D 问题。鉴于可微渲染的出现及其在减少 3D 监督的情况下理解场景的潜力，点云因此成为数据表示的自然选择。

渲染点云分三个步骤完成。首先，计算 3D 点 P^i_{xyz} 的屏幕空间坐标 p^i_{uv}。此操作是通过与网格渲染一样的矩阵乘法来实现的，因此，这个步骤是可微的。其次，计算每个 3D 点对目标像素的颜色的影响 I_i。第三，根据点的影响和 z 值对点进行，以决定像素的最终颜色值。已经提出了解决这一步骤的几种方法。

最直接的计算 I_i 的方法在于假设 p^i_{uv} 的尺寸的一个像素，然而这可能会导致一些非常稀疏的图像（有很多像素没有点被投影上去，造成空白）。解决该问题的一个方法是使得一个 p^i_{uv} 的影响范围不止一个像素，比如截断高斯模糊 (truncated Gaussian blur) 或者影响值基于像素到某个点的距离。当这些操作是在自动微分框架实现的时，这样生成的图像对点而言是可微的。然而，在渲染质量和优化收敛之间有一个权衡，当影响尺寸较大时，可以阻止一个像素与距离 P^i_{xyz} 相关的导数变为 0，但是会降低渲染质量。为了解决该问题，Yifan 等人提出了类似于 Kato 等人的不可见的估计的梯度 (invisible approximated gradient)。

有几种技术被提出，用于计算一个像素的颜色。一个直接的方法是基于点的影响值计算点的颜色的加权和。但是这个方法不考虑遮挡。Lin 等人通过选择离着给定相机像素最近的点来解决这个问题，但是这阻止了被遮挡点 (occluded points) 的优化（被遮挡点在优化时，可能位置移动以后就成了可见点）。Li 等人并不是这么做，他们提出了选择 3D 点 P_{xyz} 到相机每个像素位置的 K 最近邻点，并基于空间影响 I^i 去加权它们（端对端学习点云的局部多视角描述子）。Wiles 等人提出根据到相机的距离加权全部 3D 点，还根据空间影响 I^i 去加权（类似于 Li 等人在 Soft Rasterizer 的研究）。

Insafutdinov 等人采用了一个独特的方法，他们从点云生成 3D 体素，然后再使用 Tulsiani 等人的方法来渲染这些体素，该方法与前面的计算影响值和在屏幕空间聚集点的技术不同。

一组深度图像和相机参数是表示点云数据的替代方式。使用深度图像的主要好处之一是其空间信息可以直接用于恢复关于点之间的三角剖分 (triangulation between points) 的信息。另一方面，当将深度图像投影到 3D 空间时，点的密度 (point density) 与距离成反比地减小（越远的地方，一个像素覆盖的范围就越大，假如每个像素代表点云中的一个点，那么越远的地方点就会越稀疏）。从深度图像生成另一个视图可以被视为点云渲染的特殊情况。在单眼深度估计的自监督学习中，使用彩色图像来估计相应的深度，将其转换为点云并投影到虚拟相机视角。

2 4 Implicit Representations

最近，人们对在神经网络中以参数化的方式表示几何信息越来越感兴趣。这通常被称为 neural implicit representation。在这种模型中，一个点 $P^i_{xyz} \in \mathbb{R}^3$ 的几何信息通过神经网络 $F(P^i_{xyz})$ 的输出来表示。与基于体素的方法不同，在隐式表示中，内存使用相对于空间分辨率是恒定的。因此，可以以无限分辨率重建曲面，而不会占用过多的内存。

与基于体素的方法类似，有三种方法来表示几何信息。首先，点 P^i_{xyz} 被物体占据（比如该点在某个物体上或者物体内部）的概率可以通过神经网络 F 来建模。当给定 P^i_{xyz} 处的 ground-truth 占据时，学习 F 的过程就变成了一个二元分类问题，并且在文献中进行了广泛的研究。其次，F 可以对点 p 处的透明度进行建模。这种方法可以用于表示半透明对象，以及近似没有半透明对象的场景的占用概率。第三，物体的表面可以定义为满足 $F(P^i_{xyz}) = 0$ 的点 P^i_{xyz} 的集合，这被称为水平集方法 (level-set method)。通常，$F(P^i_{xyz})$ 定义到目标表面边界的距离，而其符号指示 P^i_{xyz} 是在表面内部还是外部。

与其他表示一样，对于真实世界的场景，获得真实的 3D 形状通常是昂贵的或不可能的。为了解决这个问题，一些工作建议使用 2D 监督（以深度图或多视点图像的形式），并利用可微渲染。与基于体素的方法类似，已经为近似占有概率、占有概率和透明度以及通过距离函数的隐式曲面开发了各种可微分渲染算法，隐式表示也需要实现来处理不同的输入类型。表 2 总结了基于体素的方法和基于神经隐式函数的方法之间的相似性。

TABLE 2
Comparison of differentiable rendering methods for voxels and neural implicit functions.

Operation	Approach	Voxel	Neural implicit functions
Data collection along a ray	Sampling along a ray	[29], [31], [32]	[42], [43], [44], [45], [46]
	(Re)sampling in 3D space	[7], [30]	[41]
Data aggregation along a ray	Approx. occupancy: taking the maximum value	[7]	[41]
	Transparency: taking a weighted sum	[29], [30], [31]	[42]
	Level set: taking the value at a hit point	[32]	[43], [44], [45], [46]

Liu 首次提出在可微渲染中使用神经隐式表示（论文:Learning to infer implicit surfaces without 3d supervision），为了找到像素 p_{uv} 的占用概率，从 p_{uv} 投射射线 R。然后，选择沿着射线 R 具有最大 occpancy 概率值的 3D 点 p_{xyz}。最后，p_{uv} 的占用概率被赋予 p_{xyz} 的值。在反向传播过程中，p_{uv} 的梯度值被分配给 p_{xyz}。相比之下，Mildenhall 等人提出了神经透明度的可微渲染。在他们的论文中，像素值是通过体渲染和对光线上所有采样点的值进行加权来计算的。因此，梯度会流入到多个点，这有望稳定优化。尽管 Liu 等人仅支持轮廓的渲染，但 Mildenhall 等人支持使用纹理场渲染 RGB 图像。Mildenhall 等人工作的另一个重要方面是，他们在计算颜色值时考虑了光线的方向，因此支持与视图相关的现象，如镜面反射。

