中国科学 信息科学
SCIENTIA SINICA Informationis

From traditional rendering to differentiable rendering: theories, methods and applications

叶子鹏, 夏雯宇, 孙志尧, 易冉, 余?F婧 and 刘永进

Citation: 中国科学: 信息科学 **51**, 1043 (2021); doi: 10.1360/SSI-2020-0272

View online: https://engine.scichina.com/doi/10.1360/SSI-2020-0272

View Table of Contents: https://engine.scichina.com/publisher/scp/journal/SSI/51/7

Published by the 《中国科学》杂志社

中国科学: 信息科学 2021 年 第 51 卷 第 7 期: 1043–1067
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
评述

从传统渲染到可微渲染: 基本原理、方法和应用

叶子鹏[1], 夏雯宇[1], 孙志尧[1], 易冉[1], 余旻婧[2], 刘永进[1*]

1. 清华大学计算机科学与技术系, 北京 100084
2. 天津大学智能与计算学部, 天津 300350
* 通信作者. E-mail: liuyongjin@tsinghua.edu.cn

收稿日期: 2020–08–30; 接受日期: 2020–09–29; 网络出版日期: 2021–06–16

国家自然科学基金 (批准号: 62002258, 61725204) 资助项目

摘要 近年来随着图形硬件的快速发展, 渲染技术和深度学习技术都飞速发展, 可微渲染作为二者之间的桥梁受到了广泛关注. 随着许多可微渲染方法的提出, 逆渲染等借助可微渲染工具的应用也随之蓬勃发展. 本文从传统渲染管线开始介绍, 逐步引入可微渲染的主要思想、基本原理和方法, 对它们进行介绍、分析和比较. 并介绍基于路径跟踪的可微渲染, 随后列出开源的可微渲染工具供大家参考并进行比较. 本文后半部分介绍可微渲染的广泛应用, 分为人脸、人体、人手和物体 4 个方面. 最后列举了一些可微渲染可能的发展方向.

关键词 可微渲染, 逆渲染, 三维重建, 人脸重建, 渲染

1 引言

渲染是从三维场景得到二维图像的过程, 如图 1 所示, 它在动画、游戏、电影和虚拟现实中有丰富的应用. 传统的渲染理论和方法已经趋于成熟, 在工业上已经有许多成熟的软件, 比如 OpenGL, 3ds Max 等. 在游戏行业, 随着图形处理器 (graphics processing unit, GPU) 的发展, 已经能够实时渲染高质量高真实感的画面. 在动画和电影行业, 基于物理的高真实感渲染能够达到以假乱真的效果. 在计算机图形学中, 一般用三角网格表示三维模型, 所以一般的渲染方法都是针对三角网格模型提出的. 除了一般的渲染方法之外, 还有许多专门针对某些特殊应用的特殊渲染方法, 比如对体素 [1]、点云 [2]、毛发 [3] 和流体 [4] 等的渲染. 本文只讨论针对三角网格模型的渲染及其可微渲染, 所以渲染在本文中特指三角网格模型的渲染.

传统的渲染模型分为局部光照模型和全局光照模型, 局部光照模型只考虑光源到物体表面的照射效果, 全局光照模型不仅考虑光源到物体的照射效果, 还考虑物体之间的相互影响. 最简单的局部光照模型是兰伯特 (Lambert) 模型, 它只考虑环境光和漫反射, 并假设物体朝着各个方向均匀地漫反射.

引用格式: 叶子鹏, 夏雯宇, 孙志尧, 等. 从传统渲染到可微渲染: 基本原理、方法和应用. 中国科学: 信息科学, 2021, 51: 1043–1067, doi: 10.1360/SSI-2020-0272
Ye Z P, Xia W Y, Sun Z Y, et al. From traditional rendering to differentiable rendering: theories, methods and applications (in Chinese). Sci Sin Inform, 2021, 51: 1043–1067, doi: 10.1360/SSI-2020-0272

图 1 (网络版彩图) 几个渲染的例子, 包括汽车、房子、游戏场景和人脸

Figure 1 (Color online) Some rendering examples, including a car, a house, a game scene and a human face

Phong 模型是影响深远的经典局部光照模型, 它假设环境光是常量, 不考虑物体之间的相互反射, 只考虑环境光、漫反射和镜面反射, 许多复杂的光照模型都是基于 Phong 模型改编而来的. 全局光照模型一般是基于物理的方法, 使用物理世界的规律来模拟计算光在各处的传播和强度, 经典的算法有辐射度法、光线跟踪和光子映射. 这些全局光照模型的结果有非常高的真实感, 但时间花费高. 为了实时渲染出高真实感的结果, 许多方法使用近似的全局光照, 比如使用预处理得到的环境光照, 为了适应实时变化的光照, 一般使用球谐函数作为基底, 根据光照的变化来组合.

与渲染相反的过程叫做逆渲染, 该过程从图片中获得物体的形状与材质信息、场景中的光照信息及相机参数等. 逆渲染是个不适定问题, 比正向渲染难度大得多, 它不仅涉及到计算机图形学中的许多问题, 也涉及到计算机视觉的很多问题. 为了正则化这个不适定问题, 一般会使用逆渲染对象的先验知识, 比如对于人体组成部分 (人脸 [5~17]、人体 [18~27]、人手 [28~31]) 或是规则的人造物体 [12,19,32~44]. 对于某些特别的任务, 对应有专门的研究方向, 比如逆渲染人脸的研究方向称为人脸重建. 近年来, 为了更好地解决逆渲染问题, 同时减少对训练数据的依赖, 许多可微渲染 (differentiable rendering) 的方法被提出 [9,18,19,32,45~49]. 可微渲染是一个可以微分求导的渲染过程, 它的正向是渲染, 逆向是求像素对场景参数的微分. 由于传统的渲染不可微, 难以设计基于优化和基于深度学习神经网络的逆渲染方法, 可微渲染技术的提出大大地增加了这些方法的设计空间.

可微渲染是近年来的研究热点, 已经建立了相应的理论基础, 并且它的应用也日益广泛. 本文从传统的抽象渲染管线 (第 2 节) 开始介绍, 逐步分析其中每个步骤, 然后介绍可微渲染的基本原理和主流方法 (第 3~5 节) 以及开源的可微渲染工具 (第 6 节), 之后介绍可微渲染的应用 (第 7 节), 最后介绍可微渲染未来可能的发展方向 (第 8 节). 最近已有关于可微渲染的综述论文 [50], 和文献 [50] 不同, 本文旨在让读者回顾传统渲染管线, 快速地理解可微渲染的思想, 并可以迅速判断可微渲染是否适合自己当前的工作, 找到最适合的可微渲染方法以及开源工具.

2 传统渲染管线

传统渲染管线被抽象为几个固定的步骤, 每个步骤有着固定且单一的任务. 我们以 OpenGL 使用的可编程渲染管线 (本文中简称为渲染管线) 为例介绍传统渲染管线, 它使用的是局部光照模型. 如图 2 所示, OpenGL 的渲染管线依次分为顶点着色器、图元装配、几何着色器、光栅化、片段着色器、测试与混合 6 个步骤. 在三角网格模型渲染过程中, 几何着色器仅输出三角片元, 假设不开启深度测试和混合, 一个简单的渲染管线如图 3 所示, 它主要由顶点着色器、光栅化和片段着色器 3 个部分组成.

顶点着色器的输入是模型的顶点属性 (包括局部坐标系的坐标、法向、材质等) 和全局的信息 (相机参数、光照等), 输出是模型经过变换之后的属性 (全局裁剪空间的坐标等). 顶点着色器负责对顶点

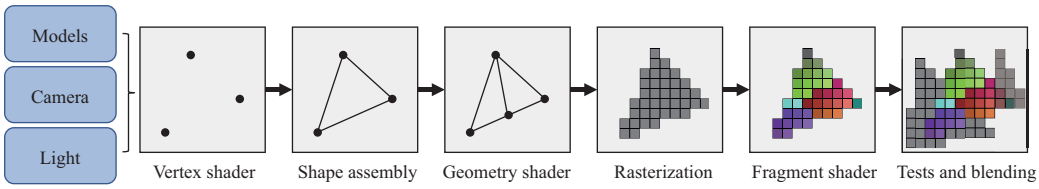

图 2 (网络版彩图) **OpenGL 的渲染管线依次分为: 顶点着色器、图元装配、几何着色器、光栅化、片段着色器、测试与混合**

Figure 2 (Color online) OpenGL pipeline: vertex shader, shape assembly, geometry shader, rasterization, fragment shader, tests and blending

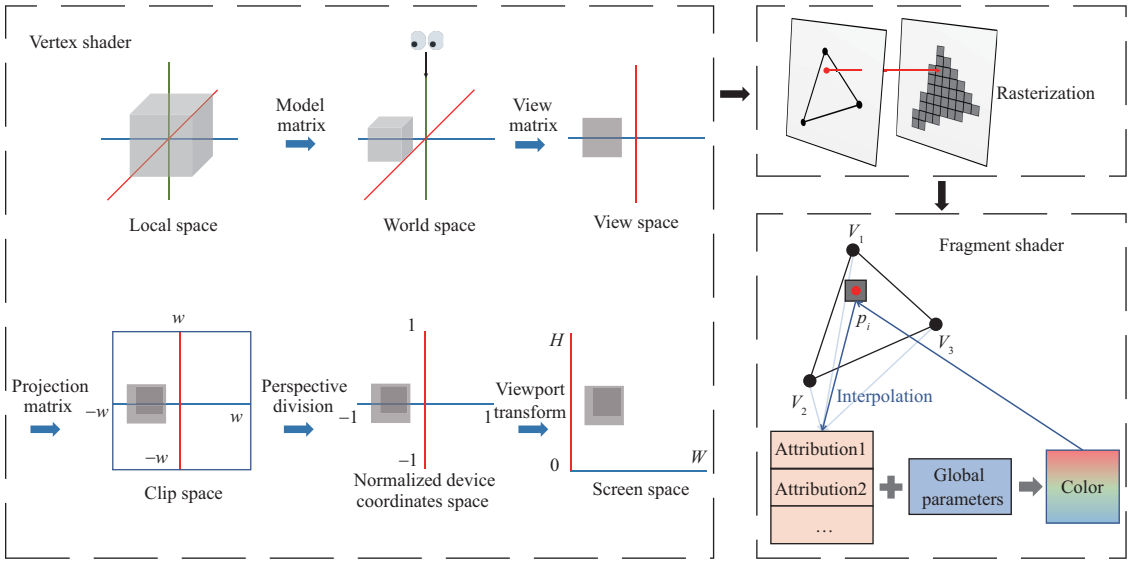

图 3 顶点着色器负责对顶点进行坐标变换, 其中模型矩阵、视图矩阵和投影矩阵需要根据场景计算得到, 透视除法和视口变换会根据窗口大小自动处理. 光栅化负责计算每个图元 (对于三角网格是三角形) 相关的像素. 片段着色器负责对每个像素的颜色进行赋值, 首先从像素的重心坐标通过插值得到每个像素的属性, 结合全局参数, 比如光照计算出像素的颜色

Figure 3 The vertex shader does coordinate transformation for vertices, where modeling matrix, viewing matrix and projection matrix are calculated from the scene, and perspective division and viewport mapping are processed according to the size of the window. Rasterization calculates every primitive's related pixels (for a triangular mesh the related area will be a triangle). The fragment shader assigns value to every pixel by interpolating barycentric coordinates to obtain a pixel's attributes, combining with global parameters such as lighting

进行坐标变换, 输入的模型数据的顶点在局部空间, 需要乘以模型矩阵、视图矩阵和投影矩阵再进行透视除法和视口变换最终到屏幕空间. 其中模型矩阵、视图矩阵和投影矩阵需要根据场景计算得到, 透视除法和视口变换和场景无关, 会根据窗口大小自动处理. 模型顶点经过这些变换后, 从局部空间, 依次到世界空间、观察空间、剪裁空间、标准化设备坐标系空间和屏幕空间. 顶点变换的过程包括平移、缩放、旋转和投影, 均为线性变换, 天然可微, 所以可微渲染的方法一般会使用和上述完全相同的顶点变换的方法.

图元装配是把顶点装配成不同的几何结构 (比如三角形、折线段等), 一般我们输入的模型是三角网格, 所以这一步把顶点装配成一系列三角形. 几何着色器的输入是一个图元及其顶点, 输出是一个或多个图元, 一般用来做细节层次 (levels of detail).

