

图像渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品

相关申请的交叉引用

本申请基于申请号为 202210202833.0、申请日为 2022 年 3 月 3 日的中国专利申请提出，并要求中国专利申请的优先权，中国专利申请的全部内容在此引入本申请作为参考。

5 技术领域

本申请涉及计算机图形图像技术，尤其涉及一种图像渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品。

背景技术

10 基于图形处理硬件的显示技术，扩展了感知环境以及获取信息的渠道，尤其是虚拟场景的显示技术，能够根据实际应用需求实现受控于用户或人工智能的虚拟对象之间的多样化的交互，具有各种典型的应用场景，例如在游戏等的虚拟场景中，能够模拟虚拟对象之间的真实的对战过程。

相关技术中，通常会对高动态范围的纹理资源进行压缩处理，从而减少纹理资源带来的带宽占用，但是基于压缩后的纹理资源进行渲染时难以达到较高的渲染效果，相关技术中尚无兼顾渲染资源以及渲染效果的方案。

15 发明内容

本申请实施例提供一种图像渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品，能够利用不同渲染图之间的损失，基于较少的带宽占用的纹理数据实现较高的渲染效果，从而提高了渲染资源利用率。

本申请实施例的技术方案是这样实现的：

20 本申请实施例提供一种图像渲染方法，包括：

获取虚拟对象的第一纹理数据、以及对应所述虚拟对象的第二纹理数据的转换参数，其中，所述第一纹理数据的数据量小于所述第二纹理数据的数据量，且所述第一纹理数据的图像信息范围小于所述第二纹理数据的图像信息范围；

25 基于所述转换参数以及所述第一纹理数据，进行拟合渲染处理，得到包括所述虚拟对象的拟合渲染图；

确定所述拟合渲染图与标准渲染图之间的渲染损失，并基于所述渲染损失更新所述转换参数以及所述第一纹理数据，其中，所述标准渲染图是基于所述第二纹理数据进行标准渲染处理得到的包括所述虚拟对象的渲染图；

30 基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标渲染图。

本申请实施例提供一种图像渲染装置，包括：

获取模块，配置为获取虚拟对象的第一纹理数据、以及对应所述虚拟对象的第二纹理数据的转换参数，其中，所述第一纹理数据的数据量小于所述第二纹理数据的数据量，且所述第一纹理数据的图像信息范围小于所述第二纹理数据的图像信息范围；

35 拟合模块，配置为基于所述转换参数以及所述第一纹理数据，进行拟合渲染处理，得到包括所述虚拟对象的拟合渲染图；

损失模块，配置为确定所述拟合渲染图与标准渲染图之间的渲染损失，并基于所述渲染损失更新所述转换参数以及所述第一纹理数据，其中，所述标准渲染图是基于所述第二纹理数据进行标准渲染处理得到的包括所述虚拟对象的渲染图；

40 渲染模块，配置为基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标渲染图。

本申请实施例提供一种电子设备，包括：

存储器，用于存储计算机可执行指令；

处理器，用于执行所述存储器中存储的计算机可执行指令时，实现本申请实施例提供的图像渲染方法。

本申请实施例提供一种计算机可读存储介质，存储有计算机可执行指令，用于被处理器执行时，实现本申请实施例提供的图像渲染方法。

5 本申请实施例提供一种计算机程序产品，包括计算机程序或计算机可执行指令，所述计算机程序或计算机可执行指令被处理器执行时实现本申请实施例提供的图像渲染方法。

本申请实施例具有以下有益效果：

通过对第一纹理数据以及第二纹理数据分别进行渲染处理，并基于渲染结果之间的损失更新第一纹理数据，并更新基于第一纹理数据进行渲染时所涉及到的转换参数，由于第二纹理数据的图像范围信息
10 优于第一纹理数据，且第一纹理数据的数据量小于第二纹理数据，从而基于更新得到的第一纹理数据以及转换参数进行实时图像渲染时，可以仅消耗较少的存储空间以及计算资源，达到对标第二纹理数据的渲染效果，进而有效提高渲染资源利用率。

附图说明

图 1A-图 1B 是本申请实施例提供的图像渲染系统的架构示意图；
15 图 2 是本申请实施例提供的用于图像渲染的电子设备的结构示意图；
图 3A-图 3C 是本申请实施例提供的图像渲染方法的流程示意图；
图 4 是本申请实施例提供的图像渲染方法的渲染示意图；
图 5 是本申请实施例提供的图像渲染方法的拟合渲染流程示意图；
图 6A-图 6E 是本申请实施例提供的图像渲染方法的动态范围转换示意图；
20 图 7A-图 7B 是本申请实施例提供的图像渲染方法的渲染示意图；
图 8 是本申请实施例提供的图像渲染方法的光栅化渲染流程示意图；
图 9 是本申请实施例提供的图像渲染方法的机器学习拟合场景示意图；
图 10 是本申请实施例提供的图像渲染方法的拟合纹理示意图；
图 11 是本申请实施例提供的图像渲染方法的机器学习拟合的流程示意图；
25 图 12 是本申请实施例提供的图像渲染方法的第一纹理数据的初始化值的示意图；
图 13 是本申请实施例提供的图像渲染方法的虚拟对象的屏幕空间渲染图；
图 14 是本申请实施例提供的图像渲染方法的虚拟对象的屏幕空间渲染图；
图 15 是本申请实施例提供的图像渲染方法的虚拟对象的拟合的屏幕空间渲染图；
图 16 是本申请实施例提供的图像渲染方法的标准渲染结果与拟合结果在屏幕空间的像素差异值的示
30 意图；
图 17 是本申请实施例提供的图像渲染方法的损失效果示意图；
图 18 是本申请实施例提供的图像渲染方法的动态范围空间转换函数的拟合示意图。