采用神经隐式函数的几项工作通过水平集方法表示表面，其中从每个像素投射光线，并使用光线和表面之间的交点来计算相机到表面的距离。交叉点处的颜色是从神经网络中采样的。距离相对于交点的导数不能通过 autodiff 框架计算，但可以通过解析计算。这些方法无法渲染可微分 silhouette（剪影，类似于背景的意思）图像，因为对于背景像素来说，到表面的距离是无限的。因此，为了在考虑轮廓的同时进行优化，通常分别对前景和背景像素使用不同的损失函数。例如，Niemeyer 等人当遇到投影到 silhouette 为假阳性 (false positive) 时，最小化交叉点的占用，而当投影为假阴性 (false negative) 时，最大化沿射线的随机点的占用。

对于基于体素的方法来说，在射线上采样点很容易，因为由于 3D 网格的离散性质，射线上有一组有限的体素。对于神经隐式表示，光线上有无限多个点，采样过程变得很有挑战性。解决这个问题的两种最常见的方法是对随机点或具有随机扰动的规则点进行采样。另一种方法是对 3D 空间中的随机点进行采样，并检查它们与光线的交叉点，当渲染多个视图时，这将变得更加有效。为了进一步优化，许多方法对射线采用从粗到细的采样和边界附近的重要性采样。

2 5 Neural Rendering

Eslami 等人建议从数据中学习渲染过程，而不是手动去设计渲染微分。这种方法通常被称为神经渲染。通常，通过最小化图像重建误差来联合训练输出神经场景表示的场景生成网络和渲染网络。

由于神经网络的最新进展，神经渲染如今能够生成高质量的图像，并被用于许多应用，如新颖的视图合成、语义照片处理、面部和身体再现、重新照明、自由视点视频以及照片逼真化身的创建。虽然手工实现渲染器并不总是准确地建模物理世界，但通过从真实世界的数据中学习，神经渲染可以生成与真实世界无法区分的新图像。

另一方面，推广到与训练数据不同的场景，扩展到由多个对象组成的场景以及让神经网络场景具备由人类修改的能力是有限的。改进神经渲染的一个很有前途的方向是为 3D 场景添加归纳偏差 (inductive biases)，并将其渲染到神经网络中。因此，将基于归纳偏差的可微渲染器与神经渲染相结合将是一个有趣的研究领域。

2 6 Summary

我们提出了基于四种数据表示类型分组的可微渲染技术：网格、体素、点云和隐式函数。图 4 说明了这些表示：

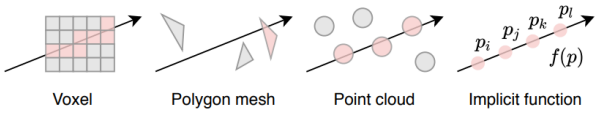

Fig. 4. Differentiable rendering algorithms differ in how the geometric information along a ray is collected and aggregated. For voxels, collecting geometric information is done by checking intersections of a ray and each voxel. For meshes, unlike non-differentiable rendering, multiple polygons have to be associated with a single ray. For point cloud, there are several ways to measure the influence of a point to a ray by pseudo-sizing. For neural implicit functions, various sampling techniques have been proposed for efficiency. Aggregation methods mainly depend on whether geometric information is treated deterministically or probabilistically, and how occlusion is handled.

本节的要点可概括如下：

基于网格的方法可以分为三类：近似梯度、近似渲染和全局光照。梯度近似允许高效和高质量的光栅化，同时旨在手工设计有意义的梯度。渲染近似可能会产生模糊的输出，但会确保所有像素的梯度为非零。基于全局光照的技术减少了渲染图像和真实世界数据之间的域差距，但由于其高计算成本，目前不适用于深度神经网络。

基于体素的方法很容易使用，但需要过多的内存和参数。沿着射线收集和聚集体素，以产生最终的像素值，是基于体素的方法的主要区别因素。尽管这种方法简单且强大，但其适用性仅限于小场景和低分辨率。基于 SDF 的体素方法允许比基于占有网格的方法更平滑地表示表面。

基于点云的方法在面临大小模糊的情况下提供了较低的计算成本。点云是许多可微渲染方法的自然选择，因为它们简单且广泛用于 3D 传感器，但无法捕捉密集的表面信息。在遮挡的情况下，选择一个好的点大小并决定渲染像素的颜色并不简单。已经提出了不同的方法来解决这些问题。

隐式表示可以是点云、体素和网格的可行替代方案，但在沿射线采样点时计算成本很高。隐式函数通过隐藏概率、透明度或到表面的距离来描述三维点处的几何体。可以以几乎无限的分辨率和低存储器成本来表示广泛的拓扑。尽管有这些优点，但在射线上聚集数据可能需要大量计算，因为对数据进行采样需要在大量点上对神经网络进行估计。

表 3 显示了不同表示和渲染方法的强度和限制的摘要。

TABLE 3
Comparison between different algorithm types.

Representation	Type	Strengths	Limitations
Mesh	Analytical derivative Approximated gradient Approximated rendering Global illumination	Usage of advanced forward rendering Usage of advanced forward rendering Auto-diff support Realistic rendering outcome	Local minima in geometry optimization Handcrafting gradient calculation Not precise rendering result Computationally too expensive
Voxel	Occupancy / transparency Signed distance function	Simple, easy to optimize Efficient ray trace	Excessive memory consumption Not suitable for transparent volume
Point Cloud	Point cloud RGBD image	Easy to render and differentiate Ordered point cloud by default	Pseudo-sizing, lack of surface Point density reduces with the distance
Implicit	Occupancy / transparency Level set	Simple, easy to optimize Clear object boundary	Inside and outside is ambiguous Difficult to optimize

2 7 Open Problems

当前的可微渲染方法没有解决一些问题。

除了 3D 对抗性示例 (3D adversarial examples) 和风格迁移 (style transfer) 之外，许多应用程序，如 3D 形状估计和姿态估计，通过最小化渲染图像和目标图像之间的差异来训练神经网络。在这样的管线中，渲染函数和图像比较函数通常是单独开发的。因此，比较函数不能利用被反馈到渲染函数中的丰富的 3D 信息。

然而，它们也可以一起考虑。如第 2.1 节所述，可微分渲染的目的不是提供精确的梯度，而是提供有助于优化的梯度。这可以通过采用 3D 模型和目标 2D 图像的可微的渲染和比较函数 (differentiable render-and-compare function) 来实现，而不是采用 3D 模型并输出图像的可微渲染函数。这个方向上的一种方法是对与 3D 模板形状 (3D template shape) 的顶点相关联的关键点进行可微投影和比较。尽管这种方法是特定于任务的，但它可以通过在渲染和图像比较函数中引入 3D 信息来推广。这些信息的整合仍然是一个