光栅化是从图元转化为片段的过程, 也就是从几何数据转化为光栅显示的像素的过程, 它计算与每个图元 (对于三角网格是三角形) 相关的像素. 由于它是从连续空间到离散空间的一步转化, 所以会带来许多问题, 在传统渲染技术中一个常见的问题就是反走样 (也称为抗锯齿), 可微渲染的难点也在光栅化这一步骤, 但和反走样不同的是, 困扰着我们的是另一个问题: 如何从像素对顶点求导. 所有的可微渲染研究都是围绕着这个问题进行的, 我们将在第 4 节详细介绍这个问题.

片段着色器负责计算出片段的颜色, 它使用一系列顶点属性和全局的参数. 我们首先找到片段所在的三角形并计算片段中心的重心坐标, 通过重心插值把顶点的属性赋给每个片段. 为了实现不同的效果, 不同的着色器可能使用不同的算法来计算片段的颜色, 片段着色器是一个着色器程序的核心, 由用户定义且可能非常复杂, 但可以将其计算过程抽象成一个黑盒, 输入是插值得到的属性和全局参数, 输出是片段的颜色. 比如纹理贴图的过程, 使用插值得到的纹理坐标和全局的纹理图片, 通过采样得到片段的颜色; 比如 Phong 模型, 使用插值得到的法向和材质以及全局的光照参数, 计算出每个片段的漫反射成分和镜面反射成分, 并和常量环境光叠加得到片段的颜色; 比如投影归一化坐标码 (projected normalized coordinate code, PNCC) [51], 它利用一张图片的三通道颜色来表示三维模型的三维坐标, 我们可以通过片段着色器直接输出该片段插值得到的三维坐标来得到. 由于片段着色器可以非常复杂, 可能包含可微的过程和不可微的过程, 所以可微渲染的方法一般会对这个步骤加以限制, 只采用可微的片段着色器. 因为许多常用的模型都是天然可微的, 只要稍加限制, 这个步骤往往无需修改即可应用到可微渲染中.

测试与混合是处理物体之间的遮挡关系以及透明物体和它背后其他物体如何显示的问题, 本质上都是处理物体重叠时产生的问题, 一般采用 Z-buffer 算法来解决. Z-buffer 算法对每个像素维护一个深度信息的缓冲区, 每次判断新片段的深度和缓冲区内的深度, 保留更靠前的片段. 由于遮挡关系对顶点坐标敏感, 但 Z-buffer 算法本身不可微, 所以这一步也为渲染的可微带来了一个障碍. 为了解决这个问题, 需要对这一步骤进行修改, 我们将在第 4 节详细介绍这个问题.

3 可微渲染的基本原理

在介绍可微渲染的方法之前, 我们首先简要地介绍可微渲染的基本原理. 可微渲染是可以微分求导的渲染过程, 分为正向和逆向的过程, 正向过程和传统渲染相同, 输入模型和参数得到一张图片, 逆向是像素对场景参数求导数, 可微渲染需要兼具这两个过程, 不仅需要得到渲染结果, 还要得到渲染结果对输入的导数. 可微渲染不能离开传统的渲染模型, 但传统的渲染方法不可微, 所以可微渲染往往是基于某种传统渲染模型, 通过引入新的技术, 使得我们可以得到渲染结果对输入的导数. 主流的可微渲染方法往往基于以下两类思想, 一类是使用近似的方法, 求得近似导数用于反向传播; 另一类是改编传统渲染模型, 让像素对顶点可导.

上述第 1 类方法不改变传统渲染的正向过程, 虽然传统渲染方法天然不可微, 但使用近似的方法, 能求得近似导数用于反向传播. 这类方法的核心在于如何更好地近似渲染过程的导数, 使得导数在某种观点下是一种有效的近似, 或是使得导数对优化输入有着指导意义. 为了使得导数对优化输入有着指导意义, 有时导数会和应用相关, 会根据损失函数的不同而不同.

对于上述第 2 类方法, 由于传统渲染方法天然不可微, 需要对其进行改编, 使得改编后的渲染方法依然拥有渲染的能力, 渲染结果不发生较大变化, 但其过程完全可微, 可以求得精确导数. 这类方法通常改编其中光栅化的步骤, 因为这一步从连续空间映射到离散空间, 是导致传统渲染不可微的原因.

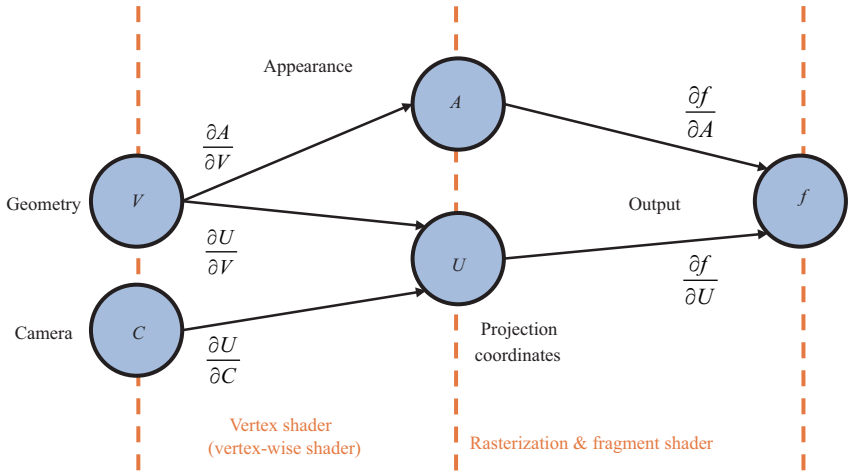

图 4 (网络版彩图) OpenDR 的流程图. 为了求得 f 对 V, A, C 的导数, 根据链式法则, 我们总共需要求 $\frac{\partial f}{\partial A}, \frac{\partial f}{\partial U}, \frac{\partial U}{\partial V}, \frac{\partial U}{\partial C}, \frac{\partial A}{\partial V}$ 五组偏导数

Figure 4 (Color online) Workflow of OpenDR. According to chain rule, 5 partial derivatives $\frac{\partial f}{\partial A}, \frac{\partial f}{\partial U}, \frac{\partial U}{\partial V}, \frac{\partial U}{\partial C}, \frac{\partial A}{\partial V}$ need to be calculated to obtain f's derivatives of V, A, C

4 基于局部光照模型的可微渲染

最近, 许多工作使用近似的方法, 求得近似导数用于反向传播 [9,18,32], 或是改编了传统渲染管线中的步骤 (一般是光栅化和测试与混合步骤), 让像素对顶点可导 [19,45~47]. 本节我们会详细介绍几种有代表性的可微渲染的方法及其理论, 从近似传统渲染方法的导数逐渐过渡到改编渲染方法, 然后简要介绍其他可微渲染的方法, 并将这些可微渲染方法进行对比分析. 它们都使用局部光照模型, 和 OpenGL 的渲染管线有着类似的步骤. 我们称渲染的过程为正向过程, 求导的过程为反向过程.

4.1 局部近似导数的方法

OpenDR [18] 提出了一种对渲染过程求近似导数的方法, 求得近似导数用于反向传播. 它对渲染过程进行了一定的限制, 传统的渲染管线中, 片段着色器是使用用户定义的函数逐片段地确定颜色, 而 OpenDR 要求颜色是逐顶点确定的, 顶点的颜色在光栅化之前已经计算完成, 之后片段着色器只是对顶点颜色进行插值得到片段的颜色. 虽然它限定了计算颜色的过程, 但无需修改光栅化的过程. 假设渲染的输入为模型的几何信息 (V), 模型的外表信息 (A) 和相机参数 (C), 其中模型的外表信息可能受到几何信息的影响, 如图 4 所示, 渲染过程为 $f(V, A, C)$, 正向过程从模型的数据和相机参数得到图片, 逆向过程分别求出 f 对 V, A, C 的导数. 设 U 为模型顶点投影到屏幕上的坐标, 可以理解为顶点着色器的输出, 它可以由模型的几何信息和相机参数计算得到.

如图 4 所示, 为了求得 f 对 V, A, C 的导数, 由链式法则, 我们总共需要求 $\frac{\partial f}{\partial A}, \frac{\partial f}{\partial U}, \frac{\partial U}{\partial V}, \frac{\partial U}{\partial C}, \frac{\partial A}{\partial V}$ 五组偏导数. 由于其中 $\frac{\partial U}{\partial V}$ 和 $\frac{\partial U}{\partial C}$ 是顶点变换, 天然可微. 对于 $\frac{\partial f}{\partial A}$, 由于 OpenDR 对顶点颜色进行插值得到片段的颜色, 所以像素对顶点颜色的导数就是该像素对应的可见片段在其三角形的重心坐标, 我们只需在正向过程中保存每个片段的重心坐标以及每个像素对应的可见片段即可. $\frac{\partial A}{\partial V}$ 是对用户定义的函数求导, 它可能是常数也可能是复杂的函数, 这取决于用户如何定义计算颜色的函数, 只要用户定义的函数可微, 那么它的导数存在. 通常用户定义的计算颜色的函数都是初等函数, 导数很容易计算. 下面我们详细介绍对 $\frac{\partial f}{\partial U}$ 的计算.

由于从 U 到 f 需要经过光栅化和测试与混合, 所以 $\frac{\partial f}{\partial U}$ 是天然不可微的, 而 OpenDR 不改变光栅化和测试与混合的步骤, 而是求近似导数. OpenDR 根据像素和遮挡边界的关系, 把像素分为内部像素和边界像素, 边界像素包含遮挡边界, 内部像素不包含遮挡边界. 遮挡边界是一条边在屏幕上的投影, 且其两侧对应的曲面发生了剧烈的变化, 这往往分为两种情况: 一种是在深度测试的过程中发生了突变 (由遮挡产生), 另一种是这条边连接了两个法向相反的三角形 (由轮廓产生). OpenDR 具体又根据像素跨越遮挡边界的数目, 将它们分为三类: 内部像素、单边界像素和多边界像素, 对这三类像素分别讨论, 采用不同的方法近似. 其中内部像素的范围内不包括遮挡边界; 单边界像素仅和一个遮挡边界相交; 多边界像素和超过一个遮挡边界相交. OpenDR 采用图像空间的一阶泰勒 (Taylor) 展开的方法来近似颜色对水平方向的导数, 考虑模型的局部往右移动了一个像素的距离, 那么每个相关的像素都被它左边的像素替代. 我们可以用相邻像素之差来近似导数, 以水平方向为例, 比如用 $\frac{1}{2}[-1,0,1]$ 来滤波, 这个过程和对图像做一次 Sobel 滤波非常类似, 实现的时候也可以用类似的方法来实现.

至此, 我们已经求出或近似求出了所需的五组导数, 梯度流得以从输出传递回输入. OpenDR 采用的近似导数的方法不改变渲染过程, 使用像素之间的离散导数来近似顶点坐标的变化对像素产生的影响. 我们发现, 这五组导数中最复杂的是 $\frac{\partial f}{\partial U}$, 因为光栅化天然不可微, 导致梯度流无法从像素传播到顶点坐标. 之后介绍的方法, 它们都是围绕光栅化这个步骤进行近似或改编的, 从而使梯度流能够从像素传播到顶点坐标.

4.2 用平滑的光栅化近似导数

如图 5 所示, 标准的光栅化像素对顶点坐标没有梯度流 (导数几乎处处为 0), 为了使得像素对顶点坐标的梯度流存在, 一种平滑的光栅化方法 [32] 被提出, 它不改变正向渲染的过程, 而是在反向传播梯度流时, 通过把顶点移动对像素的影响进行平滑, 得到一种近似像素对顶点坐标导数的方法, 下面将详细介绍这种平滑的光栅化方法.

假设我们要得到某个像素 P_j 的颜色 I_j 对某个顶点 v_i 的屏幕空间的坐标 (x_i, y_i) 的导数, 这里只讨论水平方向的导数, 对于垂直方向的微分完全类似. 我们考虑顶点 v_i 在沿着水平方向移动时, 该像素颜色的变化, 对于标准的光栅化, 如图 5 第 2 行所示, 随着顶点位置的变化, 像素的颜色在某个临界位置发生了突变. 如果三角形的颜色与顶点位置 v_i 无关, 那么 $\frac{\partial I_j}{\partial v_i}$ 几乎处处为 0, 损失函数得到的误差无法从像素的颜色传播到顶点的坐标. 梯度流无法传播的根本原因在于, 颜色没有随着顶点坐标平滑地变化, 而是所有的变化都集中在发生突变的地方. 基于这一点, 平滑的光栅化相对标准的光栅化, 在顶点移动时像素发生突变的地方进行平滑, 如图 5 第 4 行所示, 在初始位置和临界点之间使用了线性插值. 下面分两类情况给出详细定义, 一类是像素初始在三角形外, 另一类是像素初始在三角形内.