具体实施方式

为了使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请作进一步地详细描述，
35 所描述的实施例不应视为对本申请的限制，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本申请保护的范围。

在以下的描述中，所涉及的术语“第一\第二”仅仅是是区别类似的对象，不代表针对对象的特定排序，可以理解地，“第一\第二”在允许的情况下可以互换特定的顺序或先后次序，以使这里描述的本申请
40 实施例能够以除了在这里图示或描述的以外的顺序实施。

除非另有定义，本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述本申请实施例的目的，不是旨在限制本申请。

对本申请实施例进行进一步详细说明之前，对本申请实施例中涉及的名词和术语进行说明，本申请实施例中涉及的名词和术语适用于如下的解释。

1) 客户端：终端中运行的用于提供各种服务的应用程序，例如视频播放客户端、游戏客户端等。
45 2) 虚拟场景：游戏应用程序在终端上运行时显示(或提供)的虚拟游戏场景。该虚拟场景可以是对真实世界的仿真环境，也可以是半仿真半虚构的虚拟环境，还可以是纯虚构的虚拟环境。虚拟场景可以是二维虚拟场景、2.5 维虚拟场景或者三维虚拟场景中的任意一种，本申请实施例对虚拟场景的维度不加以限定。例如，虚拟场景可以包括天空、陆地、海洋等，该陆地可以包括沙漠、城市等环境元素，用户可以控制虚拟对象在该虚拟场景中进行移动。

3) 虚拟对象: 虚拟场景中可以进行交互的各种人和物的形象, 或在虚拟场景中的可活动对象。该可活动对象可以是虚拟人物、虚拟动物、动漫人物等, 例如在虚拟场景中显示的人物、动物等。该虚拟对象可以是虚拟场景中的一个虚拟的用于代表用户的虚拟形象。虚拟场景中可以包括多个虚拟对象, 每个虚拟对象在虚拟场景中具有自身的形状和体积, 占据虚拟场景中的一部分空间。

5 4) 图像渲染: 将三维的光能传递处理转换为一个二维图像的过程。场景和物体用三维形式表示, 更接近于现实世界, 便于操纵和变换, 而图像的显示设备大多是二维的光栅显示器和点阵化打印机。从三维场景的表示 N 维光栅和点阵化的表示就是图像渲染, 即光栅化。光栅显示器可以看作是一个像素矩阵, 在光栅显示器上显示的任何一个图像, 实际上都是一些具有一种或多种颜色和灰度像素的集合。

10 5) 色调映射 (Tone Mapping): 其作用是把高动态范围 HDR 的颜色映射为低动态范围 LDR 颜色, 以便显示器可以正常显示, 其对应的动态范围空间转换函数 (Inverse Tone Mapping) 的作用是把低动态范围 LDR 的颜色映射为高动态范围 HDR 颜色, 还原得到原始亮度范围的颜色。

6) 高动态范围 (HDR, High-Dynamic Range), 高动态范围的图像相比于普通图像可以提供更多的动态范围 (例如, 动态范围的宽度大于宽度阈值), 并具有更高的细节程度 (例如, 细节程度超过细节程度阈值), 根据不同的曝光时间的低动态范围 (LDR, Low-Dynamic Range) 的图像, 利用每个曝光时间相对应最佳细节的 LDR 图像来合成最终 HDR 图像, 能够更好的反映人真实环境中的视觉效果。

7) 低动态范围 (LDR, Low-Dynamic Range), 动态范围的宽度不大于宽度阈值, 并且细节程度不超过细节程度阈值, 称为低动态范围, 会导致高光或阴影中的细节丢失, 在摄影中, 动态范围以曝光值差异来衡量。

20 高动态范围渲染 (HDRR, High Dynamic Range Rendering) 与高动态范围贴图 (HDRT, High Dynamic Range Texture) 是游戏实时渲染中的重要技术, 高动态范围贴图是具有高动态范围的贴图, 两者结合所实现的渲染效果大幅超越低动态范围渲染 (LDRR, Low Dynamic Range Rendering) 与低动态范围贴图 (LDRT, Low Dynamic Range Texture) 两者结合所实现的渲染效果, 低动态范围贴图是具有低动态范围的贴图, 当游戏中的光照亮度范围比较大或者不同区域亮度差异明显时, 渲染效果的差异越发明显。申请人在实施本申请实施例时发现虽然 HDRR 和 HDRT 的结合能提供更为逼真的渲染效果, 但是 HDRR 和 HDRT 的结合所需要的渲染成本较高, 例如, 在未经压缩的 HDRT 中, 每个通道需要 32 个比特位, 从而每个像素 (包括 3 个通道) 需要 96 个比特位, 然而在未经压缩的 LDRT 中, 每个通道需要 8 个比特位, 压缩完成之后甚至可以更小。申请人在实施本申请实施例时发现 HDRR 和 HDRT 不仅会给游戏带来包体占用和内存占用负担, 在渲染的时候也会带来更多的带宽开销和计算负担, 所以针对各个游戏, 尤其是资源更为匮乏的手机端游戏, 均会对 HDRT 进行压缩, 并对 HDRR 进行计算优化, 目的是尽量减少 HDRT 带来的包体占用以及带宽占用, 减少 HDRR 的计算消耗, 并尽量逼近原始 HDRT 和 HDRR 提供的渲染效果。