开放的研究领域。

当前基于局部照明的可微分渲染方法非常简单，无法生成显示阴影和反射的照片级真实感图像。此外，全局照明方法对于训练神经网络来说太慢了。另一方面，在游戏引擎中，实时渲染方法的进步使得在不使用计算密集型路径跟踪的情况下渲染高度逼真的图像成为可能。然而，在可微分渲染的背景下，这两者之间的区域尚未被探索。集成快速但复杂的实时渲染方法，如阴影映射和环境映射，是一种尚未尝试过的方法，但有可能提高渲染质量。视频的可微渲染也是一个值得探索的有趣研究方向。为了实现这一点，需要将可微渲染与物理模拟器集成（以包含额外的物理约束）。理论上，应该可以训练将视频数据与可微物理模拟器 (differentiable physics simulators) 相结合的端到端管线，但尚未进行实验。

由于渲染假设一个物理模型来生成图像，因此如果物理模型和现实世界之间存在很大差异，则渲染的图像将显得不现实。另一方面，神经渲染可以产生高度逼真的图像，因为它几乎不对物理模型进行假设。然而，这种缺乏假设的情况有时可能会产生违反物理模型的图像。例如，在神经渲染中，对象的形状可以根据视点的不同而变化。由神经网络渲染的对象和场景的图像不能保证不同视点之间的形状一致性。虽然将对物理世界的归纳偏见纳入神经渲染的方法很重要，但将基于学习的方法纳入可微渲染也是值得考虑的。当相机移动时，人类可以直观而即时地理解场景将如何变化。这种能力是基于过去的经验，而不是计算大脑中每个像素的值。因此，这种学习可以促进梯度计算。

基于学习的方法已经用于渲染和高效光线采样中的图像去噪。这些方法在可微分渲染中也可能被证明是有用的。

三 Evaluation

可微分绘制方法的评估不是一个微不足道的问题，因为绘制是一个复杂的函数。在本节中，我们回顾了评估算法的常见做法，并提出了它们的问题。一种简单的方法是直接梯度评估。

对于基于全局照明的方法，当渲染积分包括可见性项（如对象边界）时，有效的梯度计算方法仍然是一个开放的研究问题。由于目标是计算分析正确的梯度，因此使用有限差分计算的梯度作为 ground-truth。然而，由于缺乏用于评估的通用数据集，因此无法对算法进行定量评估。

另一方面，一些论文专注于计算局部照明模型的近似梯度。在这种情况下，梯度对于优化应该是有意义的，而不是解析正确的。由于这个原因，有限差分不能用于 ground-truth。它们只近似于解析正确的导数，因此可能不适用于优化。Loper 和 Black 将他们的方法计算的梯度与有限差分计算的梯度进行了比较，并声称所提出的方法更好。他们的推理是，在有限差分中确定一个好的 epsilon 用于优化是具有挑战性的。在目标函数优化过程中，可视化梯度（不显示 ground-truth）并分析其收敛效率是另一种用于估计的方法。

估计优化的场景参数是正在采用的另一种方法。由于缺乏用于比较的通用数据集，导致每篇论文都使用自己的数据集来评估算法。许多论文不是直接优化 3D 场景的参数，而是训练神经网络进行单视图 3D 对象重建，并报告重建精度。

计算时间也是一个重要的度量标准，尤其是对于基于光线跟踪的渲染方法。因此，有几篇论文将计算时间作为评估方法的一部分。

在评估局部和全局光照模型时，可以注意到两个主要差异。对于全局梯度，由于算法的目的是计算解析正确 (analytically-correct) 的梯度，因此有限差分可以用作 ground-truth。对于局部梯度，目的是近似有用的梯度，因此应使用优化结果 (optimization results)。然而，在这两种情况下，由于缺乏共同的数据集，无法对不同的算法进行公平的比较。一种可能的解决方案是创建一组 toy problems，这些问题可用于评估几何、材质、光线和相机参数的导数或优化。因此，有必要为不同的渲染提供一种公平、准确和有意义的新的评估方法。

四 Application

可微渲染已被用于解决计算机视觉和计算机图形学的各种 3D 相关问题。在本节中，我们回顾了可微渲染的应用，并讨论了它们的局限性。表 4 显示了 DR 的代表性应用。

TABLE 4
Applications and representative literature of differentiable rendering.

Application	Literature
3D object reconstruction	[9], [20], [7], [8], [68], [22], [23], [69], [38], [41], [43], [62], [70], [71], [30], [72], [73], [74], [45]
Body shape estimation	[11], [10]
Hand shape estimation	[13], [75], [76]
Face reconstruction	[77]
Object/camera pose estimation	[78]
Object part segmentation	[79]
Material estimation	[17]
Light/shading estimation	[80], [81], [82], [83]
Adversarial examples	[84], [85], [86], [87], [88]
Auto-labeling	[46]
Teeth modeling	[89]

4 1 Object Reconstruction

单视图 3D 对象重建是根据单个图像估计对象的 3D 形状的任务。与使用多个图像的多视图立体和运动结构不同，这项任务是通过机器学习而不是几何估计来解决的。现代方法能够从自然图像中学习高质量的 3D 重建，如图 6 所示。

Fig. 6. Single-view 3D object reconstruction by Kanazawa *et al.* [90]. The leftmost column shows input images, and the rest are reconstructed objects.

单视图 3D 对象重建的一种直接方法是使用真实 3D 形状进行监督学习。监督学习需要对与图像相对应的 3D 形状进行注释，这在时间和工作量方面是一个昂贵的过程。通过用 2D 注释替换 3D 注释，可以显著减少这种负担。通常采用自监督来实现这一点。由于可微渲染允许从图像空间观察 3D 场景参数，因此输入图像可用于创建监督并帮助训练单视图 3D 对象重建管线，如图 5 所示。

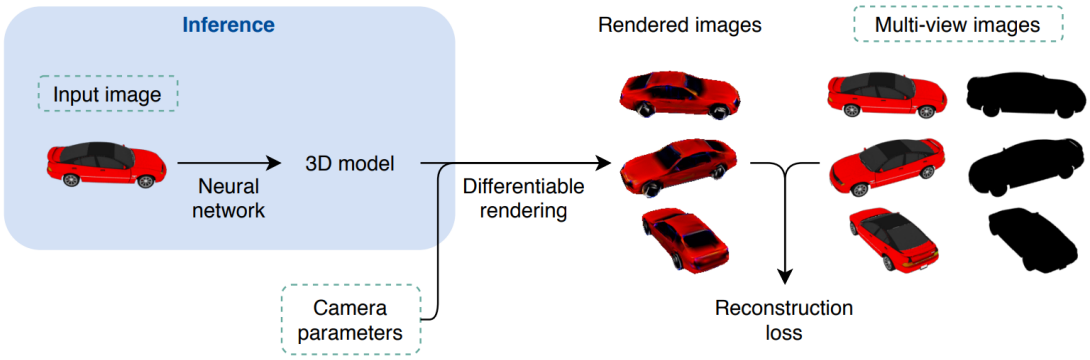

Fig. 5. Standard training pipeline of learning single-view 3D object reconstruction from 2D images. Dashed rectangles represent training data.