当像素初始在三角形外时, 如图 5 左半部分所示. 我们记 v_i 的初始位置为 x_0, 此时 P_i 的颜色为 $I(x_0)$, 当 v_i 向右移动到 x_1 时, 三角形的边和像素接触, 此时 P_i 的颜色为 $I(x_1)$, 我们记 $\delta_i^x = x_1 - x_0$, $\delta_j^I = I(x_1) - I(x_0)$, 我们将突变改为渐变, 也就是在 x_0 和 x_1 之间用 $\frac{\delta_j^I}{\delta_i^x}$ 来近似 $\frac{\partial I_j}{\partial v_i}$. 由于 x_0 是一个拐点, 在 x_0 的导数的定义需要特别定义, 其实我们最关心的正是导数在 x_0 处的定义. 为了让梯度流更好地传播, 在 x_0 的导数如下定义:

$$\left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0} = \begin{cases} \dfrac{\delta_j^I}{\delta_i^x}, \delta_j^P \delta_j^I < 0, \\ 0, \quad \delta_j^P \delta_j^I \geqslant 0, \end{cases} \tag{1}$$

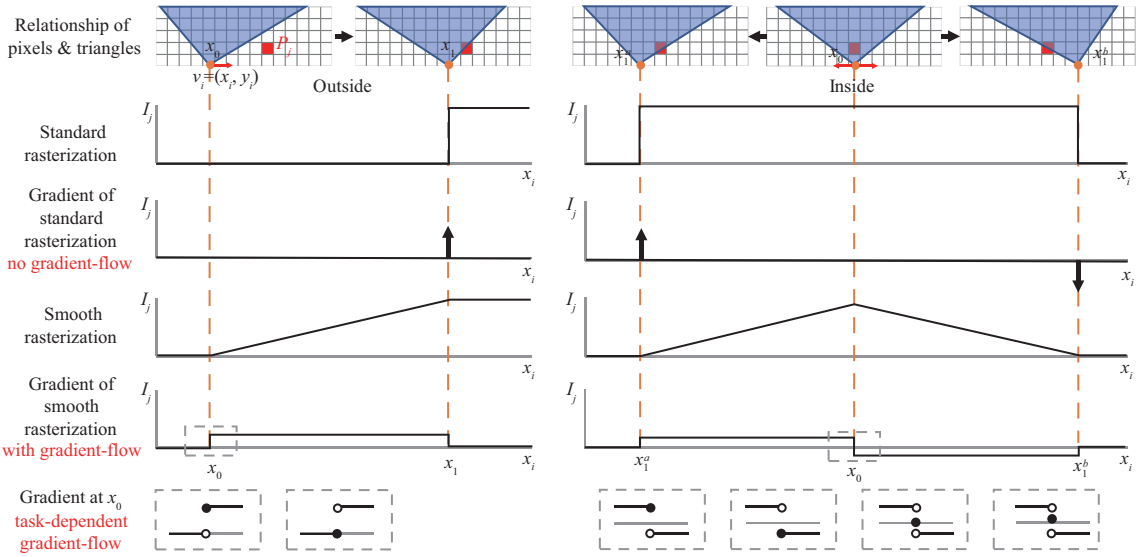

图 5 (网络版彩图) 在梯度反向传播时, 如果三角形的颜色和被求导的顶点位置无关, 标准的光栅化对顶点坐标没有梯度流 (导数几乎处处为 0), 为了使得梯度流存在, 一种通过平滑的光栅化得到近似梯度的方法被提出, 它的导数在顶点当前位置 x_0 附近不恒为 0, 在 x_0 处的导数根据目标函数返回的梯度流分情况讨论, 可以反向传播. 左半部分是像素初始在三角形外的情况, 右半部分是像素初始在三角形内的情况

Figure 5 (Color online) During back-propagation, if a triangle's color is unrelated to the vertex position being differentiated, standard rasterization passes no gradient flow to the vertex (gradient is almost constantly 0). In order to maintain a gradient flow, a smooth approximation for rasterization is proposed. Its gradient around the vertex's current position x_0 is decided by gradients passed back by the target function instead of constant 0, which allows back-propagation. The left part demonstrates smooth rasterization when the pixel is originally outside the triangle, while the right part shows when it is inside

其中 δ_j^P 为损失函数反向传播的梯度信号, 它表示该像素是否过量或过暗. 梯度信号 $\delta_j^P > 0$ 表示, 为了优化损失函数像素需要更暗. 如果顶点向右移动, 像素会变得更亮, 即 $\delta_j^I > 0$, 但为了优化损失函数像素需要更暗, 即 $\delta_j^P > 0$, 此时向左向右移动均无效, 所以在这种情况下, 导数定义为 0.

当像素初始在三角形内时, 如图 5 右半部分所示. 我们记 v_i 的初始位置为 x_0, 此时 P_i 的颜色为 $I(x_0)$, 当 v_i 向左移动到 x_1^a 时, 三角形的边和像素接触, 此时 P_i 的颜色为 $I(x_1^a)$, 我们记 $\delta_x^a = x_1^a - x_0$, $\delta_j^{I^a} = I(x_1^a) - I(x_0)$, 当 v_i 向右移动到 x_1^b 时, 三角形的边和像素接触, 此时 P_i 的颜色为 $I(x_1^b)$, 我们记 $\delta_x^b = x_1^b - x_0$, $\delta_j^{I^b} = I(x_1^b) - I(x_0)$. 类似像素初始在三角形外时的情形, 在 x_0 和 x_1^a 之间用 $\frac{\delta_j^{I^a}}{\delta_x^a}$, 在 x_0 和 x_1^b 之间用 $\frac{\delta_j^{I^b}}{\delta_x^b}$, 来近似 $\frac{\partial I_j}{\partial v_i}$. 由于 x_0 为拐点, 在 x_0 的导数的定义需要特别定义. 为了让梯度流更好地传播, 在 x_0 的导数如下定义:

$$\left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0} = \left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0}^{a} + \left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0}^{b}, \tag{2}$$

其中 $\left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0}^{a}$ 和 $\left.\frac{\partial I_j}{\partial x_i}\right|_{x_i=x_0}^{b}$ 的定义和式 (1) 类似, 都是根据损失函数反向传播的梯度信号 δ_j^P 来定义的, 分别判断顶点向左和向右移动是否有利于损失函数降低. 如图 5 第 6 行右半部分所示, 总共分为 4 种情况.

上面讨论了对于单个三角形, 如何进行光栅化以及对光栅化求导. 对于多个三角形, 绘制的时候和传统渲染管线一样, 需要经过测试与混合, 每个像素的颜色由深度最小的三角形决定 (这可以用 Z-buffer 算法实现). 在反向传播梯度流时, 我们需要计算某个像素对某个顶点位置的导数, 该导数可能

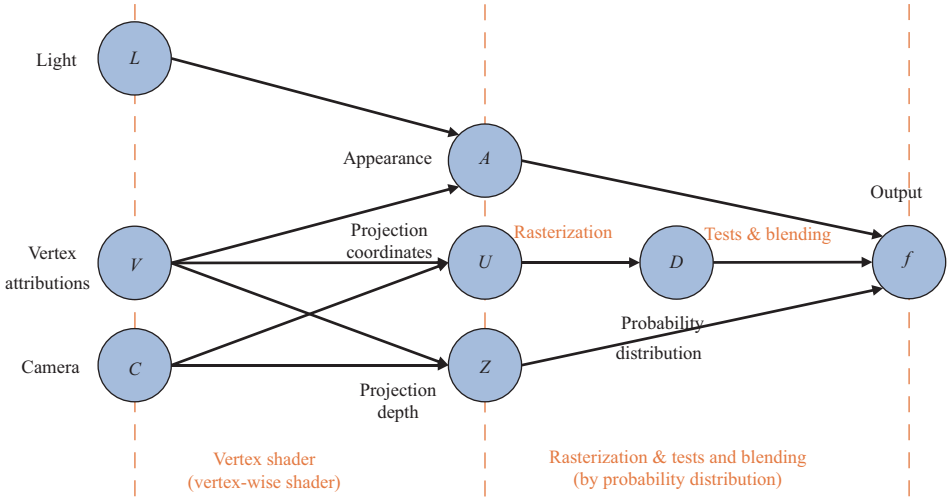

图 6 (网络版彩图) SoftRas 的渲染管线. 在顶点着色器步骤, 逐顶点计算顶点的颜色, 天然可微; 在光栅化和测试与混合步骤, 光栅化改编为计算概率密度分布, 测试与混合改编为按概率密度叠加

Figure 6 (Color online) SoftRas's rendering pipeline. Inside the vertex shader, each vertex's color is calculated separately, which is naturally differentiable. Rasterization is changed to obtain probability density distribution. The tests and blending stage are changed to add up probability density

与和该顶点相邻的所有三角形有关, 对于每个和该顶点相邻的所有三角形, 我们首先判断顶点移动到临界位置时像素是否会被遮挡 (可以利用存储的深度信息来判断), 如果像素总是被不包含该顶点的面遮挡, 也就是说无论如何移动该顶点, 像素的颜色都不会变化, 说明该顶点无法影响该像素的颜色, 那么我们将它的梯度设为 0.

4.3 基于概率分布的光栅化

由于传统的光栅化对顶点坐标不可微 (导数几乎处处为 0), 为了使得梯度流能够反向传播, 一种基于概率分布的光栅化方法 SoftRas [19] 被提出. 不同于前面介绍的两种近似导数的方法, 该方法使用概率分布的方法改编了光栅化和测试与混合两个步骤, 使得渲染的正向过程完全可微, 所以能够求出精确的导数.

SoftRas 的渲染管线如图 6 所示. 它的输入是顶点属性 V, 光照参数 L 和相机参数 C. 顶点着色器对顶点坐标进行变换, 得到投影坐标 U 和投影深度 Z, 并根据光照参数逐顶点计算颜色 A, 我们容易得到它们的微分, 梯度流可以反向传播. SoftRas 利用投影坐标计算每个三角形在图片上的概率分布 D 代替传统的光栅化过程, 之后使用投影深度将这些分布叠加代替传统的测试与混合过程, 最终得到渲染结果 f. 下面分别介绍如何计算每个三角形在图片上的概率分布, 以及如何叠加不同三角形的概率分布.

对于某个三角形 t_j, 我们估计它在图片上的概率分布 $\mathcal{D}_j$, 对于某个像素 p_i, 我们定义 t_j 在 p_i 上的概率为

$$\mathcal{D}_j^i = \text{sigmoid}\left(\delta_j^i \frac{d^2(p_i, t_j)}{\sigma}\right), \tag{3}$$

其中 σ 用于表示概率分布的锐利程度, 它的不同取值产生的效果如图 7 所示, δ_j^i 为符号函数, 若 p_i 在 t_j 内则为 $+1$, 否则为 -1; $d(p_i, t_j)$ 表示 p_i 到 t_j 的距离, 它可以采用不同的度量, 一个常用的选择是欧式距离. 我们不仅计算每个像素的概率分布, 还计算它的重心坐标, 对于三角形内的像素, 重心坐标

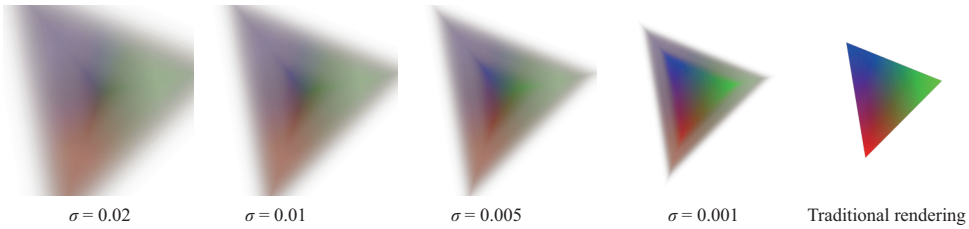

图 7 (网络版彩图) 单个三角形在不同参数下的概率密度分布及重心坐标, 其中 RGB 表示重心坐标乘以概率密度分布. 随着光滑参数越来越小, 结果越来越接近传统渲染

Figure 7 (Color online) A single triangle's probability density distribution and barycentric coordinates under different parameters. RGB represents that barycentric coordinates times probability density distribution. As σ decreases, results look more similar to that of traditional rendering

正常定义, 对于三角形外的像素, 重心坐标截取到 $[0,1]$ 的范围内. 如图 7 所示, RGB 值表示重心坐标. 我们用重心坐标插值顶点颜色得到三角形在像素上的颜色 C_j^i.