30 相关技术中提供了一种将 HDRT 压缩到 LDRT 的方法, 通过计算 LDRT 和 HDRT 在贴图空间的差距值, 然后使用 Levenberg-Marquadt 算法来求得使差异值最小的色调映射函数的转换参数, 利用这个转换参数就可以把 HDRT 转换成 LDRT 进行打包封装。申请人在实施本申请实施例时发现相关技术中只考虑了贴图空间的纹理差异值, 对于渲染来说应该考虑渲染结果之间的差异值, 这样得到的结果会更为准确。

35 本申请实施例提供一种图像渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品, 能够利用不同渲染图之间的损失, 基于较少的带宽占用的纹理数据实现较高的渲染效果, 从而提高了渲染资源利用率。为便于更容易理解本申请实施例提供的图像渲染方法, 首先说明本申请实施例提供的图像渲染方法的示例性实施场景, 本申请实施例提供的图像渲染方法中的虚拟对象可以完全基于终端输出, 或者基于终端和服务端协同输出。

40 在一些实施例中, 虚拟场景可以是供游戏角色交互的环境, 例如可以是供游戏角色在虚拟场景中进行对战, 通过控制游戏角色的行动可以在虚拟场景中进行双方互动, 从而使用户能够在游戏的过程中舒缓生活压力。

45 在一个实施场景中, 参见图 1A, 图 1A 是本申请实施例提供的图像渲染系统的架构示意图, 适用于一些完全依赖于终端 400 的图形处理硬件计算能力即可完成虚拟场景 100 的相关数据计算的应用模式, 例如单机版/离线模式的游戏, 通过智能手机、平板电脑和虚拟现实/增强现实设备等各种不同类型的终端 400 完成虚拟场景的输出。

作为示例, 图形处理硬件的类型包括中央处理器 (CPU, Central Processing Unit) 和图形处理器 (GPU, Graphics Processing Unit)。

50 当形成虚拟场景 100 的视觉感知时, 终端 400 通过图形计算硬件计算显示所需要的数据, 并完成显示数据的加载、解析和渲染, 在图形输出硬件输出能够对虚拟场景形成视觉感知的视频帧, 例如, 在智能手机的显示屏呈现二维的视频帧, 或者, 在增强现实/虚拟现实眼镜的镜片上投射实现三维显示效果的视频帧; 此外, 为了丰富感知效果, 终端 400 还可以借助不同的硬件来形成听觉感知、触觉感知、运

动感知和味觉感知的一种或多种。

作为示例，终端 400 上运行有客户端 410（例如单机版的游戏应用），在客户端 410 的运行过程中输出包括有角色扮演的虚拟场景，虚拟场景可以是供游戏角色交互的环境，例如可以是用于供游戏角色进行对战的平原、街道、山谷等等；以第一人称视角显示虚拟场景 100 为例，在虚拟场景 100 中显示虚拟对象 401，虚拟对象 401 可以是受用户（或称玩家）控制的游戏角色，将响应于真实用户针对按钮（包括摇杆按钮、攻击按钮、防御按钮等）的操作而在虚拟场景中操作，例如当真实用户向左移动摇杆按钮时，虚拟对象将在虚拟场景中向左部移动，还可以保持原地静止、跳跃以及使用各种功能（如技能和道具）；虚拟对象 401 也可以是通过训练设置在虚拟场景对战中的人工智能（AI，Artificial Intelligence）；虚拟对象 401 还可以是设置在虚拟场景互动中的非用户角色（NPC，Non-Player Character）；虚拟对象 401 还可以是虚拟场景 100 中不可活动对象或者可活动对象。

举例来说，终端 400 为游戏开发人员所使用的终端，在游戏的开发打包阶段，终端 400 获取虚拟对象 401 的第一纹理数据、以及对应虚拟对象的第二纹理数据的转换参数，其中，第一纹理数据的数据量小于第二纹理数据的数据量，且第一纹理数据的图像信息范围小于第二纹理数据的图像信息范围；基于转换参数以及第一纹理数据，进行拟合渲染处理，得到包括虚拟对象的拟合渲染图；确定拟合渲染图与标准渲染图之间的渲染损失，并基于渲染损失更新转换参数以及第一纹理数据，其中，标准渲染图是基于第二纹理数据进行标准渲染处理得到的包括虚拟对象的渲染图，至此均是游戏开始前所执行的处理，在测试阶段，即游戏安装运行过程中，终端 400 基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标渲染图，并基于目标渲染图进行虚拟场景的人机交互，例如游戏对抗。

在另一个实施场景中，参见图 1B，图 1B 是本申请实施例提供的图像渲染系统的架构示意图，应用于终端 400 和服务端 200，适用于依赖于服务端 200 的计算能力完成虚拟场景计算、并在终端 400 输出虚拟场景的应用模式。

以形成虚拟场景 100 的视觉感知为例，服务端 200 进行虚拟场景相关显示数据（例如场景数据）的计算并通过网络 300 发送到终端 400，终端 400 依赖于图形计算硬件完成计算显示数据的加载、解析和渲染，依赖于图形输出硬件输出虚拟场景以形成视觉感知，例如可以在智能手机的显示屏幕呈现二维的视频帧，或者，在增强现实/虚拟现实眼镜的镜片上投射实现三维显示效果的视频帧；对于虚拟场景的形式感知而言，可以理解，可以借助于终端 400 的相应硬件输出，例如使用麦克风形成听觉感知，使用振荡器形成触觉感知等等。