早期的工作使用不可微分 (non-differentiable) 的 OpenGL 渲染器基于策略梯度算法进行单视图 3D 对象重建。然而，要重建的一组形状仅限于粗略和简单的形状。基于体素的、基于网格的、基于点云的和基于神经隐式函数的可微渲染的进步显著提高了重建精度。

尽管图 5 中的管线允许学习高质量的 3D 对象重建，但它有两个主要问题。首先，收集对象的多视图、

真实世界的图像通常是不可能的，或者成本太高。其次，为一些场景参数（如相机和灯光）创建准确的标注 (annotations) 对人类来说很困难。

第一个问题的一个可能的解决方案是使用每个对象的单个图像而不是多视图图像。然而，由于其不适定性质，当在训练期间使用单个视图时，3D 形状的重建变得模糊 (ambiguous)。为了说明这一点，Kanazawa 等人建议使用隐含包含粗略 3D 信息（例如鸟类的喙）的语义关键点的重投影误差作为额外的训练目标。同样，Liu 等人提出以自监督的方式使用语义部分的重投影误差。尽管大多数方法都没有对光照效果进行建模，但 Henderson 和 Ferrari 明确地将光作为渲染器的额外输入，以使用着色来帮助减少重建的 3D 形状的模糊性。Kato 和 Harada 建议使用对抗训练来重建从任何角度看都逼真的形状。

对于第二个问题，一些研究试图消除或减少对相机参数注释的需要。Tulsiani 等人、Insafutdinov 和 Dosovitskiy、Henderson 和 Ferrari 提出将相机参数估计网络集成到形状估计网络中，并使用预测参数进行渲染，而不是使用 ground-truth 参数。Henzler 等人的训练目标不是提高重建质量，而是能够从随机视点渲染逼真的图像。另一方面，Yang 等人提出了一种用于学习相机参数的半监督管线。

该领域的其他主题包括学习纹理 3D 对象的生成模型，通过利用 3D CAD 模型重建驾驶场景中的汽车，以及使用可微渲染来微调使用 3D 监督训练的形状估计模型。

4 1.1 Limitations and Open Problems

学习从各种类别的自然图像数据集中重建物体是一个可能的研究方向。尽管不需要 3D 监督 (3D supervision) 是从图像中学习的优势之一，但合成数据集在实验中被广泛使用。这样的数据集是在受控的环境中生成的，在该环境中对象不被遮挡，图像质量非常高，轮廓分割几乎没有噪声。

然而，在现实世界的应用中，这些假设往往不成立，这使得合成数据集的使用对于此类场景来说不切实际。常用的合成数据集之一是 ShapeNet，但其他方法使用噪声较大的自然图像数据集，如加州理工学院 UCSD Birds-200-2011 和 PASCAL 进行训练。由于可用数据集的固有限制，可训练对象类别仅限于鸟类、汽车、飞机和椅子，因为相机参数估计需要关键点标注。依赖于自然图像的新方法将是实际应用的关键。或者，从视频或场景元素的交互中学习仍然是一个具有挑战性的问题。人类也会根据他们的观察来学习 3D 重建，包括时间变化和物理交互。这种类型的监督研究较少，但这将是一个值得探索的有趣话题。

4 2 Human Reconstruction

4 2.1 Body Shape and Pose Reconstruction

人体形状和姿势重建是视觉界长期研究的一个问题，因为它为广泛的现实世界应用开辟了可能性。虽然通过处理来自替代传感器 (alternative sensors)、图像序列 (image sequences) 或多摄像机视图 (multi-camera views) 的信息，在该领域取得了显著的成功，但由于 2D 到 3D 映射的模糊性 (ambiguity)，仅考虑单眼图像 (monocular images) 的应用的进展受到了限制。

可微渲染的最新进展允许通过学习可用于恢复精确 3D 形状的先验知识来解决这种模糊性。我们调查过的大多数方法都采用了统计形状模型被优化的管线，以确保其在图像平面上的投影和关键 2D 图像观测之间的一致性。Loper 等人提出了 SMPL，这是一种基于皮肤顶点的模型 (skinned vertex-based model)，可以捕捉整个人群中人体形状的相关性，以解决整个人体重建问题。该模型是一种流行的统计体型模型 (statistical body shape model)，只需一小组参数，就可以代表自然人体姿势中的各种体型。

Bogo 等人提出了一种从单个图像预测人体姿势和 3D 网格的方法。核心观察结果是，身体关节拥有丰富的信息集，可以用来恢复形状。他们的方法使用 Pischuilin 等人的方法预测 2D 空间中的关节位置。身体形状使用 SMPL 建模，姿势由具有 23 个关节的骨骼装备表示。每个关节对身体各部分之间的相对旋转进行编码。为了恢复 3D 形状和姿势，该方法最小化了一个占五项的目标函数：三个姿势先验，用于惩罚膝盖和肘部的异常弯曲，一个形状先验，用于确保与 SMPL 模型的一致性，以及一个误差项，用于测量 SMPL 关节的投影与 2D 中估计关节的投影之间的距离。Kanazawa 等人在不使用任何 3D 标注的情况下进一步重建人体形状。

Pavlakos 等人（图 7）也处理了体型和姿势的估计问题。他们提出了一种管线，在该管线中，卷积神经网络被训练来预测与预定义关键点相对应的轮廓和热图 (silhouettes and heatmaps)。该信息用于学习姿

势和形状先验，这些先验被馈送到身体网格生成器（基于 SMPL）中，该身体网格生成器被训练为与所学习的分布一致。针对网格到图像投影和 2D 注释之间的一致性，可微分渲染器被进一步优化。Lassner 等人同样使用轮廓进行重建。最近的一些工作试图通过 render-and-compare 管线重建人体形状与衣服，而不使用任何统计形状模型和昂贵的标注。

Fig. 7. Human body estimation trained with silhouette and keypoint supervision by Pavlakos *et al.* [10] .