当得到了所有三角形在图片上的概率分布之后, 我们将它们按概率叠加, 这个过程需要用到深度信息. 离屏幕越近的三角形我们希望它的权重越高 (特别地, 对于传统渲染使用的 Z-buffer 算法, 只有最近的三角形权重为 1, 其他三角形权重都为 0). 对每个像素 p_i 求出每个三角形 t_j 的权重 w_j^i:

$$w_j^i = \frac{\mathcal{D}_j^i \exp(\frac{-z_j^i}{\gamma})}{\sum_k \mathcal{D}_k^i \exp(\frac{-z_k^i}{\gamma}) + \exp(\frac{\epsilon}{\gamma})}, \tag{4}$$

和环境光的权重 w_b^i:

$$w_b^i = 1 - \sum_j w_j^i = \frac{\exp(\frac{\epsilon}{\gamma})}{\sum_k \mathcal{D}_k^i \exp(\frac{-z_k^i}{\gamma}) + \exp(\frac{\epsilon}{\gamma})}, \tag{5}$$

其中 z_j^i 为三角形 t_j 在像素 p_i 的归一化深度, ϵ 为表示环境光的参数, γ 为表示叠加锐利程度的参数, 它的不同取值产生的效果如图 8 所示. 我们可以按如下公式计算像素颜色:

$$I(p_i) = w_b^i C_b + \sum_j w_j^i C_j^i, \tag{6}$$

其中 C_b 为环境光颜色. 可以看到像素颜色的计算过程完全可微, 梯度流可以传播回输入.

4.4 其他的方法

前面几节详细介绍了 3 种主流的可微渲染方法, 本小节将简要介绍几种其他的可微渲染方法.

TF Mesh Renderer [9] 和传统的渲染相同, 使用重心坐标插值顶点属性来计算像素的颜色. 它记录每个像素最近的三角形, 使用重心坐标来计算导数, 在三角形内的像素正常计算, 对于三角形外的像素, 使用负的重心坐标. 这种方法相当于把物体局部看作平的, 是对遮挡边界的导数的一种近似. 实践上, 对于具有光滑的遮挡边界且数目较少的情况处理较好, 难以处理遮挡边界复杂, 互相遮挡的情况.

DIB-R [46] 对前景和背景分开处理. 对于被至少一个三角形覆盖到的像素称为前景, 对于没有被三角形覆盖到的像素称为背景. 对于前景, 它采用和 TF Mesh Renderer [9] 类似的方法, 渲染时和传统的渲染相同, 求导时使用最靠近的三角形的重心坐标. 对于背景, 它改编了渲染的方法, 和 SoftRas [19] 类似, 它用概率分布近似三角形对三角形外的影响, 影响大小和像素到三角形的距离相关, 通过这些影响进行计算得到背景像素的 Alpha 值. 对于前景像素, 它将梯度传播回所在三角形的顶点; 对于背景像素, 它会将梯度传播回每一个顶点.

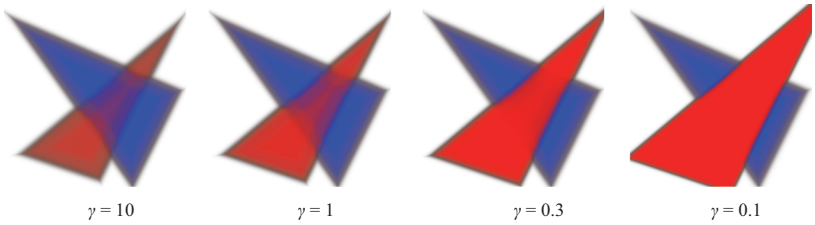

图 8 (网络版彩图) 不同参数下按概率分布叠加的结果. 其中红色三角形深度为 0, 蓝色三角形深度为 1, σ 固定为 0.001. γ 越小, 遮挡关系越明显, 当 $\gamma \to 0$ 时, 只有最近的三角形被绘制, 此时和传统的 Z-buffer 算法有完全相同的效果

Figure 8 (Color online) Results under different parameters. Red triangle has a depth of 0. Blue triangle has a depth of 1. σ equals 0.001. As γ decreases, occlusion becomes more apparent. When $\gamma \to 0$, only the triangle up front is rendered, which achieves the same effect as the traditional Z-buffer approach

DEODR [45] 在遮挡边界上使用重画不连续边技术 (discontinuity-edge-overdraw) [52], 把渲染改编成可微的形式. 它是一种可微的抗锯齿的方法, 通过沿着遮挡边界线性混合前面的三角形和后面的三角形, 使得当顶点运动时渲染结果连续变化. 它的整个渲染过程是可微的, 并且可以处理遮挡边界的变化对损失函数的影响, 也就是说损失函数的梯度流可以精确地传播回输入.

Versatile Scene Model [47] 提出了一种光滑的场景表示, 得到了一种可微的渲染方法, 它和传统的渲染完全不同, 有些类似于 SoftRas. 它把物体表示成一系列各向同性高斯 (Gauss) 分布, 每个高斯分布有各自的反射率, 并针对这种表示提出了特别的渲染模型和渲染方法. 这样场景函数是解析的, 渲染的过程可以看作光滑函数, 渲染的结果是较为模糊的图片.

4.5 不同方法之间的对比

首先, 本小节介绍的可微渲染方法均基于局部光照模型, 使用的方法大致分为两类, 一类是求得近似导数用于反向传播 [9,18,32], 另一类改编了传统渲染管线中的步骤, 使得像素对顶点可导 [19,45~47].

从上述可微渲染方法的渲染结果来看, 使用近似导数的方法 [9,18,32] 的渲染结果和传统渲染方法的结果相同, 但近似中使用的方法不能支持所有的渲染方式, 比如文献 [18] 只能支持逐顶点计算属性和颜色. 对于改编了渲染步骤的方法, 渲染结果一般会和传统渲染不同. DIB-R [46] 的渲染结果在前景上和传统渲染结果相同, 在背景上颜色和传统渲染相同, 但会有一定的透明度; DEODR [45] 的渲染结果在内部和传统渲染相同, 但在遮挡边界上会更加光滑; SoftRas [19] 的渲染结果会把不同三角形的颜色混合在一起, 带有一定的透明度和混合效果, 它的透明度和混合效果可以根据参数来调整, 当参数较大时, 它和传统渲染有明显差异, 当参数较小时, 它接近传统渲染的结果; Versatile Scene Model [47] 的渲染结果会比较模糊, 就像对图片做了一次高斯模糊. 我们在图 9 中展示了 3 种方法 [19,32,45] 的渲染结果供读者参考.

可微渲染的最终目的, 不同于传统渲染, 不是尽可能渲染出逼真的图片, 也不是为了求出尽可能精确的导数, 而是为了作为一个工具, 使得顶层应用 (一般是一个优化问题) 优化得更好 (渲染出逼真的结果以及求出精确的导数, 往往有助于优化问题). 具体表现为损失函数的梯度流能够传播回输入, 指导整个优化过程. 所以衡量可微渲染方法的重要指标就是, 梯度流是否能够很好地传播回输入, 是否能够正确地引导输入朝着损失函数更低的方向改变. 这个问题需要结合应用具体分析, 不同的应用有不同的需求, 实践的时候可以参考成功的案例, 详见第 7 节. 这里我们一般化地分析一下不同方法梯度流的行为. 首先对于近似导数的方法 [9,18,32] 和某些改编了传统渲染的方法 [45,46], 它们的梯度只能

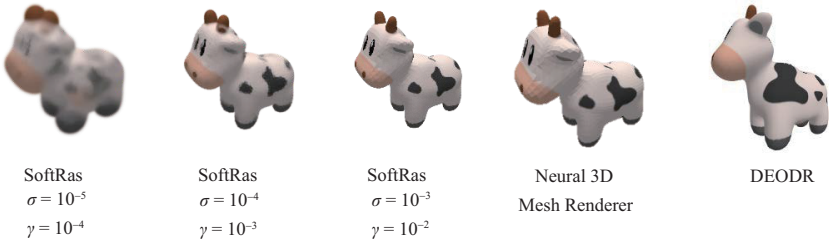

图 9 (网络版彩图) 展示了 3 种可微渲染的方法: 基于概率分布的光栅化 (SoftRas) [19]、用平滑的光栅化近似导数 (Neural 3D Mesh Renderer) [32]、DEODR [45] 的渲染结果供读者参考. 其中 Neural 3D Mesh Renderer [32] 代表了近似导数的一类方法, 它的渲染结果和传统渲染相同. SoftRas [19] 的渲染结果会把不同三角形的颜色混合在一起, 带有一定的透明度和混合效果, 它的透明度和混合效果可以根据参数来调整, 当参数较大时, 它和传统渲染有明显差异, 当参数较小时, 它接近传统渲染的结果. DEODR [45] 的渲染结果在内部和传统渲染相同, 但在遮挡边界上会更加光滑

Figure 9 (Color online) Three differentiable rendering methods' results for reference: probability distribution based rasterization (SoftRas) [19], smooth approximation for gradients (Neural 3D Mesh Renderer) [32], and DEODR [45]. Neural 3D Mesh Renderer [32] represents a group of methods that use approximated gradients and render images similar to traditional renderer. SoftRas [19] mixes different triangles' colors and produces translucent or blended effects, the extent of which is controlled by a parameter. When the parameter is relatively bigger, rendered results diverge further from traditional renderer. DEODR [45] are similar to traditional renderer except that it produces smoother occlusion boundaries

传播到最近的三角形, 不能传播到被遮挡的三角形; 并且由于只考虑了深度之间的遮挡关系, 没有考虑深度的大小与尺度, 所以只能在 xOy 平面上优化坐标, 它们的梯度也不能传播到深度信息来优化深度 z. SoftRas [19] 使用深度信息来混合不同深度的三角形, 所以梯度可以传播到被遮挡的三角形, 也可以优化深度; Versatile Scene Model [47] 使用高斯模型来表示场景, 光线会随着深度衰减, 所以可以优化深度. 对于 xOy 平面上的坐标, 对于优化几何的应用 (或是同时优化几何、材质和其他参数的应用), 这是我们最关心的部分, 也是不同方法之间区别的体现. 不同方法的感知范围不同, 文献 [18] 使用局部来近似梯度, 所以感知范围很小; 文献 [32] 使用光滑临界点之间的范围来近似梯度, 文献 [45] 使用了重画不连续边技术, 所以感知范围会更大; 文献 [9] 对遮挡边界做了平坦的近似, 虽然感知范围很大, 但无法很好反应遮挡边界; 文献 [19, 46, 47] 使用概率分布来表示场景或渲染背景, 可以感知到整张图片, 也能处理遮挡边界. 对于具体梯度流在 xOy 平面上如何传播, 图 10 展示了不同方法下像素对顶点的梯度. 对于颜色和相机参数, 由于不同方法用它们计算的方法类似, 所以梯度流的行为也较为相似, 并且梯度流的行为也和渲染方法相关.