作为示例，终端 400 上运行有客户端 410（例如网络版的游戏应用），通过连接服务端 200（例如游戏服务器）与其他用户进行游戏互动，终端 400 输出客户端 410 的虚拟场景 100，以第一人称视角显示虚拟场景 100 为例，在虚拟场景 100 中显示虚拟对象 401，虚拟对象 401 可以是受用户（或称玩家）控制的游戏角色，将响应于真实用户针对按钮（包括摇杆按钮、攻击按钮、防御按钮等）的操作而在虚拟场景中操作，例如当真实用户向左移动摇杆按钮时，虚拟对象将在虚拟场景中向左部移动，还可以保持原地静止、跳跃以及使用各种功能（如技能和道具）；虚拟对象 401 也可以是通过训练设置在虚拟场景对战中的人工智能（AI，Artificial Intelligence）；虚拟对象 401 还可以是设置在虚拟场景互动中的非用户角色（NPC，Non-Player Character）；虚拟对象 401 还可以是虚拟场景 100 中不可活动对象或者可活动对象。

举例来说，服务端 200 获取虚拟对象 401 的第一纹理数据、以及对应虚拟对象的第二纹理数据的转换参数，其中，第一纹理数据的数据量小于第二纹理数据的数据量，且第一纹理数据的图像信息范围小于第二纹理数据的图像信息范围；基于转换参数以及第一纹理数据，进行拟合渲染处理，得到包括虚拟对象的拟合渲染图；确定拟合渲染图与标准渲染图之间的渲染损失，并基于渲染损失更新转换参数以及第一纹理数据，其中，标准渲染图是基于第二纹理数据进行标准渲染处理得到的包括虚拟对象的渲染图，至此均是游戏开始前所执行的处理，例如，游戏的开发打包阶段，终端 400 接收由服务端 200 发送的更新的第一纹理数据以及更新的转换参数（可以封装在安装包中），在游戏安装运行过程中，终端 400 基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标渲染图，并基于目标渲染图进行虚拟场景的人机交互，例如游戏对抗。

在一些实施例中，终端 400 可以通过运行计算机程序来实现本申请实施例提供的图像渲染方法，例如，计算机程序可以是操作系统中的原生程序或软件模块；可以是本地（Native）应用程序（APP，Application），即需要在操作系统中安装才能运行的程序，例如换装游戏 APP（即上述的客户端 410）；也可以是小程序，即只需要下载到浏览器环境中就可以运行的程序；还可以是能够嵌入至任意 APP 中的游戏小程序。总而言之，上述计算机程序可以是任意形式的应用程序、模块或插件。

以计算机程序为应用程序为例，在实际实施时，终端 400 安装和运行有支持虚拟场景的应用程序。该应用程序可以是第一人称射击游戏（FPS，First-Person Shooting game）、第三人称射击游戏、虚拟现实应用程序、三维地图程序或者多人枪战类生存游戏中的任意一种。用户使用终端 400 操作位于虚拟场景

中的虚拟对象进行活动,该活动包括但不限于:调整身体姿态、爬行、步行、奔跑、骑行、跳跃、驾驶、拾取、射击、攻击、投掷、建造虚拟建筑中的至少一种。示意性的,该虚拟对象可以是虚拟人物,比如仿真人物角色或动漫人物角色等。

5 在一些实施例中,本申请实施例还可以借助于云技术(Cloud Technology)实现,云技术是指在广域网或局域网内将硬件、软件、网络等系列资源统一起来,实现数据的计算、储存、处理和共享的一种托管技术。

云技术是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台技术、以及应用技术等的总称,可以组成资源池,按需所用,灵活便利。云计算技术将变成重要支撑。技术网络系统的后台服务需要大量的计算、存储资源。

10 示例的,图1B中的服务器200可以是独立的物理服务器,也可以是多个物理服务器构成的服务器集群或者分布式系统,还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN、以及大数据和人工智能平台等基础云计算服务的云服务器。终端400可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表等,但并不局限于此。终端以及服务器可以通过有线或无线通信方式进行直接或间接地连接,本发明实施例中不做限制。

15 参见图2,图2是本申请实施例提供的用于图像渲染的电子设备的结构示意图,图2所示的终端400包括:至少一个处理器410、存储器450、至少一个网络接口420和用户接口430。终端400中的各个组件通过总线系统440耦合在一起。可理解,总线系统440用于实现这些组件之间的连接通信。总线系统440除包括数据总线之外,还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见,在图2中将各种总线都标为总线系统440。

20 处理器410可以是一种集成电路芯片,具有信号的处理能力,例如通用处理器、数字信号处理器(DSP, Digital Signal Processor),或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等,其中,通用处理器可以是微处理器或者任何常规的处理器等。

25 用户接口430包括使得能够呈现媒体内容的一个或多个输出装置431,包括一个或多个扬声器和/或一个或多个视觉显示屏。用户接口430还包括一个或多个输入装置432,包括有助于用户输入的用户接口部件,比如键盘、鼠标、麦克风、触屏显示屏、摄像头、其他输入按钮和控件。

存储器450可以是可移除的,不可移除的或其组合。示例性的硬件设备包括固态存储器,硬盘驱动器,光盘驱动器等。存储器450可选地包括在物理位置上远离处理器410的一个或多个存储设备。