Deng 等人解决了无监督的三维人体部位分割问题。该方法基于一个模型，该模型通过零件 (parts) 与相机和可变形三角形网格 (deformable triangular meshes) 的关系对预定义数量的部分 (pre-defined number of parts) 进行参数化。神经网络将参数化的部分组合到相同的 3D 空间中，以获得特定姿势的整个对象的 3D 模型。然后，它使用渲染和比较的方法对未知参数进行优化。在推理过程中，除了 3D 模型之外，还可以通过对参数空间进行采样来检索额外的姿势。

4 2.2 Hand Shape and Pose Reconstruction

在精确重建身体其他部位方面也有不少工作，如手形和姿势重建。

Oberweger 等人的工作是从 2D 图像中学习 3D 手部姿势的早期尝试之一。他们的优化管线依赖于手的合成深度渲染，这是通过训练的神经网络实现的。有几个与之不同的工作使用可微渲染来实现这一目的。Baek 等人通过提出一个表示 3D 可变形和关节手网格的参数模型来解决这些挑战。他们训练了一个特征提取器，该提取器结合了手的纹理和形状信息，以及来自彩色图像的姿势估计器。该信息进一步用于通过迭代优化过程回归手部网格的顶点并细化其姿势。作者还使用经过训练的模型通过生成与渲染的 2D mask 和 RGB 图像相对应的 3D 网格对来执行数据扩充。Zhang 等人采取了类似的方法。其中与 2D 关键点相对应的 heatmaps 充当监督信号，用于估计 3D 手模型的参数。定性实验 (Qualitative experiments) 表明，即使在严重闭塞 (occlusions) 的情况下，他们的方法也能够准确地恢复手的形状和姿势。

Zimmermann 等人报告称，手形和姿势重建方法通常在训练的数据集上表现良好，但并不总是推广到其他数据集或现实世界场景中。他们引入了一个多视图手部数据集，并附有 3D 姿势和形状标注，以及一种允许在适度手动干预下进行精确标注的方法。他们的方法使用一组稀疏的 2D 关键点，并通过将可变形的手模型拟合到多个视图上，从多个视图半自动生成分割 mask。拟合过程为每个视图产生 3D 手姿势和形状，用于训练 3D 手姿势估计网络。在推理时，该网络预测未标记数据上的手姿势和相应的置信度得分。这允许人工标注器通过标注最不自信的预测并验证其余预测来节省时间。

4 2.3 Face Reconstruction

考虑到游戏和动画行业的实际应用，其他作品专注于人脸重建。Genova 等人提出了一种无监督的自动编码器架构，该架构从图像像素回归纹理 3D 可变形模型的参数。他们利用面部识别网络的特征，使用真实世界和合成的面部图像，对面部统计形状模型的参数进行回归。可微渲染器用于保持身份之间的特征一致性，以确保批内的参数分布与可变形模型保持一致，并确保网络能够正确解释其输出。

Wu 等人提出了一种完全无监督的方法，用于从人脸图像中学习人脸重建，该方法不依赖于现有的 3D 可变形人脸模型 (3D morphable face models)（图 8）。独立的相机姿态、深度、光照和 albedo 估计网络通过利用阴影信息来学习重建 ground-truth。假设形状的大多数部分是对称的，则使用对称预测网络来减少 3D 几何形状的模糊性。

Fig. 8. Unsupervised 3D face reconstruction by Wu *et al.* [114].

4 2.4 Limitations and Open Problems

所提出的大多数应用都围绕着从单个图像重建 3D 形状展开。对于人体形状重建，可以使用统计模型来学习形状先验来缓解 2D 到 3D 映射的模糊性，形状先验的参数通过可微渲染进行优化。

然而，现有的模型并没有通过包括婴儿来实现年龄的多样化，因为它们是根据成人数据进行训练的。Hesse 等人报告称，由于身体比例不同，为了适应婴儿数据而简单地缩放身体模型并不能产生预期的结果。找到在不同年龄组和身体比例之间进行概括的方法仍然是一个悬而未决的研究问题。现有的体型数据集也要么稀缺，要么偏向于各种体型和/或种族。需要更多的数据种类来改进形状模型，但由于昂贵的 3D 扫描仪、隐私问题以及人类在记录过程中难以正确站立，收集此类数据具有挑战性。

我们调查过的大多数体型重建方法都将统计形状模型拟合到 2D 身体关节位置 (joint locations)，渲染结果并与可用的观测结果进行比较。然而，关节位置不能保证产生最佳结果，应考虑研究替代特征。避免相互渗透 (interpenetration) 是人体重建方法面临的另一个挑战。一些工作通过在目标函数中引入一个惩罚不寻常姿态的误差项来解决这个问题。然而，这种方法并不总是避免相互渗透，因为出于效率的目的，肢体是用几何图元近似的。在渲染过程中加入基于物理的约束来帮助解决这个问题仍然是一个开放的研究领域。

在动画和游戏行业，制作逼真的语音动画是一项具有挑战性的任务。由于演员的嘴唇同步表演需要映射到虚拟角色，因此确定嘴唇的准确位移至关重要。Bhat 等人使用头盔安装的相机来解决这个问题，该相机跟踪面部标记，以找到面部模型的混合形状权重。由于使用标记并不总是一种选择，其他作研究专注于使用 2D 特征、深度或低分辨率图像。最近，基于肌肉的系统已经显示出提高混合形状模型的准确性和语义可解释性的潜力。然而，将这样的模型公式化并集成到基于可微渲染的管线中仍然是一个悬而未决的研究问题。

4 3 3D Adversarial Examples

计算机视觉中的许多机器学习模型容易受到对抗攻击，对抗攻击被定义为具有视觉上不可察觉的扰动的输入，旨在故意触发错误分类。像素扰动由于其简单性而具有吸引力，但没有考虑图像形成的知识。这使得它们不适合真实世界的安全应用程序，因为攻击者无法访问像素级信息。

为了解决这些缺点，最近的研究工作集中在扰动物体几何结构和场景照明参数空间。Liu 等人提出了一种扰动场景光的方法，场景光被表示为球面谐波，以过滤不实际的高频照明，同时允许使用解析导数。另一方面，Zeng 等人扰动场景/对象的内在参数。图像强度的这些变化被限制为在视觉上不可察觉，以攻击 2D 对象分类器。Xiao 等人用纹理扰动对象顶点，并从不同角度进行渲染，以研究网格的脆弱区域。Alcorn 等人通过生成 3D 对象的不受限制的 6D 姿势，并分析深度神经网络如何对这种不寻常的配置做出反应，来研究由分布误差引起的攻击。对抗攻击也可以应用于与图像分类无关的问题。Wu 等人在视觉跟踪的背景下分析了 Siamese networks 的漏洞。作者声称，这些跟踪器对相似性得分 (similarity score)、余弦窗惩罚 (cosine window penalty) 和不对称区域提议网络 (asymmetrical region proposal network) 的过度依赖构成了潜在的对抗性威胁。他们提出了一种可微渲染管线，为 3D 对象生成扰动纹理图，成功地降低了跟踪精度，甚至使跟踪器漂移。