5 基于全局光照模型的可微渲染

本节介绍基于全局光照模型的可微渲染. 传统的 OpenGL 渲染管线及根据其改编的可微渲染使用的是局部光照模型, 而全局光照模型和局部光照模型有很大区别, 相较于局部光照模型更加复杂但能渲染出更具有真实感的结果. 路径跟踪算法 [53] 是基于全局光照的可微渲染方法 [48, 49, 54] 的基础. 其基于蒙特卡洛 (Monte Carlo) 方法采样并跟踪进入眼睛的光线, 对于从眼睛出发的每一根光线, 在场景中遇到每一个物体, 根据物体的材质, 按概率得到一个方向进行反射或折射, 直到遇到光源, 超出场景范围或达到最大迭代次数. 也就是按概率分布采样光线所有可能的路径, 最后根据贡献进行积分, 这种渲染方法是一个无偏的全局光照渲染模型, 可以得到极具真实感的渲染结果. 下面介绍 3 种基于路径跟踪的全局光照可微渲染方法: 基于边采样的蒙特卡洛路径跟踪方法 [48]、重参数化不连续的被积

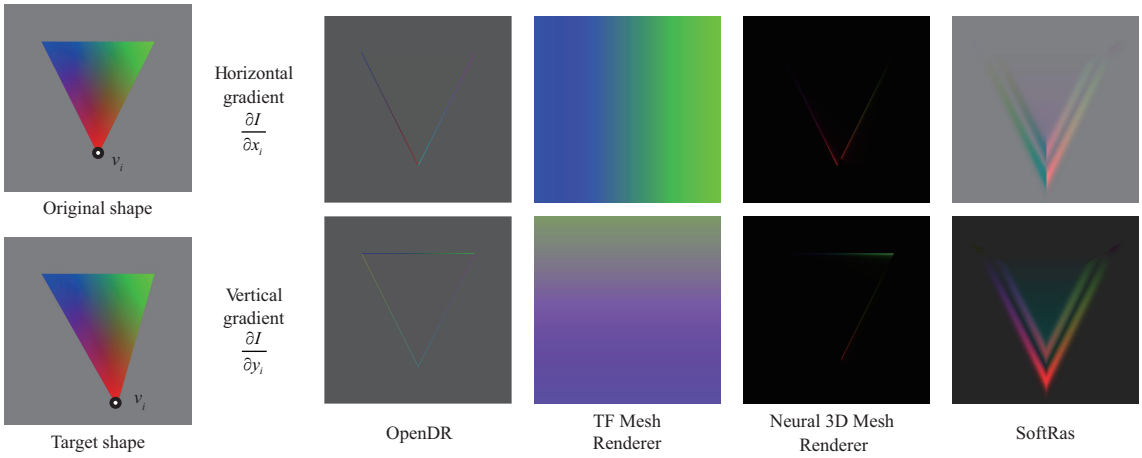

图 10 (网络版彩图) 像素对顶点 $v_i(x_i, y_i)$ 的梯度的直观展示. 比较了 4 种不同的方法, 分别是 **OpenDR** [18] (局部近似导数的方法), **TF Mesh Renderer** [9] (直接使用重心坐标计算导数), **Neural 3D Mesh Renderer** [32] (用平滑的光栅化近似导数) 和 **SoftRas** [19] (基于概率分布的光栅化). 可以看出 **TF Mesh Renderer** [9] 过于光滑, 对遮挡边界处理的不好, **SoftRas** [19] 感知的范围最大

Figure 10 (Color online) Pixel's derivative to vertex, including four different methods. OpenDR [18] (local approximation for gradients), TF Mesh Renderer [9] (directly use barycentric coordinates for gradient calculations), Neural 3D Mesh Renderer [32] (smooth approximation for gradients) and SoftRas [19] (probability distribution based rasterization). TF Mesh Renderer [9] produces an overly smooth effect with bad handling of occlusion boundaries. SoftRas [19] has the largest perceptive range

函数方法 [49] 和路径空间的可微渲染 [54].

5.1 基于边采样的蒙特卡洛路径跟踪

基于边采样的蒙特卡洛路径跟踪方法 [48] 可以分为两个阶段来理解, 第 1 阶段只考虑首要可见性, 在屏幕空间采样, 类似于局部光照模型, 第 2 阶段把该方法扩展到次要可见性, 可以处理阴影和全局光照, 这个阶段需要考虑不同的视角, 在三维空间采样, 还需要借助精妙设计的数据结构来处理更复杂的情况, 但主要思想类似.

在第 1 阶段主要考虑屏幕空间, 且只考虑像素对水平方向和垂直方向的偏导数, 对其他参数的偏导数可以通过链式法则和一些天然可微的部分求出. 我们写出路径跟踪算法计算某个像素值 I 的公式:

$$I = \iint f(x, y; \Phi) \mathrm{d}x\mathrm{d}y, \tag{7}$$

其中 Φ 为场景中需要优化的参数, $f(x, y; \Phi)$ 为场景函数. 由于上述积分一般没有闭形式解, 一般采用蒙特卡洛法来估计积分的结果. 我们的目的是求出像素 I 对场景参数 Φ 的导数, 但场景函数 f 不一定对场景参数 Φ 可导, 所以我们无法通过蒙特卡洛法进行采样来直接计算导数. 这是传统的按面积采样方法导致的, 在三角形的边上可能会产生不连续的结果, 比如像素颜色不随着顶点位置连续变化, 而是在某个临界位置发生突变, 这和局部光照模型有类似的结论. 该方法的核心思想就是在不连续的边上进行采样来计算积分, 对于某条边 (在屏幕空间) 我们用 $\alpha_i(x, y)$ 来表示这条边所在的直线, 它把平面分成两个半平面, 我们借助阶跃函数 θ 重写式 (7):

$$I = \sum_i \iint \theta(\alpha_i(x, y)) f_i(x, y) \mathrm{d}x\mathrm{d}y, \tag{8}$$

这里按照三角形的边把场景函数 f 划分成若干个子场景函数 f_i. 对上式求微分有

$$\begin{aligned}\nabla \iint \theta(\alpha_i(x,y))f_i(x,y)\mathrm{d}x\mathrm{d}y&\\ =\iint \delta(\alpha_i(x,y))\nabla\alpha_i(x,y)f_i(x,y)\mathrm{d}x\mathrm{d}y&\\ +\iint \theta(\alpha_i(x,y))\nabla f_i(x,y)\mathrm{d}x\mathrm{d}y,&\end{aligned}\tag{9}$$

其中 δ 为冲激函数. 该梯度分为两部分, 每个部分可以分别使用蒙特卡洛采样来估计. 第 1 部分包含冲激函数, 只在三角形的边上非零, 是积分中不连续的部分, 使用边采样来估计; 第 2 部分和渲染过程完全相同, 只是把场景函数换成了它的梯度, 是积分中连续的部分, 使用传统的面积采样来估计. 下面详细介绍如何对上述第 1 部分进行估计.

由于式 (9) 的第 1 部分包含冲激函数, 我们首先用冲激函数的性质把它改写成不包含冲激函数的形式, 考虑到只在三角形的边上非零, 我们重新参数化并把它改写成线积分:

$$\iint \delta(\alpha_i(x,y))\nabla\alpha_i(x,y)f_i(x,y)\mathrm{d}x\mathrm{d}y=\frac{1}{l_i}\int_a^b \nabla\alpha_i(x,y)f_i(x,y)\mathrm{d}s,\tag{10}$$

其中 l_i 为边的长度, s 为这条边的弧长参数, $[a,b]$ 为这条边在像素范围内的部分, $0\leqslant a\leqslant s\leqslant b\leqslant l_i$. 该积分是在三角形边上积分, 所以不再使用传统的面积采样, 而是使用边采样.

5.2 重参数化不连续的被积函数

一般化的路径跟踪的积分如下所示:

$$I=\int_{\mathcal{X}} f(x;\Phi)\mathrm{d}x,\tag{11}$$

其中 $\mathcal{X}$ 为积分的域, 一般为单位球面. 它在可见性发生变化时会变得不连续, 并且不连续的位置并非固定不变的, 它和场景参数 Φ 有关. 一种重参数化不连续的被积函数方法 [49] 被提出, 它通过重新参数化路径跟踪的积分, 使得不连续的位置和场景参数无关 (并非完全消除不连续性). 其核心思想是在 Φ 的邻域 $U(\Phi)$ 内找到一个参数变换 $T:\mathcal{Y}\times U(\Phi)\to\mathcal{X}$ 来重新计算积分:

$$\int_{\mathcal{X}} f(x;\Phi)\mathrm{d}x=\int_{\mathcal{Y}} f(T(y,\Phi);\Phi)|\det J_T|\mathrm{d}y,\tag{12}$$

其中 J_T 为 T 的雅可比 (Jacobi) 矩阵. 为了不影响渲染的正向计算, 参数变换被要求在当前场景参数 Φ 下为恒等变换. 参数变换也相当于一种新的采样过程, 该方法通过参数变换, 使得采样过程中不连续的位置不随场景参数变化而变化.

下面我们介绍如何得到这个参数变换, 下面的讨论假设积分的域 $\mathcal{X}$ 为单位球面 S^2. 对于某些场景只需要在非常小的角度内积分, 比如点光源加上很小的立体角, 这个时候积分中不连续的地方集中在物体的轮廓上, 此时可以用球面的旋转来近似参数对不连续物体轮廓的扰动, 将这个扰动也作用到采样过程上, 从而抵消扰动的影响. 在这种情况下, 假设存在一个旋转 $R(\omega,\Phi)$ 使得 $f(R(\omega,\Phi);\Phi)$ 在 ω 取任何值下都与 Φ 连续, 此时雅可比矩阵的行列式的绝对值为 1. 计算这个旋转不需要知道这个积分包含了哪些轮廓, 只需要知道场景参数在微小扰动下轮廓上点的变化量, 实现上使用几束光线来采样这个变化量. 对于需要在更大角度内积分的场景, 需要更复杂的参数变换才能抵消参数的影响, 这

个复杂的参数变换可以通过首先对场景进行一次卷积, 在卷积之后再进行旋转得到. 考虑到一个事实, 下述积分等于进行卷积之后积分:

$$\int_{S^2} f(\omega)\mathrm{d}\omega = \int_{S^2}\int_{S^2} f(\mu)k(\mu,\omega)\mathrm{d}\mu\mathrm{d}\omega, \tag{13}$$

其中 k 是一个球面上的卷积核, 对任何 ω 满足

$$\int_{S^2} k(\mu,\omega)\mathrm{d}\mu = 1, \tag{14}$$

这个卷积操作不会改变渲染的结果, 但会影响计算导数的过程. 原论文中使用 $\kappa = 1000$ 的 von Mises-Fisher 分布作为卷积核.

5.3 路径空间的可微渲染

在路径空间对路径积分进行微分的方法 [54] 的核心思想是推导出微分路径积分公式, 使用这个积分公式设计出一种蒙特卡洛法来无偏地估计像素值 (或辐照度) 对任意场景参数的微分. 由于该方法的推导过于复杂, 本小节只介绍核心方法和核心结论.

记场景为 $\mathcal{M}(\pi)$, π 为场景参数, 一条从光源到眼睛的路径表示为 $\bar{x} = (x_0, x_1, \ldots, x_N)$, 其中 $N \geqslant 1, x_i \in \mathcal{M}(\pi)$, 路径空间为 $\Omega = \cup_{N=1}^{\infty}\Omega_N, \Omega_N = \mathcal{M}(\pi)^{N+1}$. 路径积分可以表示为

$$I = \sum_{N=1}^{\infty}\int_{\Omega_N} f(\bar{x})\mathrm{d}\mu(\bar{x}), \tag{15}$$

其中 $\mathrm{d}\mu(\bar{x}) = \prod_{n=0}^{N}\mathrm{d}A(x_n)$ 为估计的路径密度, 它是表面面积 $\mathrm{d}A(x_n)$ 的乘积, $f(\bar{x})$ 是这条光线路径的能量, 它是每个点 x_n 的双向散射分布 (bidirectional scattering distribution function, BSDF) 在路径中的对应入射角出射角的值 $g(x_{n+1}; x_n, x_{n-1})$ 的乘积再乘上权重.

路径积分 I 对场景参数 π 求微分, 结果分为内部和边界两项, 这个微分原文中有两种形式, 一种是空间形式 (spatial form), 另一种是客观形式 (material form). 由于原文中的蒙特卡洛法使用的是客观形式, 我们这里写出客观形式. 我们首先简要介绍参考布局 (reference configuration), 它是一个抽象的二维流形 $\mathcal{B}$, 某个特定的参数 π 存在一个从 $\mathcal{B}$ 到 $\mathcal{M}(\pi)$ 的双射, 可以理解为抽象的参数空间. 客观形式是在参考布局上计算微分:

$$\frac{\partial I}{\partial \pi} = \int_{\hat{\Omega}}(\hat{f})'(\bar{p})\mathrm{d}\mu(\bar{p}) + \int_{\partial\hat{\Omega}}\Delta\hat{f}_K(\bar{p})V_{\Delta\mathcal{B}_K}(p_K)\mathrm{d}\mu'(\bar{p}), \tag{16}$$

其中第 1 项是内部项, 第 2 项是边界项, $\hat{\Omega}$ 是和参数 π 无关的在参考布局上的客观路径空间, $\partial\hat{\Omega}$ 是其中至少有一个顶点在边界上 (进行微小扰动产生的变化不连续) 的路径, $\hat{f}$ 是在客观路径空间上定义的对光线路径能量的计算函数, 可以由 f 重新参数化后得到, 对应地其中的 BSDF g 重新参数化为 $\hat{g}$, V 是和 π 无关的标量法向速度 (scalar normal velocity), $\Delta\mathcal{B}_K$ 由一系列 $\hat{g}$ 对参数 p_K 不连续的边界曲线组成.