30 存储器450包括易失性存储器或非易失性存储器,也可包括易失性和非易失性存储器两者。非易失性存储器可以是只读存储器(ROM, Read Only Memory),易失性存储器可以是随机存取存储器(RAM, Random Access Memory)。本申请实施例描述的存储器450旨在包括任意适合类型的存储器。

在一些实施例中,存储器450能够存储数据以支持各种操作,这些数据的示例包括程序、模块和数据结构或者其子集或超集,下面示例性说明。

操作系统451,包括用于处理各种基本系统服务和执行硬件相关任务的系统程序,例如框架层、核心库层、驱动层等,用于实现各种基础业务以及处理基于硬件的任务;

35 网络通信模块452,用于经由一个或多个(有线或无线)网络接口420到达其他电子设备,示例性的网络接口420包括:蓝牙、无线相容性认证(WiFi)、和通用串行总线(USB, Universal Serial Bus)等;

呈现模块453,用于经由一个或多个与用户接口430相关联的输出装置431(例如,显示屏、扬声器等)使得能够呈现信息(例如,用于操作外围设备和显示内容和信息的用户接口);

40 输入处理模块454,用于对一个或多个来自一个或多个输入装置432之一的一个或多个用户输入或互动进行检测以及翻译所检测的输入或互动。

在一些实施例中,本申请实施例提供的图像渲染装置可以采用软件方式实现,图2示出了存储在存储器450中的图像渲染装置455,其可以是程序和插件等形式的软件,包括以下软件模块:获取模块4551、拟合模块4552、损失模块4553以及渲染模块4554,这些模块是逻辑上的,因此根据所实现的功能可以进行任意的组合或进一步拆分。将在下文中说明各个模块的功能。

45 下面将结合附图对本申请实施例提供的图像渲染方法进行具体说明。本申请实施例提供的图像渲染方法可以由图1A中的终端400单独执行,也可以由图1B中的终端400和服务器200协同执行。

下面,以由图1A中的终端400单独执行本申请实施例提供的图像渲染方法为例进行说明。参见图3A,图3A是本申请实施例提供的图像渲染方法的流程图,将结合图3A示出的步骤101-步骤104进行说明。

50 需要说明的是,图3A示出的方法可以由终端400上运行的各种形式的计算机程序执行,并不局限于上述的客户端410,还可以是上文的操作系统451、软件模块和脚本,因此客户端不应视为对本申请实施例的限定。

在步骤 101 中, 获取虚拟对象的第一纹理数据、以及对应虚拟对象的第二纹理数据的转换参数。

作为示例, 转换参数指利用第二纹理数据进行渲染的过程中涉及到的参数, 在渲染之前虚拟对象处于待渲染的状态, 处于待渲染的状态的虚拟对象为基础对象模型, 其中, 基础对象模型包括虚拟对象的躯干, 不包括用于装饰虚拟对象的展示信息, 例如展示信息包括用于修饰虚拟对象面部的妆容 (例如唇形、眼影、瞳孔、虹膜、腮红等)、用于修饰虚拟对象的四肢的服饰 (例如古装、战斗服等) 等。

作为示例, 第一纹理数据可以是 LDRT 纹理, 第二纹理数据可以是 HDRT 纹理, LDRT 纹理包括每个纹理像素的第一颜色值, HDRT 纹理包括每个纹理像素的第二颜色值, 第一纹理数据的数据量小于第二纹理数据的数据量, 在 HDRT 中, 每个通道需要 32 个比特位, 从而每个像素 (包括 3 个通道) 需要 96 个比特位, 然而在 LDRT 中, 每个通道需要 8 个比特位, 从而每个像素 (包括 3 个通道) 需要 24 个比特位。第一纹理数据的图像信息范围小于第二纹理数据的图像信息范围, 图像信息范围包括动态范围, 动态范围是最高亮度与最低亮度的比值, HDRT 的动态范围高于 LDRT 的动态范围。

作为示例, 第一纹理数据可以是初始化的纹理图像中每个像素的像素参数, 包括材质参数以及纹理参数, 第一纹理数据还可以是通过迭代过程中第 n-1 次迭代得到的, 从而将第 n-1 次迭代得到的第一纹理数据作为第 n 次迭代的输入, n 为大于 1 的整数, 每次迭代均需要执行后续步骤 102 以及步骤 103。

需要说明的是, 当第一纹理数据是初始化的纹理图像中每个像素的像素参数时, 在步骤 101 之前, 还需要初始化渲染场景, 包括加载模型 (基础对象模型)、纹理、材质等等, 然后设置摄像机的位置、朝向, 设置模型的位置、朝向, 设置各种灯光的参数。

在步骤 102 中, 基于转换参数以及第一纹理数据, 进行拟合渲染处理, 得到包括虚拟对象的拟合渲染图。

在一些实施例中, 拟合渲染处理是对虚拟对象进行的, 基于转换参数以及第一纹理数据对虚拟对象进行拟合渲染处理, 得到包括虚拟对象的拟合渲染图。

需要说明的是, 在实际的游戏运行中, 虚拟对象不会一直处于同一个朝向, 所以需要拟合虚拟对象各个朝向的渲染效果, 这样才能保证虚拟对象各个朝向的拟合效果。以获取的第一纹理数据作为资源, 用不受光材质 (Unlit) 的渲染方式对第一纹理数据进行拟合渲染处理, 得到包括虚拟对象的拟合渲染图。