4 3.1 Limitations and Open Problems

由于可用的训练数据有限或偏向搜索空间的某些部分，一些应用程序比其他应用程序更容易受到对抗攻击。例如，Alcorn 等人报告称，ImageNet 分类器只标记了物体 6D 姿态空间的 3.09%，并对许多生成的人类可识别的对抗示例进行了错误分类。因此，使用具有更好照片逼真度重建的可微分渲染来增强训练

数据可能有助于减少搜索空间中的偏差。尽管照片真实感渲染有一定的局限性（如第 5.3 节所述），但使用简单照明模型的可微渲染来探索数据增强的影响是值得的。

4 4 Other Applications

尽管可微分渲染主要被集成到 3D 对象/人的重建和对抗攻击管线中，但有几项工作专注于不同的主题。

Zakharov 等人提出了一种自动标注管线，该管线从预训练的现成 2D 探测器和稀疏激光雷达数据中恢复汽车的 9D 姿态（平移、旋转、缩放）。他们介绍了一种基于 SDF（带符号距离场）和 NOCS（归一化对象坐标空间）的新型可微分形状渲染器。在给定学习的形状先验的情况下，渲染器针对几何和物理参数进行优化。Beker 等人通过利用单目深度估计和纹理 3D 对象重建，进一步提高了精度（不使用激光雷达信息）。图 9 举例说明了这些：

Fig. 9. Automatic generation of 3D object detection labels via rendering by Zakharov *et al.* [46] (upper) and Beker *et al.* [125] (lower).

一些工作专注于从图像中估计光源。Nieto 等人试图在给定场景的 RGB 和深度图像的情况下，将渲染图像和观测图像之间的光度误差 (photometric error) 降至最低。与大多数以前的工作不同，在估计照明时不考虑投射阴影，他们的方法基于 Blinn-Phong 反射模型。

另一方面，Azinovic 等人估计了室内场景中的光源位置和对象材质特性。它们使用 3D 数据和单个或多个 RGB 帧，以及相应的相机姿势，并使用可微蒙特卡洛渲染器来优化未知参数。

可微渲染也可以集成到多视图几何管线中，以推断相机参数和密集表面模型，或合成新视图（图 10）。

Fig. 10. Novel view syntheses by Mildenhall *et al.* [42]. Ground-truth object (first image), cropped ground-truth (second and fourth), and cropped synthesized images (third and fifth).

此外，可微路径跟踪目前用于捕捉和建模人类牙齿，估计着色器参数，重建传感器无法直接看到的对象，联合估计 3D 场景的材质和光线参数，估计姿势，以及分析场景中的光照、阴影和阴影。可微渲染对于通过渲染图像处理 3D 模型也很有用（图 11）。

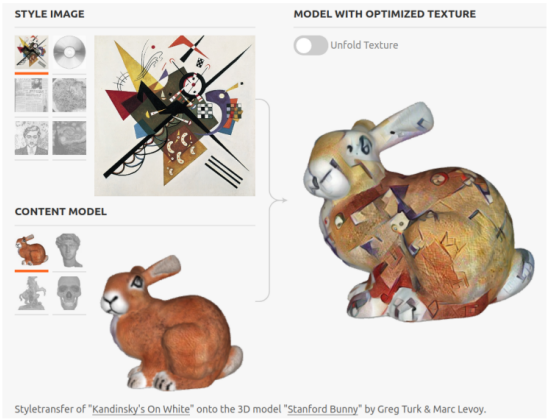

Fig. 11. Style transfer of a 3D model by Mordvintsev *et al.* [126].

4 4.1 Open Problems

使用可微渲染可以改进先前研究的几个问题。示例包括手跟踪 (hand tracking)、制造半透明材料 (fabricating translucent materials)、创建折射焦散 (refractive caustics) 或作为建筑设计过程的一部分的优化日光 (daylight)。通过三维重建进行图像处理是可微渲染的一种新应用。

尽管最初的工作仅限于驾驶场景，但由于人类和物体重建的最新进展，人脸图像操作等其他领域可以通过可微渲染来利用。

TABLE 5
Comparison between existing differentiable rendering libraries

	PyTorch 3D	Kaolin	TensorFlow Graphics	Mitsuba 2
Core implementation	PyTorch/CUDA	PyTorch/CUDA	TensorFlow/C++/GPU	C++/CUDA Python bindings
Supported primitives	Triangle mesh Point cloud	Triangle/quad mesh Point cloud Voxel grid SDF	Triangle mesh Point cloud	Triangle mesh Custom
Differentiable rendering algorithms	Soft Rasterizer (extendable)	NMR Soft Rasterizer DIB-Renderer	Analytical derivatives	Loubet *et al.*
Rendering Method	Rasterization	Rasterization	Rasterization	Ray tracing
3D operations	Graph convolutions 3D transformations Point Cloud Operations (Umeyama, ICP, KNN)	Primitive conversions 3D transformations	Graph convolutions 3D transformations	3D transformations
Shader support	Hard/soft Phong Hard/soft Gouraud Hard flat Soft silhouette	Phong	Phong	Wide range BSDF
Lighting support	Point Directional	Ambient Directional	Point Spherical harmonic	Area Point Spot Constant environment Environment map Directional
Loss functions	Chamfer Distance Mesh edge Laplacian smoothing Normal consistency	Chamfer distance Directed distance Mesh Laplacian	Chamfer distance	Not supported
Camera support	Perspective Ortographic	Perspective Ortographic	Perspective Ortographic Quadratic distortion	Perspective (pinole, thin lens) Irradiance meter Radiance meter
Data I/O support	OBJ PLY	External	External	OBJ PLY
Data support	ShapeNet R2N2	ScanNet ModelNet ShapeNet	Not supported	Not supported
Architectures	Not supported	DIB-Renderer PointNet PointNet++ GResNet 3D-GAN Pixel2Mesh GEOMetrics Occupancy Networks	Not supported	Not supported
Other functionalities	Heterogeneous batches	Model zoo Visualization	TensorBoard (mesh)	Scene file support Extensible by plugin
Version	0.2	0.1.0	1.0.0	2.0.0

五 Libraries

在本节中，我们将介绍现有的可微渲染库。我们还简要介绍了不可微的渲染库，因为它们有着相对悠久的历史，可以指导可微渲染的未来发展。我们还讨论了局限性和悬而未决的问题。我们将可微分渲染库定义为支持单个或多个可微分渲染算法、实现可与渲染层一起使用的多个实用函数、可与现有神经网络框架集成并设计为模块化和可扩展的软件。我们将单个算法的实现排除在库定义的范围之外。