下面介绍对内部项和边界项的估计算法. 由于内部项的形式和路径积分相似, 对内部项的估计可以在计算路径积分时同时求出, 算法流程和路径跟踪算法相同, 论文中采用单向路径跟踪算法. 单向路径跟踪的算法是一个采样过程, 每次在参考布局中采样一条路径, 通过跟踪路径计算这条路径的能量. 对于边界项的估计, 论文中给出了一种新的基于蒙特卡洛法的无偏估计算法: 多向路径采样算法 (multi-directional sampling). 由于边界点是路径中间的某一段导致的, 如何采样这样的边界路径是这

个问题最大的挑战, 单向的路径跟踪需要每一步都变换视角来寻找边界点, 为了回避这个困难, 这个算法的核心思想是: 根据边界点把路径分成两部分, 前一段是连接光源的子路径, 后一段是连接视点的子路径, 在采样的过程中, 先在场景曲面上采样一个点和一个方向, 过这一点作这个方向的直线, 向两端延伸得到两个顶点, 将这两个顶点加入路径, 采样点本身不属于这条路径, 然后从这两个顶点出发采样两个子路径, 这样采样得到的路径为边界路径. 计算边界项的过程依然是路径跟踪框架, 其中采样的过程使用多向路径采样算法, 被积函数可以由参考布局上的路径计算得到.

5.4 比较和分析

上述 3 种方法 [48, 49, 54] 分别提出了一种不同于传统路径跟踪的采样过程. 边采样的方法把路径积分中不连续的部分剥离出来, 在不连续的边上进行采样来计算积分的不连续部分, 在连续的部分使用传统的面积采样; 重参数化的方法通过对积分进行参数变换, 使得采样过程中不连续的位置不随场景参数变化而变化; 路径空间的可微渲染把路径积分的微分拆成内部项和边界项两项, 分别在路径空间使用蒙特卡洛法进行估计. 其中边采样的方法具有完全可微的正向过程, 所以它的导数是完全精确的, 但它对场景有一定的限制和假设, 比如不包含点光源; 重参数化的方法求得的是近似导数, 并且对某些场景并不能很好地支持, 求得的导数有很大的误差; 路径空间的可微渲染不改变路径跟踪的算法, 并且微分的过程也是路径跟踪的框架, 在路径空间求得路径积分的微分的无偏估计.

6 开源的可微渲染工具

已经有许多开源的可微渲染工具, 比如 DEODR [45]1), SoftRas [19]2), OpenDR [18]3), DIRT [55]4), PyTorch3D [56]5), Neural 3D Mesh Renderer [32]6), TF Mesh Renderer [9]7), Tensorflow Graphics [57]8), DIB-R [46]9), Redner [48]10), Mitsuba2 [49]11) 等. 它们的特性如表 1 所示, 其中 DEODR, SoftRas, OpenDR, DIRT, Neural 3D Mesh Renderer, TF Mesh Renderer, Tensorflow Graphics, 和 DIB-R 使用的是局部光照模型, Redner 和 Mitsuba2 使用的是全局光照模型, 其渲染的结果更接近现实世界, 对输入为现实世界的照片的应用可能会有更好的效果, 但时间开销会更大, 而且参数更多更难优化. 其中 SoftRas, DIB-R, PyTorch3D 和 Neural 3D Mesh Renderer 基于深度学习框架 PyTorch; TF Mesh Renderer, DIRT 基于深度学习框架 Tensorflow; Tensorflow Graphics 是 Tensorflow 的图形库; DEODR 和 Redner 同时适用于深度学习框架 PyTorch 和 Tensorflow; OpenDR 是 Python 的包; Mitsuba2 是 C++ 的库, 也有深度学习框架 PyTorch 的接口. 其中 DEODR, SoftRas 和 Redner 使用的方法具有完全可微的正向过程, 所以能求出精确的导数; OpenDR, DIRT, Neural 3D Mesh Renderer 和 TF Mesh Renderer 使用不同的方法对导数进行近似, 而不改变渲染的正向过程; TF Mesh Renderer 和 Tensorflow Graphics 无法很好地处理遮挡边界的梯度, 有一定的局限性; Redner 和 Mitsuba2 使用的是全局光照模型, 能够对间接

1) https://github.com/martinResearch/DEODR.
2) https://github.com/ShichenLiu/SoftRas.
3) https://github.com/mattloper/opendr.
4) https://github.com/pmh47/dirt.
5) https://github.com/facebookresearch/pytorch3d.
6) https://github.com/hiroharu-kato/neural_renderer, https://github.com/daniilidis-group/neural_renderer.
7) https://github.com/google/tf_mesh_renderer.
8) https://github.com/tensorflow/graphics.
9) https://github.com/nv-tlabs/DIB-R.
10) https://github.com/BachiLi/redner.
11) https://github.com/mitsuba-renderer/mitsuba2.

表 1　开源可微渲染工具特性

Table 1　Features of DR tools with open source

Method	PyTorch	Tensorflow	Exact derivative	Global illumination
DEODR [45]	√	√	√	×
SoftRas [19]	√	×	√	×
OpenDR [18]	×	×	×	×
DIRT [55]	×	√	×	×
PyTorch3D [56]	√	×	×	×
Neural 3D Mesh Renderer [32]	√	×	×	×
TF Mesh Renderer [9]	×	√	×	×
Tensorflow Graphics [57]	×	√	×	×
DIB-R [46]	√	×	×	×
Redner [48]	√	√	√	√
Mitsuba2 [49]	√	×	×	√

光照求导, 但由于需要对光线的分布进行大量采样, 计算花费的时间也是巨大的, 不过 Redner 支持使用局部光照的模式来加速, 只考虑主要可见性, 在这个模式下依然具有完全可微的正向过程和精确的导数.

纹理贴图在渲染中很重要, 我们简要地介绍不同的开源工具对纹理贴图的支持. 其中 DEODR, SoftRas, OpenDR, Neural 3D Mesh Renderer, DIB-R, DIRT, Redner, Mitsuba2 支持带纹理贴图的可微渲染, TF Mesh Renderer, Tensorflow Graphics 不支持带纹理贴图的可微渲染, 其中 OpenDR 采用 Mipmap, Neural 3D Mesh Renderer 使用特殊的按重心坐标访问的纹理, 每个三角形的纹理是一个立方体.

7　可微渲染的应用

可微渲染有着广泛的应用, 主要集中在逆渲染上, 包括几何重建、材质和纹理估计、相机参数估计和光照参数估计等. 如表 2 [5~44, 58~62] 所示, 我们把不同的工作按照使用的开源可微渲染工具、研究对象的类别和领域分类.

7.1　物体

从二维到三维的重建任务通常可以定义为, 希望根据输入的一张或多张图片推导得到场景中出现的物体的三维模型 (包括几何、纹理和材质) 以及物体的位置等信息, 即从渲染的输出得到渲染的输入. 对于渲染的每一类输入, 都有专门的应用模型.

7.1.1　几何和位置优化

几何是指关于一个物体的形状的信息, 一般表现为三角网格 (mesh), 对于具有先验信息的一类物体, 可能有参数表示的形式. 位置由物体在场景中的模型矩阵进行表示, 包括平移、旋转和放缩变换. 三维重建的一类重要的方法是合成分析 (analysis-by-synthesis), 即根据三维参数渲染得到二维图片, 与真实图片进行对比, 从而不断调整三维参数直到二者接近相同. 如果采用传统的渲染方法, 为了训练网络则需要大量的关于场景中物体的三维标注或者多视角标注, 而可微渲染可以做到将像素的导数反

表 2 可微渲染应用分类

Table 2 Classification of applications

Differentiable rendering method	Ref.
OpenDR [18]	[23∼27, 40, 58]
Nueral 3D Mesh Renderer [32]	[12, 14, 20∼22, 28∼31, 33, 34, 36∼42, 59, 60]
SoftRas [19]	[15, 43, 44]
TF Mesh Renderer [9]	[13, 16, 17]
Object	**Ref.**
Artifact and ordinary object	[12, 19, 32∼40, 40∼44]
Face	[5∼17]
Body	[18∼27]
Hand	[28∼31]
Field	**Ref.**
Semantic segmentation	[33, 34, 36]
Geometry and posture	[7, 9, 12, 13, 15, 16, 18∼21, 23∼28, 30∼33, 38∼40, 42, 43, 58]
Texture and reflectance	[10, 17, 23, 35, 38, 40, 43, 61, 62]
Illumination estimation	[6, 7]
Camera calibration	[38, 43, 60]
Image synthesis	[14, 22, 29, 37, 44, 59]
Adversarial examples	[41, 42]

向传播至三维模型的顶点, 因此网络可以接受单张图片作为输入, 直接利用更容易得到的二维标注信息, 甚至让弱监督乃至无监督训练成为可能.

为了根据合成图片与真实图片几何的差异计算损失函数, 最常使用的二维标注是物体的掩膜 [33, 38∼40], 也有模型在掩膜的基础上计算语义关键点的距离 [38, 40]. 除了利用手工标注的数据集之外, 由于计算机视觉在一些特定任务下对于处理图片存在一些成熟的模型, 也有一些工作选择利用目前最高水平的既有模型自动标注, 进一步降低了数据集构造的成本, 例如文献 [33] 利用现有的语义分割的模型对输入图片处理得到掩膜和包围盒. 除此之外, 还有模型利用了现有的真实感渲染器, 在合成图片的数据集上训练, 因此可以直接保留包括掩膜、语义关键点、位置、材质等渲染参数作为标注 [40]. 计算两张图像之间像素级的距离是最直接的设计损失函数的方式, 例如掩膜 [33, 38∼40]、像素的强度 [12]. 然而如果仅仅直接利用像素数据, 可能导致对小的几何瑕疵敏感, 导致重建图片模糊. 为此一些模型引入感知损失, 即将合成图片与真实图片输入训练后的神经网络模型, 在多个隐层的特征图上定义距离 [12, 39, 40].

7.1.2 纹理和材质优化

在渲染管线中, 着色器会利用物体表面材质、光照、视角等信息, 计算待渲染物体表面像素的颜色. 一种常用的着色方式是使用纹理贴图 (texture mapping). 根据用途不同, 纹理贴图可分为法向贴图、漫反射贴图、粗糙度贴图、高光贴图等, 其包含了物体表面的材质信息. 传统的纹理材质建模需要采集大量不同光照、视角下目标物体的图片, 以进行数据驱动的重建. 而近年来, 基于深度学习的重建方法使得仅从少量甚至单张图片估计物体的纹理材质信息成为可能. 其中, 可微渲染器的使用保证了此类方法的可行性.

纹理材质重建通常是三维模型重建问题的一部分, 为了获取真实的纹理材质信息, 研究者们提出了大量的方法. 一类方法 [38, 40, 43] 基于纹理流. Kanazawa 等 [38] 提出了一种从单张图片重建特定类别的物体的三维模型的方法, 在纹理重建部分, 其采用了预测纹理流的方法, 即预测从输入图像上采样的坐标值, 然后通过采样和双线性插值生成纹理贴图, 克服了直接回归像素值产生的纹理贴图较为模糊的缺点. Li 等 [43] 对文献 [38] 的方法提出了两点的改进. 其提出了 EM 式的渐进训练方法, 迭代地提升预测纹理流的准确度; 同时提出了一种基于投影闭环的纹理一致性误差, 解决了纹理流预测中具有相近颜色的面的纹理从同一个像素采样的问题. Zuffi 等 [40] 与文献 [38] 有类似的思路, 但在回归网络给出预测结果后, 还增加了优化步骤. 另一类方法则从建立高质量的纹理模型入手. 文献 [10, 17] 在解决人脸重建的问题时, 将纹理生成对抗网络引入到重建网络中替代三维可变形模型 (3D morphable model, 3DMM) 的纹理模型, 使得重建的纹理细节丰富, 具有较高的保真度.

7.2 人体相关

相比于普遍意义下的物体, 与人相关的重建问题受到人们格外的关注, 理解人在图片或视频中几何与位置的三维信息在自动驾驶、机器人、虚拟现实等大方向中有重要的应用价值. 专注于人的重建问题的模型与其他模型有着相同的基本思路, 与此同时人也有一系列特有的性质, 使得一些针对性的解法可以达到更佳效果, 这些性质包括: 人作为整体以及各个部位基本形状已知, 有多种成熟的参数模型; 除了多角度拍摄, 人可以自主做出动作, 重建模型的输入可能是一段视频序列; 人有定义在生理结构上的关节, 关节包含了重要的几何信息. 与人相关的重建问题中最受关注的主要包括人脸、人体, 以及人手重建.