在一些实施例中, 参见图 3B, 图 3B 是本申请实施例提供的图像渲染方法的流程图, 步骤 102 中基于转换参数以及第一纹理数据, 进行拟合渲染处理, 得到包括虚拟对象的拟合渲染图, 可以通过图 3B 中的步骤 1021 至步骤 1022 实现。

在步骤 1021 中, 基于转换参数, 对第一纹理数据进行朝向第二纹理数据的空间转换处理, 得到与第二纹理数据处于相同动态范围空间的第三纹理数据。

在一些实施例中, 第一纹理数据包括每个纹理像素的第一颜色值, 第二纹理数据包括每个纹理像素的第二颜色值, 转换参数包括第一一阶参数以及第二一阶参数; 步骤 1021 中基于转换参数, 对虚拟对象的第一纹理数据进行朝向第二纹理数据的空间转换处理, 得到与第二纹理数据处于相同动态范围空间的第三纹理数据, 可以通过以下技术方案实现: 针对第一纹理数据中的每个纹理像素执行以下处理: 确定纹理像素的第一颜色值与第一一阶参数之间的第一乘积结果; 确定第一乘积结果与第二一阶参数的第一求和结果; 将纹理像素的第一颜色值与第一求和结果之间的比值, 确定为纹理像素的与第二颜色值处于相同动态范围空间的第三颜色值; 将多个纹理像素的第三颜色值, 组成第三纹理数据。通过空间转换处理, 能够将第一纹理数据映射至对应第二纹理数据的动态范围空间, 从而可以具有更宽的亮度范围, 因此可以实现更为优秀的渲染效果。

作为示例, 空间转换处理是通过一阶高动态范围空间转换函数实现的, 一阶高动态范围空间转换函数参见公式 (1):

$$c_H = \frac{c_L}{p c_L + q} \quad (1);$$

其中, p, q 均是一阶色调映射函数的需要拟合的参数, p 为第一一阶参数, q 为第二一阶参数, 不同的 p, q 对应不同的曲线, c_H 是第三颜色值, c_L 是第一颜色值, 图 6D 中示出了不同 p, q 对应的不同曲线, 曲线的横坐标是第一颜色值, 曲线的纵坐标是第三颜色值。

在一些实施例中, 第一纹理数据包括每个纹理像素的第一颜色值, 第二纹理数据包括每个纹理像素的第二颜色值, 转换参数包括第一二阶参数、第二二阶参数、第三二阶参数、第四二阶参数以及第五二阶参数; 步骤 1021 中基于转换参数, 对虚拟对象的第一纹理数据进行朝向第二纹理数据的空间转换处理, 得到第三纹理数据, 可以通过以下技术方案实现: 针对第一纹理数据中的每个纹理像素执行以下处理: 确定纹理像素的第一颜色值的平方与第一二阶参数之间的第二乘积结果、以及纹理像素的第一颜色值与第二二阶参数之间的第三乘积结果; 对第二乘积结果、第三乘积结果以及第三二阶参数进行求和处理, 得到第二求和结果; 确定纹理像素的第一颜色值与第四二阶参数之间的第四乘积结果; 对第二求和结果的平方根、第四乘积结果以及第三二阶参数的平方根进行求和处理, 得到第三求和结果; 确定纹理像素的第一颜色值与第五二阶参数的第四求和结果; 将第三求和结果与第四求和结果之间的比值, 确定为纹

理像素的与第二颜色值处于相同动态范围空间的第三颜色值；将多个纹理像素的第三颜色值，组成第三纹理数据。通过空间转换处理，能够将第一纹理数据映射至对应第二纹理数据的动态范围空间，从而可以具有更宽的亮度范围，因此可以实现更为优秀的渲染效果。

作为示例，空间转换处理是通过二阶高动态范围空间转换函数实现的，一阶高动态范围空间转换函数参见公式(2)：

$$c_H = \frac{-\sqrt{pc_L^2 + qc_L + r} + sc_L + \sqrt{r}}{c_L + t} \quad (2);$$

其中， p, q, r, s, t 均是二阶高动态范围空间转换函数的需要拟合的参数， p 为第一二阶参数， q 为第二二阶参数， r 是第三二阶参数， s 是第四二阶参数， t 是第五二阶参数，不同的 p, q, r, s, t 的组合对应不同的曲线， c_H 是第三颜色值， c_L 是第一颜色值，图 6E 中示出了不同 p, q, r, s, t 的组合对应的不同曲线，曲线的横坐标是第一颜色值，曲线的纵坐标是第三颜色值。

在步骤 1022 中，基于第三纹理数据，进行拟合渲染处理，得到包括虚拟对象的拟合渲染图。

在一些实施例中，第三纹理数据包括每个纹理像素的第三颜色值；步骤 1022 中基于第三纹理数据，进行拟合渲染处理，得到包括虚拟对象的拟合渲染图，可以通过以下技术方案实现：获取虚拟对象的二维纹理坐标；获取对应拟合渲染处理的可微分渲染框架；将二维纹理坐标以及每个纹理像素的第三颜色值在可微分渲染框架中进行正向传播，得到包括虚拟对象的拟合渲染图。可微分渲染框架是将基于硬件的渲染过程进行软件化封装得到的，其中基于硬件的渲染过程可以是无受光材质的渲染方式，由于进行软件化封装，从而可微可导，因而后续可以基于梯度进行反向传播。