5 1 Differentiable Rendering Libraries

截至 2020 年第二季度，市场上可用的可微渲染库有 TensorFlow Graphics、Kaolin、PyTorch3D 和 Mitsuba 2。我们对表 5 中的库功能进行了比较。由于不断的发展，到本文发表时，功能列表有可能得到扩展。

5 1.1 TensorFlow Graphics

TensorFlow Graphics 由谷歌开发，是该领域的第一个库。它将可微渲染与其范围内的几何层相结合，从而更容易与基于 TensorFlow 的神经网络架构集成。然而，它只支持第 2.1 节中描述的渲染的解析导数，不提供可见性变化的梯度。该库支持表面和材质特性的建模、具有点光源和球谐光照的朗伯曲面的 Phong 反射率模型、图形卷积支持和倒角距离损失 (Chamfer distance loss)。它还允许在 TensorBoard 中轻松地可视化网格。此外，还提供了 Levenberg-Marquardt 算法进行优化。

5 1.2 Kaolin

Kaolin 由 NVIDIA 开发，基于 PyTorch 深度学习框架。除了不同的渲染功能外，该库还包括各种 3D 神经架构的实现，以及对 3D 对象的三角形/四边形网格、点云、体素网格和 SDF 表示的支持。神经架构实现由 model zoo 支持，该 model zoo 为每个实现提供预先训练的权重。Kaolin 保留了现有可微渲染算法的原始实现，并为各种数据集的读取和预处理提供了支持。为了更容易与不同的数据集和体系结构集成，该库允许在各种基元类型之间进行转换。

5 1.3 PyTorch3D

PyTorch3D 由 Facebook 开发，基于 PyTorch 深度学习框架。核心渲染算法及其所有依赖项都通过 CUDA 实现进行了优化，用于反向和前向过程。渲染管线由光栅器和着色器模块组成。光栅器将 3D 相机变换和光栅化应用于给定的形状基本体、网格和点云。输出被传输到着色器组件，该组件应用光照、插值纹理并计算反射率阴影，包括硬/软 Phong、硬/软 Gouraud、硬平面和软轮廓阴影，以创建最终图像。除了核心渲染管线外，PyTorch3D 还支持几个 3D 损失函数，包括倒角距离、网格边缘损失、拉普拉斯正则化损失和正态一致性损失，以及图卷积。作为一项独特的功能，PyTorch3D 可以支持 3D 数据的异构批处理 (heterogeneous batching)，这允许使用不同大小的网格作为渲染器的输入。这样的功能对于用不同级别的细节有效地表示 3D 对象以根据应用找到顶点数量和视觉质量之间的平衡是至关重要的。

5 1.4 Mitsuba 2

Mitsuba 2 提出了一种具有自动区分功能的高效高性能渲染的新方法。它被设计为一种高效的、内部可移植的计算单元，可用于多种用途。出于优化目的，其实时（JIT）编译器支持 GPU 上算法的符号执行。Enoki 库也支持自动微分。Mitsuba 2 证明了其在解决具有挑战性的问题方面的有效性和易用性：马尔可夫链蒙特卡罗（MCMC）渲染技术用于相干地探索附近的光路以确保更快的收敛，这是一种创建梯度索引光学器件 (gradient-index optics) 的方法，该光学器件将入射照明聚焦到焦散中，以及一种用于从多个图像重建异构参与介质的参数的方法。

与其他库不同，Mitsuba 2 主要通过三个指导原则关注速度和效率：无重复 (no duplication)、不冒失 (unobtrusiveness) 和模块化 (modularity)。它执行符号运算，将内核的求值 (evaluation of a kernel) 延迟到变量被访问为止。对于基于自动微分的算法，所有计算的变量和中间值都保存在内存中。特别是对于相

互反射场景，这种方法消耗了大量的内存，这限制了渲染图像的大小和一次可以处理的 3D 对象多边形数量。为了克服这个问题，Mitsuba 2 在每次 JIT 编译之前使用顶点消除定期简化计算图。

除了高效的实现和设计外，Mitsuba 2 还提出了两种新的 MCMC 方案，相干伪边界 (Coherent Pseudo-Marginal)MLT 和多重尝试 (Multiple-Try)Metropolis。相干伪边界 MLT 算法基于对修改的目标函数 π 进行采样的想法，该目标函数 π 是将路径贡献函数 f 与高斯核 G 卷积的结果。这种方法在每次迭代时产生相干光束，并在假设相干光束的物理存储位置彼此接近的情况下提高存储器访问速度。Multiple Try Metropolis（MTM）是之前工作的修改版本，它将 MTM 集成到矢量化的 MTL 中。新的 MTM 算法在每次迭代时生成一组 N 个建议供选择，而不是传统的随机游走。

5 1.5 Comparison

所有库要么通过深度学习框架进行优化，这些框架是为使用 Python 进行 CPU/GPU 不可知编码而建立的，要么通过自定义 C++/CUDA 访问进行优化。Mitsuba 2 是唯一一个基于自定义自动差异化模块 Enoki 的库。它还在 C++ 核心之上提供 Python 绑定，而其他库则受益于 Python 中 PyTorch/TensorFlow 库的自动区分功能。

Tensorflow Graphics 和 Kaolin 旨在通过保留原始实现，将新算法作为第三方集成到其结构中。这种面向集合的设计选择减少了集成新算法所需的工作，并允许不同算法之间的容易比较。然而，由于每种算法的渲染管道及其优化级别并不集中，新算法的学习曲线、多种算法之间的集成以及渲染速度都很容易发生显著变化。另一方面，PyTorch3D 和 Mitsuba 2 旨在提供一个可定制的管线，用户可以在其中扩展其每个子模块的功能。这种面向定制的设计选择允许使用可扩展的组件集成和开发新的算法，用于照明、着色、纹理和混合。虽然这样的设计选择允许用户灵活地操作渲染器组件并有效地实验新的研究想法，但如果第三方算法作为独立实现发布，则需要额外的努力来集成它们。

除了每个库的渲染功能外，Kaolin 的即用的最先进的体系结构实现，以及预训练的 model zoo，降低了重新实现和重新训练的成本。此外，与其他替代方案相比，具有数据集加载和预处理功能是一个强大的优势。另一方面，PyTorch3D 和 Mitsuba 2 中对 OBJ 和 PLY 格式的本地支持减少了 I/O 操作对其他库的依赖。TensorBoard 和 TensorFlow Graphics 之间的集成使用户可以轻松地可视化渲染输出。另一方面，Kaolin 通过 pptk 后端提供了几个可视化网格、体素和点云的功能。

除了 Mitsuba 2，每个库都专注于光栅化，而不是光线跟踪。这使得它们可以很容易地用于合成数据集和简单场景，而无需复杂的反射率。考虑到支持的着色算法和表面材质的数量，到目前为止，现有库在现实世界中的场景应用受到了限制。