7.2.1 人脸

从二维图像重建人脸的三维模型是近期计算机视觉和图形学领域的研究热点之一. 带标注的三维人脸数据的缺乏, 使得采用无监督或弱监督的方式训练三维重建网络的方法受到了大量的关注. 其中, 部分工作 [9, 10, 13, 17] 在网络中引入了诸如 3DMM [63] 的三维人脸模型来对三维人脸数据进行表示, 部分工作 [12] 则不涉及现成模型.

图 11 [64, 65] 展示了基于参数模型重建三维人脸模型的无监督或弱监督网络的典型框架. 输入图像通过回归网络预测参数, 由可微渲染器合成图片, 再利用输入图像与输出图像计算损失, 从而更新回归网络的参数. 在具体实现上, 每一部分可能有不同的实现方式, 也可能存在结构和步骤的新增与替换.

文献 [9] 提出的方法采用编码器 – 解码器结构, 利用合成图像和无标注的二维真实图像进行训练. 其使用了可微渲染器 [9] 来计算并反向传播损失函数, 损失函数包括像素损失、关键点损失、身份损失和内容 (人脸识别网络的隐藏层特征) 损失. 文献 [13] 提出了一种基于 CNN 的从二维人脸图像重建精确的三维人脸模型的弱监督方法. 其使用 3DMM 模型表示人脸, 采用 ResNet-50 [66] 来预测三维模型参数及成像参数, 然后通过可微渲染器生成重建人脸的二维图像. 该方法采用图像和认知两个层面的损失函数以及 3DMM 系数的正则项指导网络的训练, 其中, 图像层面采用了结合注意力机制的像素误差和人脸的关键点误差, 认知层面采用 FaceNet [67] 提取人脸特征计算误差. 文献 [12] 提出了一种无监督的不依赖现有模型重建具有对称性的物体的方法. 其思想是从单张图像分解出图像的构成信息, 即深度、反照率、视角与光照, 然后结合对称性以及置信图, 采用可微渲染器 [32] 重建二维图像, 然后使用输入图像与生成图像的重建误差作为损失函数.

可微渲染理论在为三维人脸重建的问题提供新的范式的同时, 也促进了无监督或弱监督的三维人

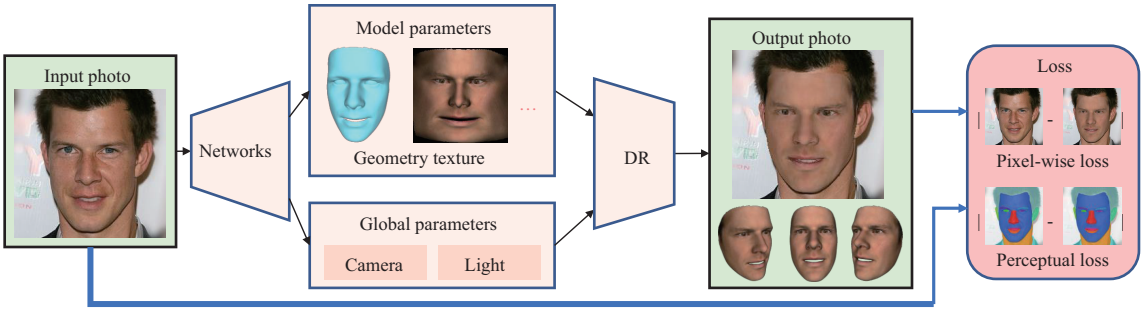

图 11 (网络版彩图) 输入图像通过回归网络得到几何、纹理等三维人脸模型的参数以及全局参数, 然后通过可微渲染器合成人脸图片, 再利用输入图像与输出图像计算损失, 更新回归网络的参数. 图中认知层面损失的示意图使用文献 [64] 生成

Figure 11 (Color online) The regression network receives images as input and outputs 3D human faces model parameters such as geometry and texture. Differentiable renderer uses the produced parameters and re-render face images which are used to calculate loss and to update the regression network. The reference images in the perception layer are produced by [64]

脸重建网络在相关领域的模块化应用. 文献 [11] 在提出的人脸视频去模糊网络中, 创新性地集成了使用可微渲染器的三维人脸重建网络以提供人脸的先验信息. 通过借助重建网络渲染的清晰人脸结构和提取的人脸身份信息, 该方法克服了传统方法定位面部结构不准确、结果不够自然的缺点, 能够产生高质量的去模糊视频. 文献 [5] 使用可微渲染器的三维人脸重建网络到替换身份的生成网络中. 其借助重建网络从输入图像重建三维人脸模型, 然后替换身份参数渲染具有不同身份信息的二维人脸图像, 再经过生成对抗网络的修正, 生成身份信息已混淆的、具有较好细节和真实感的图像, 达到保护隐私的目的.

7.2.2 人体

人体可以被视为一类形状特定且位姿服从一定规律的物体, 在单目图片作为输入导致问题不适定的情况下, 利用人体相关的先验知识可以提升模型性能. 人体参数模型到面片模型的相互转化过程可微, 可以参与重建管线的整体训练. 人体的关节是天然的语义关键点, 通过关节往往就可以获得足够的信息重建人体几何. 根据二维关节标注, 以及预测的三维关节在二维上的投影, 我们可以计算它们的距离来设计损失函数. 文献 [25] 在应用可微渲染之后, 损失函数定义在有二维标注的掩膜和关节上. 文献 [20] 在应用可微渲染之后, 重建部分的损失函数仅定义在关节标注上.

7.2.3 人手

人手重建问题对机器手控制、虚拟现实手势识别等方向有着重要意义. 人手的手指与手掌构成精巧的结构、自遮挡、背景杂乱和与物体的互动等复杂场景给重建带来多种困难. 目前缺乏有完整三维标注的数据集, 可微渲染的出现提高了二维标注信息的可用性, 一方面可以使用可微渲染搭建利用二维标注的重建模型, 另一方面可以将可微渲染视为产生三维标注的一步. 如果有模型可以从二维标注信息恢复高质量的三维信息, 就可以生成更多的标注数据. 即便模型结果的质量不足以作为标注投入使用, 也可以在交互的过程中降低成本.

8 可微渲染未来可能的发展方向

8.1 下一代可微渲染的方法

现有的可微渲染方法的研究和应用主要集中在局部光照模型上, 因为局部光照模型对于简单的场景已经十分有效. 全局光照模型计算开销过大, 也会大大增加训练过程的时间花费. 对于局部光照模型的可微渲染, 它的定位应当是主要处理具有高度先验信息的简单场景, 比如人脸或是人手, 除此以外还可以用来渲染某些属性, 比如 PNCC 或是法向投影. 所以它的发展趋势应当是: (1) 梯度流更精确且更具有指导意义; (2) 对渲染模型的假设和限制更少, 比如不再要求对颜色的计算是逐顶点的; (3) 对一些先进的实时渲染技术进行支持, 比如阴影和近似的全局光照; (4) 支持先进的抗锯齿技术, 得到无锯齿的合成图片, 从而在物体边界上有更精确的结果. 对于全局光照模型的可微渲染, 最迫切的需求是进行加速, 比如减少对采样的次数的需求, 或是对渲染质量和渲染速度进行折衷. 对于全局光照模型在逆渲染的应用, 由于我们已经有积分的结果, 也就是输入的真实图片, 所以基于全局光照的可微渲染方法不一定需要局限于路径跟踪的框架, 利用积分的结果来估计参数和导数可能存在更简单的方法.

可微渲染的一大应用是逆渲染, 对于逆渲染这个特别的应用, 往往已经具有真实的渲染结果了, 我们可能还有对渲染输入的一个初始估计. 但通常可微渲染方法都没有考虑到我们已经有这样一个渲染结果了, 在渲染时只考虑了渲染的输入, 因为大多数可微渲染方法在设计时并不局限于逆渲染的应用. 对于固定逆渲染应用, 设计可微渲染方法的一个可能的尝试是, 将这个真实的渲染结果利用上, 使渲染速度更快并让渲染结果更好.

一个某些模块可以自由编程的可微渲染平台也是非常具有价值的, 这种平台类似于 OpenGL 的可编程管线, 又类似于自动求导的工具. 因为现有的可微渲染工具, 大多步骤和模块都是固定的, 这可能是由于现有的可微渲染方法对渲染模型的假设和限制过大. 所以当一个足够通用的可微渲染方法被提出, 这样的可编程可微渲染平台也会随之出现.

8.2 可微渲染的应用

可微渲染已经有丰富的应用, 在第 7 节已经进行了详细的介绍. 除了这些应用之外, 可微渲染还有很大的潜力等待被发掘. 可微渲染应用的发展不能离开可微渲染方法和理论的发展, 也促进了可微渲染方法和理论的发展. 在我们之前提到的应用中, 逆渲染 (包括重建几何和位置、纹理和材质、光照和相机等参数) 是最主要也是最自然的一类应用; 除此之外使用可微渲染合成图片 (包括语义分割和图片合成) 则更进了一步, 它利用可微渲染的正向过程来生成结果; 使用可微渲染生成对抗样本则跨度更大, 解决了看似不相关的问题. 许多毫不相关的问题实际上可以通过某种精妙设计的方法联系起来, 而可微渲染本身就是图形学和计算机视觉之间的桥梁, 也是三维模型和二维图片之间的桥梁, 它更可能被用于各种意想不到的应用, 比如可能用作分析数据分布和结构的工具. 除此以外, 利用可微渲染制作生成数据或是带标注的数据集是一件有意义的事情.

对于现有的应用, 依然有许多未解决的问题. 对于从单张照片进行重建的应用, 由于信息的缺失, 一方面只能恢复部分的三维模型; 另一方面, 如何完全解耦光照相机等全局参数和形状纹理参数依然是一个大问题, 比如人脸重建领域, 在弱透视投影下无法解耦位姿参数和表情参数 [68]. 现有的许多方法都是借助可微渲染来训练一个深度神经网络的, 通过神经网络预测出参数, 但神经网络的稳定性、鲁棒性和可解释性都有待提高. 对于不同视角的照片或是对照片进行微小旋转, 预测的结果变化很大. 对于视频应用, 许多预测的结果不够光滑从而产生大量抖动, 或是不同帧的结果差异较大. 可微渲染本

身具有稳定性、鲁棒性和可解释性, 今后可能借助可微渲染来设计通用具有稳定性、鲁棒性和可解释性的神经网络或是其他方法.

8.3 分析数据的工具

对于解析的可微的渲染方法, 我们可以利用它来分析数据分布的结构, 比如数据的概率分布和流形结构. 许多机器学习的方法对数据都有一定的假设, 比如概率分布的假设或是流形结构的假设, 但这些假设很难验证, 也很难去确认它的结构. 一方面是因为得到的数据只是数据真实分布的一些采样点, 对于维度非常高的数据空间, 很难用少量采样点来描述. 生成的数据能够很好地帮助我们解决这个问题, 比如对于图片相关的工作, 利用传统渲染方法生成数据能够有效地帮助我们解决许多问题, 但还不足以帮助我们完全了解数据的真实分布和结构. 我们可以把传统的渲染方法当作黑盒的数据采样器, 解析的可微渲染方法就是白盒的数据采样器. 想要完全了解数据的真实分布, 黑盒的数据采样器还不足够, 因为数据分布和结构可能很复杂. 使用可微渲染器生成对抗样本 [41, 42] 可能已经是这个方向的萌芽. 假设数据具有流形结构, 了解数据的内在维度, 是了解数据的一种方法. 对于生成一类数据, 可能有很多输入参数, 但相比于数据的内在维度可能要多很多, 因为许多参数之间互相耦合并不独立. 如果我们有一个完全解析的渲染过程, 我们可以借助这个渲染过程来分析数据的内在维度. 当我们对数据的分布和结构有一定的理解之后, 一方面能帮助我们设计更好的模型来解决实际问题, 另一方面也能帮我们解释某些黑盒模型.