作为示例，获取虚拟对象的二维纹理坐标，二维纹理坐标是预先给定的或者自动化生成的，二维纹理坐标可以是 UV2 坐标，所有的图像文件都是二维平面。水平方向是 U，垂直方向是 V，通过二维平面可以定位图像上任意一个像素，它定义了图像上每个点的位置信息，这些点与模型是相互联系的，以决定表面纹理贴图的位置，UV 坐标就是将图像上每个点精确对应到模型物体的表面，获取对应拟合渲染处理的可微分渲染框架，将二维纹理坐标以及每个纹理像素的第三颜色值在可微分渲染框架中进行正向传播，得到包括虚拟对象的拟合渲染图，参与正向传播的数据还包括材质、光照等渲染资源，参见图 8，图 8 是本申请实施例提供的光栅化渲染流程示意图，改造后的整个光栅化渲染流程通过统一电子设备架构（CUDA，Compute Unified Device Architecture）来实现，并且作为机器学习库（例如 PyTorch）的一个模块来运行，这里的 CUDA 即为上述可微分渲染框架，整个渲染过程是可以逆向传播梯度的，所以整个过程是可微的。将所有相关资源（Assets）通过可传递梯度的计算连接（Connection with Gradients），获取模型切线/法线数据（Compute Tangent/Normal），生成光照图纹理坐标（Lightmap UV Generation），计算动画和蒙皮（Animation & Skinning），将模型切线/法线数据、光照图纹理坐标、动画和蒙皮通过可传递梯度的计算连接（Connection with Gradients），获取裁剪空间中的坐标（Clip Space Pos），将所有相关资源（Assets）通过常规的计算连接（Ordinary Connection），获取三角形索引数据（Index Buffer），基于裁剪空间中的坐标以及三角形索引数据进行光栅化（Rasterization），分别得到坐标（ u, v ）以及 $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}$ ，基于坐标（ u, v ）、 $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}$ 以及顶点属性数据（Vertex Attributes）进行插值（Interpolation）处理，得到插值后的属性数据（Interpolate Attributes）以及像素级属性数据的导数（Attribute Pixel Derivatives），基于差值后的属性数据、像素级属性数据的导数以及纹理（Texture）进行纹理采样（Texture Sampling）处理，得到过滤后的采样数据（Filtered Samples），基于光照（Lights Camera）、材质参数（Material Parameters）、插值后的属性数据以及过滤后的采样数据进行待拟合的目标渲染（GroundTruth Shading），得到有锯齿的渲染图像（Aliased Image），对有锯齿的渲染图像进行抗锯齿（Antialiasing）处理，得到最终拟合渲染图（Final Image）。

在步骤 103 中，确定拟合渲染图与标准渲染图之间的渲染损失，并基于渲染损失更新转换参数以及第一纹理数据。

作为示例，标准渲染图中虚拟对象的朝向与迭代过程中拟合渲染时设置的朝向相同，标准渲染图是基于第二纹理数据进行标准渲染处理得到的包括虚拟对象的渲染图。标准渲染处理可以是基于物理的渲染处理，具体而言，基于虚拟对象的第二纹理数据，进行对应虚拟对象的基于物理的渲染处理，得到标准渲染图。例如，通过基于物理规律的渲染技术（PBR，Physically Based Rendering）加载虚拟对象的第二纹理数据，并基于物理规律对第二纹理数据进行渲染处理，得到符合物理规律的标准渲染图，这种渲染方式可以带来最优的渲染效果，但是代价是需要付出较大的存储资源以及计算资源。

在一些实施例中，参见图 3C，图 3C 是本申请实施例提供的图像渲染方法的流程示意图，步骤 103 中确定拟合渲染图与标准渲染图之间的渲染损失，可以通过图 3C 中的步骤 1031 至步骤 1032 实现。

在步骤 1031 中，确定标准渲染图与拟合渲染图在屏幕空间的整体像素值差异。

在一些实施例中，步骤 1031 中确定标准渲染图与拟合渲染图在屏幕空间的整体像素值差异，可以通

过以下技术方案实现：针对拟合渲染图以及标准渲染图在屏幕空间中任一相同像素执行以下处理：确定拟合渲染图中对应像素的第一像素值，并确定标准渲染图中的对应像素的第二像素值；将第一像素值与第二像素值之间的差值的绝对值作为对应像素的像素值差异；对与多个像素对应的像素值差异进行求和处理，得到整体像素值差异。以每个像素作为差异衡量的最小单位，可以有效提高渲染损失的价值，从而基于渲染损失进行更新时，可以得到具有更优渲染效果的转换参数以及第一纹理数据。

在一些实施例中，针对拟合渲染图以及标准渲染图在屏幕空间中每个相同像素执行以下处理：确定拟合渲染图中对应像素的第一像素值，并确定标准渲染图中的对应像素的第二像素值；将第一像素值与第二像素值之间的差值的绝对值作为对应像素的像素值差异；对与多个像素对应的像素值差异进行求和处理，得到整体像素值差异。

- 10 在步骤 1032 中，基于整体像素值差异、拟合渲染图的长度以及拟合渲染图的宽度，确定渲染损失。作为示例，渲染损失是基于公式 (3) 得到的：

$$L1(\text{Img1}, \text{Img2}) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W |\text{Img1}_{i,j} - \text{Img2}_{i,j}| \quad (3);$$

- 15 其中，Img1 和 Img2 分别表示标准渲染图与拟合渲染图，H 和 W 分别表示 Img1 (或 Img2) 的长宽， $\sum_{i=1}^H \sum_{j=1}^W |\text{Img1}_{i,j} - \text{Img2}_{i,j}|$ 表示标准渲染图与拟合渲染图在屏幕空间的像素值差异，(i,j) 表示标准渲染图中在屏幕空间中任一像素。