5 2 Non-Differentiable Rendering

在过去的几十年里，已经开发了几个基于不可微分光栅化器和基于光线跟踪的渲染库。对于渲染速度比视觉质量更重要的应用程序，基于光栅化的库是首选。

它们通常依赖 Direct3D、OpenGL 或 Vulkan API 作为后端，而不是提供自定义实现。这些 API 支持通用功能，如方便的 GPU 访问、随时可用的光栅化管道和 z 缓冲区算法。他们实现了自己的着色器语言，以允许开发人员扩展他们的功能。然而，由于直接使用这些 API 会产生巨大的开发成本，因此使用高级包装器 Unity 或虚幻引擎也是很常见的。游戏引擎提供了额外的实用功能，并允许设计界面。这使得渲染软件在多个硬件设备之间的可移植性更容易，这些硬件设备的直接访问 API 可能会有显著差异。它还将开发人员的知识需求减少到硬件特定的内容。高级库的主要优势是易于集成复杂的灯光和阴影、动画功能、用于保存/加载各种资产的高级 I/O 系统以及用于降低研发成本的场景信息。引擎还支持多种编程语言进行开发，这有助于减少编写生产级代码所需的学习曲线。最后，它们提供了易于交互和可视化的 GUI。这加快了开发过程，并通过可视化中间结果帮助开发人员尽早发现可能的错误。

光线跟踪的照片逼真度渲染比光栅化消耗更多的资源。因此，对于视觉质量比渲染速度更重要的应用程序，基于光线跟踪的库是首选。Arnold，V-Ray 和 RenderMan 是流行的基于光线跟踪的库。

它们集成到数字内容创建（DCC）工具中，如 Maya 和 3ds Max，以提供最终用户 GUI 访问。对于材质和着色器，RenderMan 提供了自己的着色器语言 RSL，作为通过 GUI 编辑功能的扩展。除了网格之

外，这些库还支持体素和参数曲面作为形状基本体。与光栅化类似，工业光线跟踪库基于几种低级 API，如 Embree、OSPRay、OptiX 和 Iray。Embre 和 OSPRay 通过使用 SIMD 指令针对英特尔 CPU 进行了优化。虽然 Embree 提供了光线跟踪加速内核，但 OSPRay 包含额外的功能，如体渲染、全局光照、更容易的行业采用、分布式计算和对多个形状基元的支持。另一方面，基于 GPU 的库 OptiX 和 Iray 也遵循相同的结构。虽然 OptiX 提供核心内核功能，但 Iray 增加了 AI 去噪、RTX 硬件支持、多 GPU 设置和更容易的第三方集成等额外功能。

5 3 Limitations and Open Problems

尽管出现了可微分渲染库的发展，但仍有局限性和可能的发展方向需要解决。我们在下面列出了其中的一些：

对嵌入式环境的支持是有限的。优化算法对于使其能够在硬件资源和处理能力有限的嵌入式环境中运行至关重要。对于 NVIDIA GPU，TensorRT 已被开发用于优化经过训练的网络的推理速度。然而，用于优化嵌入式环境的可微渲染算法的类似功能仍然有限。

当前的库没有为扩展提供标准接口。当前的可微渲染库要么需要从头开始实现新的算法，要么通过基于插件的架构提供基本的扩展层。提供跨框架通用的扩展功能，类似于不可微分的渲染社区，将是一种附加值。

现有的实现具有有限的功能。可微渲染库主要是为计算机视觉研究人员和工程师设计的。目前缺乏不可微的功能，如与 DCC 工具的集成、对三角形以外的基元的支持、各种输出格式，如高动态范围图像、元数据/动画渲染。我们相信，增加游戏和电影行业所需的更多功能将有助于开发新型应用程序。此外，现有的库（Kaolin 除外）并没有为在可微渲染研究中常用的数据集提供现成的支持。这反过来又导致数据集管理的重复实施，同时增加了开发成本。

对光和材质的支持有限。由于光线和表面材质质量对于照片级真实感 RGB 渲染至关重要，因此对光栅化和基于光线跟踪的库的高级光线支持的开发是有限的，并且不是统一的。光模型的这种限制阻碍了渲染的 RGB 图像的逼真外观，并因此阻碍了监督信号的质量。

有限的调试和基准测试支持。调试工具和基准测试支持对于高效的开发和研究至关重要。现有库受益于用于基准测试的特定语言和/或第三方调试工具（如果有的话）。专门为可微分渲染设计的基准测试和调试工具尚未开发，该工具旨在检测渲染和梯度计算的不准确性。

目前不可能在不同的库之间共享模型。支持跨多个深度学习库的模型共享是另一个限制。最近，ONNX format 已经被开发出来，以便轻松共享经过训练的模型。然而，由于不同库之间的实现差异，仍然不可能导出包含可微分渲染层的神经网络模型，并将其用于不同的库中，以便通过 ONNX 接口进行推理。

六 Conclusion

本文概述了可微渲染的研究现状——流行的算法和数据表示技术，以及评估指标和常用实践。它还介绍了使用基于 DR 的技术的应用，涵盖了广泛的领域，如 3D 对象/身体形状和姿势重建、对抗攻击、自动标记或光源估计。也介绍并比较了几种常用于开发新的基于可微渲染的方法的库。最后，讨论了算法、应用程序和库的开放性问题。

尽管可微渲染是一个新领域，但在旨在简化其使用的新工具的不断开发的帮助下，它正在迅速成熟。这将使更多的研究人员能够开发新的神经网络模型，从 2D 图像数据中理解 3D 信息。因此，可以提高各种应用程序的性能，同时可以减少对 3D 数据收集和注释的需求。

一旦成熟到可以部署到嵌入式设备上，我们可能会开始看到基于可微渲染的方法来解决实时约束的问题（如自动驾驶）。我们生活的时代令人兴奋，深度神经网络的进步帮助我们解决日常问题，最好的还在以后。

参考文献

[1] Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., & Gaidon, A. (2020). Differentiable rendering: A survey. arXiv preprint arXiv:2006.12057.

[2] https://towardsdatascience.com/differentiable-rendering-d00a4b0f14be

[3] https://blog.qarnot.com/an-overview-of-differentiable-rendering/

[4] https://www.zhihu.com/question/364770565/answer/1266067986

[5] https://www.youtube.com/watch?v=7LU0KcnSTc4

[6] https://www.youtube.com/watch?v=Tou8or1ed6E

[7] https://zhuanlan.zhihu.com/p/372338398

[8] https://zhuanlan.zhihu.com/p/556048189

[9] https://zhuanlan.zhihu.com/p/582896865