9 总结

本文首先介绍了传统渲染管线, 以 OpenGL 的可编程渲染管线为例, 介绍了传统渲染管线中每个步骤的任务. 随后逐步引入可微渲染的主要思想、基本原理和方法, 对它们进行介绍、分析和比较. 其中详细介绍了 3 种主流的可微渲染方法和几种其他的可微渲染方法, 大致可以分为两类, 一类是求得近似导数用于反向传播, 另一类是改编了传统渲染管线中的步骤, 让像素对顶点可导. 本文介绍了 3 种基于路径跟踪的可微渲染, 一种是基于边采样的蒙特卡洛路径跟踪的方法, 一种是重参数化不连续的被积函数的方法, 另一种是路径空间的可微渲染方法. 本文随后列出了一些开源的可微渲染工具, 并整理了它们的特性, 供读者参考并进行比较. 本文介绍了可微渲染的广泛应用, 分为人脸、人体、人手和物体 4 个方面. 本文在最后列举了一些可微渲染可能的发展方向以及对未来进行展望. 希望本综述能帮助读者快速理解可微渲染的思想, 找到最适合的可微渲染方法以及开源工具.

参考文献

1 Tulsiani S, Zhou T, Efros A A, et al. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 2626–2634

2 Sainz M, Pajarola R. Point-based rendering techniques. Comput Graph, 2004, 28: 869–879

3 Ward K, Bertails F, Kim T, et al. A survey on hair modeling: styling, simulation, and rendering. IEEE Trans Visual Comput Graph, 2007, 13: 213–234

4 Macklin M, Müller M. Position based fluids. ACM Trans Graph, 2013, 32: 1–12

5 Tewari A, Zollhöfer M, Kim H, et al. Mofa: model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: Proceedings of IEEE International Conference on Computer Vision, Venice, 2017. 3735–3744

6 Calian D A, Lalonde J F, Gotardo P, et al. From faces to outdoor light probes. In: Proceedings of Computer Graphics Forum, 2018. 51–61

7 Sengupta S, Kanazawa A, Castillo C D, et al. SfSNet: learning shape, reflectance and illuminance of faces 'in the wild'. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 6296–6305
8 Tewari A, Zollhöfer M, Garrido P, et al. Self-supervised multi-level face model learning for monocular reconstruction at over 250 HZ. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 2549–2559
9 Genova K, Cole F, Maschinot A, et al. Unsupervised training for 3D morphable model regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018
10 Gecer B, Ploumpis S, Kotsia I, et al. GANFIT: generative adversarial network fitting for high fidelity 3D face reconstruction. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 1155–1164
11 Ren W, Yang J, Deng S, et al. Face video deblurring using 3D facial priors. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, Seoul, 2019. 9387–9396
12 Wu S Z, Rupprecht C, Vedaldi A. Unsupervised learning of probably symmetric deformable 3D objects from images in the wild. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020
13 Deng Y, Yang J, Xu S, et al. Accurate 3D face reconstruction with Weakly-supervised learning: from single image to image set. 2020. ArXiv:1903.08527
14 Zhou H, Liu J, Liu Z, et al. Rotate-and-render: unsupervised photorealistic face rotation from single-view images. 2020. ArXiv:2003.08124
15 Zhu W, Wu H, Chen Z, et al. ReDA: reinforced differentiable attribute for 3D face reconstruction. In: Proceedings of Capacitated Vehicle Routing Problem, 2020
16 Wang K, Peng X, Yang J, et al. Suppressing uncertainties for large-scale facial expression recognition. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020
17 Lee G H, Lee S W. Uncertainty-aware mesh decoder for high fidelity 3D face reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020
18 Loper M M, Black M J. OpenDR: an approximate differentiable renderer. In: Proceedings of European Conference on Computer Vision. Berlin: Springer, 2014. 154–169
19 Liu S, Li T, Chen W, et al. Soft rasterizer: a differentiable renderer for image-based 3D reasoning. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019
20 Jiang W, Kolotouros N, Pavlakos G, et al. Coherent reconstruction of multiple humans from a single image. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020
21 Hasson Y, Tekin B, Bogo F, et al. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. 2020. ArXiv:2004.13449
22 Liu W, Piao Z, Min J, et al. Liquid warping GAN: a unified framework for human motion imitation, appearance transfer and novel view synthesis. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, Seoul, 2019. 5903–5912
23 Bogo F, Black M J, Loper M, et al. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In: Proceedings of 2015 IEEE International Conference on Computer Vision, Santiago, 2015. 2300–2308
24 Bogo F, Kanazawa A, Lassner C, et al. Keep it SMPL: automatic estimation of 3D human pose and shape from a single image. In: Proceedings of the 14th European Conference on Computer Vision Amsterdam, 2016. 561–578
25 Pavlakos G, Zhu L, Zhou X, et al. Learning to estimate 3D human pose and shape from a single color image. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 459–468
26 Huang Y, Bogo F, Lassner C, et al. Towards accurate marker-less human shape and pose estimation over time. In: Proceedings of 2017 International Conference on 3D Vision (3DV), 2017. 421–430
27 Lassner C, Romero J, Kiefel M, et al. Unite the people: closing the loop between 3D and 2D human representations. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 2017. 4704–4713
28 Zhang X, Li Q, Mo H, et al. End-to-end hand mesh recovery from a monocular RGB image. 2019. ArXiv:1902.09305
29 Zimmermann C, Ceylan D, Yang J, et al. FreiHAND: a dataset for markerless capture of hand pose and shape from single RGB images. 2019. ArXiv:1909.04349

30 Baek S, Kim K I, Kim T. Pushing the envelope for RGB-based dense 3D hand pose estimation via neural rendering. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 1067–1076
31 Baek S, Kim K I, Kim T K. Weakly-supervised domain adaptation via gan and mesh model for estimating 3D hand poses interacting objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020
32 Kato H, Ushiku Y, Harada T. Neural 3D mesh renderer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018
33 Yao S, Hsu T M H, Zhu J Y, et al. 3D-aware scene manipulation via inverse graphics. 2018. ArXiv:1808.09351
34 Kulkarni N, Gupta A, Tulsiani S. Canonical surface mapping via geometric cycle consistency. 2019. ArXiv:1907.10043
35 Gao D, Li X, Dong Y, et al. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. ACM Trans Graph, 2019, 38: 1–15
36 Gur S, Shaharabany T, Wolf L. End to end trainable active contours via differentiable rendering. 2019. ArXiv:1912.00367
37 Luo A, Zhang Z, Wu J, et al. End-to-end optimization of scene layout. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 3754–3763
38 Kanazawa A, Tulsiani S, Efros A A, et al. Learning category-specific mesh reconstruction from image collections. 2018. ArXiv:1803.07549
39 Kato H, Harada T. Learning view priors for single-view 3D reconstruction. 2019. ArXiv:1811.10719
40 Zuffi S, Kanazawa A, Berger-Wolf T Y, et al. Three-D safari: learning to estimate zebra pose, shape, and texture from images "in the wild". In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, Seoul, 2019. 5358–5367
41 Alcorn M A, Li Q, Gong Z, et al. Strike (with) a pose: neural networks are easily fooled by strange poses of familiar objects. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 4845–4854
42 Xiao C, Yang D, Li B, et al. Meshadv: adversarial meshes for visual recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 6898–6907
43 Li X, Liu S, Kim K, et al. Self-supervised single-view 3D reconstruction via semantic consistency. In: Proceedings of European Conference on Computer Vision, 2020
44 Liao Y, Schwarz K, Mescheder L M, et al. Towards unsupervised learning of generative models for 3D controllable image synthesis. 2019. ArXiv:1912.05237
45 de La Gorce M, Fleet D J, Paragios N. Model-based 3D hand pose estimation from monocular video. IEEE Trans Pattern Anal Mach Intell, 2011, 33: 1793–1805
46 Chen W, Ling H, Gao J, et al. Learning to predict 3D objects with an interpolation-based differentiable renderer. In: Proceedings of Advances in Neural Information Processing Systems, 2019. 9609–9619
47 Rhodin H, Robertini N, Richardt C, et al. A versatile scene model with differentiable visibility applied to generative pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, 2015. 765–773
48 Li T M, Aittala M, Durand F, et al. Differentiable Monte Carlo ray tracing through edge sampling. ACM Trans Graph, 2019, 37: 1–11
49 Loubet G, Holzschuch N, Jakob W. Reparameterizing discontinuous integrands for differentiable rendering. ACM Trans Graph, 2019, 38: 1–14
50 Kato H, Beker D, Morariu M, et al. Differentiable rendering: a survey. 2020. ArXiv:2006.12057
51 Zhu X, Lei Z, Liu X, et al. Face alignment across large poses: a 3D solution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 146–155
52 Sander P V, Hoppe H, Snyder J, et al. Discontinuity edge overdraw. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics, Chapel Hill, 2001. 167–174
53 Kajiya J T. The rendering equation. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, 1986. 143–150
54 Zhang C, Miller B, Yan K, et al. Path-space differentiable rendering. ACM Trans Graph, 2020, 39: 143
55 Henderson P, Ferrari V. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. Int J Comput Vis, 2020, 128: 835–854

56 Ravi N, Reizenstein J, Novotny D, et al. Accelerating 3D deep learning with PyTorch3D. 2020. ArXiv:2007.08501
57 Valentin J, Keskin C, Pidlypenskyi P, et al. Tensorflow graphics: computer graphics meets deep learning. 2019. https://github.com/tensorflow/graphics
58 Zuffi S, Kanazawa A, Jacobs D W, et al. 3D menagerie: modeling the 3D shape and pose of animals. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 2017. 5524–5532
59 Poursaeed O, Kim V G, Shechtman E, et al. Neural puppet: generative layered cartoon characters. In: Proceedings of IEEE Winter Conference on Applications of Computer Vision, Snowmass Village, 2020. 3335–3345
60 Feng Q, Meng Y, Shan M, et al. Localization and mapping using instance-specific mesh models. In: Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, 2019. 4985–4991
61 Deschaintre V, Aittala M, Durand F, et al. Single-image SVBRDF capture with a rendering-aware deep network. ACM Trans Graph, 2018, 37: 1–15
62 Li Z, Sunkavalli K, Chandraker M. Materials for masses: SVBRDF acquisition with a single mobile phone image. In: Proceedings of the 15th European Conference on Computer Vision, Munich, 2018. 74–90
63 Blanz V, Vetter T. A morphable model for the synthesis of 3D faces. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, 1999. 187–194
64 Yu C, Wang J, Peng C, et al. BiseNet: bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the 15th European Conference on Computer Vision, Munich, 2018. 334–349
65 Lee C-H, Liu Z W, Wu L Y. MaskGAN: towards diverse and interactive facial image manipulation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020
66 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016. 770–778
67 Schroff F, Kalenichenko D, Philbin J. FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Boston, 2015. 815–823
68 Sariyanidi E, Zampella C J, Schultz R T, et al. Can facial pose and expression be separated with weak perspective camera? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 7173–7182

From traditional rendering to differentiable rendering: theories, methods and applications

Zipeng YE[1], Wenyu XIA[1], Zhiyao SUN[1], Ran YI[1], Minjing YU[2] & Yong-Jin LIU[1*]

1. *Department of Computer Science and Technology, Tsinghua University, Beijing* 100084*, China*;
2. *College of Intelligence and Computing, Tianjin University, Tianjin* 300350*, China*
* Corresponding author. E-mail: liuyongjin@tsinghua.edu.cn

Abstract In recent decades, with the development of computer hardware, the technologies of render engines in computer graphics and deep learning in artificial intelligence evolve rapidly. Differentiable rendering is the bridge of the two technologies, which also develops rapidly in recent years. Many methods of differentiable rendering are proposed and therefore provide new opportunities for next generation applications. In this survey paper, we first review the traditional render pipeline and then introduce the methods of differentiable rendering with both local illumination and global illumination. We list tools of differentiable rendering whose source codes are publicly available. We also collect the applications of differentiable rendering including those concerning the 3D models of face, body, hand and free-from object. At last, we point out some future research directions on differentiable rendering techniques.

Keywords differentiable rendering, inverse rendering, 3D reconstruction, face reconstruction, rendering

Zipeng YE is a Ph.D. student with Department of Computer Science and Technology, Tsinghua University. He received his B.E. degree from Tsinghua University, China, in 2017. His research interests include computational geometry and computer vision.

Wenyu XIA is an undergraduate student with Department of Computer science and Technology, Tsinghua University. She is expected to receive her B.E degree in 2021. Her research interests include computer graphics and data visualization.

Minjing YU is currently an assistant professor at College of Intelligence and Computing, Tianjin University, China. She received her B.E. degree from Wuhan University, China, in 2014 and her Ph.D. degree from Tsinghua University, in 2019. Her research interests include computer vision, cognitive computation and computer graphics.

Yong-Jin LIU is a professor with Department of Computer Science and Technology, Tsinghua University, China. He received his B.E. degree from Tianjin University, China, in 1998, and his Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2004. His research interests include computational geometry, computer vision and computer graphics.