需要说明的是，本申请实施例的渲染损失并不局限于公式 (3)，还可以是其他的变形公式。

- 20 在一些实施例中，步骤 103 中于渲染损失更新转换参数以及第一纹理数据，可以通过以下技术方案实现：基于渲染损失对第一纹理数据进行偏导处理，得到对应第一纹理数据的梯度；基于渲染损失对转换参数进行偏导处理，得到对应转换参数的梯度；基于对应第一纹理数据的梯度，更新第一纹理数据，并基于对应转换参数的梯度，更新转换参数。

- 25 在一些实施例中，上述基于第一纹理数据的梯度，更新第一纹理数据，可以通过以下技术方案实现：将设定学习率与对应第一纹理数据的梯度相乘，得到第一纹理数据的数据变化值；将第一纹理数据的数据变化值与第一纹理数据相加，得到更新的第一纹理数据；上述基于对应转换参数的梯度，更新转换参数，可以通过以下技术方案实现：将设定学习率与对应转换参数的梯度相乘，得到转换参数的数据变化值；将转换参数的数据变化值与转换参数相加，得到更新的转换参数。

- 30 需要说明的是，基于渲染损失更新第一纹理数据的过程类似于机器学习的反向传播过程，将第一纹理数据输入到可微分光栅化渲染器，通过可微分光栅化渲染器的正向渲染过程，输出拟合渲染图，由于可微分光栅化渲染器的输出结果 (拟合渲染图) 与标准渲染图有误差 (渲染损失)，则计算输出结果与标准渲染图之间的误差，并基于误差进行反向传播，反向传播是基于梯度下降算法实现的，在反向传播的过程中，根据误差调整第一纹理数据以及转换参数，例如，求出渲染损失对第一纹理数据的偏导数，生成渲染损失对第一纹理数据的梯度，由于梯度的方向表明误差扩大的方向，采用梯度下降算法更新第一纹理数据，求出渲染损失对转换参数的偏导数，生成渲染损失对转换参数的梯度，由于梯度的方向表明误差扩大的方向，采用梯度下降算法更新转换参数，不断迭代上述过程，直至满足迭代结束条件。

- 35 在步骤 104 中，基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括虚拟对象的目标渲染图。

- 40 在一些实施例中，步骤 104 中基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，可以通过以下技术方案实现：当渲染损失小于损失阈值时，基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理；或者当更新次数达到更新次数阈值时，基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理。

- 45 作为示例，迭代结束条件包括以下至少之一：渲染损失函数的值小于损失阈值；迭代次数达到设定次数。当满足迭代结束条件时，将更新的转换参数以及更新的第一纹理数据作为最终保存的用于进行实施渲染处理的渲染基础资源，当不限定迭代次数时，通过设定损失阈值来界定目标渲染图与标准渲染图之间的相似程度，当损失阈值越小时，目标渲染图与标准渲染图之间的越相似；当不限定损失阈值时，通过设定迭代次数来界定目标渲染图与标准渲染图之间的相似程度，当设定的迭代次数越大时，目标渲染图与标准渲染图之间的越相似。通过损失阈值或者更新次数来判断是否能够基于当前更新的第一纹理数据以及转换参数进行实时渲染，可以提高拟合效率，防止浪费拟合资源。

- 50 在一些实施例中，步骤 104 中基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括虚拟对象的目标渲染图，可以通过以下技术方案实现：基于更新的转换参数，对更新的第一纹理数据进行朝向第二纹理数据的空间转换处理，得到与第二纹理数据处于相同动态范围空间的第四纹理数据；确定虚拟对象的至少一个二维纹理坐标；针对每个二维纹理坐标执行以下处理：从第四纹理数据中采样出与二维纹理坐标对应的纹理图像，并对采样得到的纹理图像进行贴合处理；基于每个二维纹理坐标的贴合结果，生成包括虚拟对象的目标渲染图。

作为示例,保存更新的第一纹理数据到纹理图像中,获得纹理图像后,在游戏过程中就可以利用该纹理图像进行实时图像渲染,避免加载多种纹理,节约了相关的存储空间以及计算资源,进而提高虚拟对象的渲染效率。

作为示例,获取虚拟对象的二维纹理坐标,二维纹理坐标是预先给定的或者自动化生成的,二维纹理坐标可以是 UV2 坐标,UV2 坐标定义了图像上每个点的位置信息,这些点与模型是相互联系的,以决定表面纹理贴图的位置,UV 坐标就是将图像上每个点精确对应到模型物体的表面,通过 UV2 来采样第四纹理数据,将采样得到的第四纹理数据构成的纹理贴图贴在虚拟对象的基础模型上,从而实现实时渲染过程。

通过本申请实施例对第一纹理数据以及第二纹理数据分别进行渲染处理,并基于渲染结果之间的损失更新第一纹理数据以及基于第一纹理数据进行渲染时所涉及到的转换参数,由于第二纹理数据的图像范围信息优于第一纹理数据,且第一纹理数据的数据量小于第二纹理数据,从而基于更新得到的第一纹理数据以及转换参数进行实时图像渲染时,可以达到对标第二纹理数据的渲染效果,仅消耗较少的存储空间以及计算资源,进而有效提高渲染资源利用率。

下面,将说明本申请实施例在一个实际的应用场景中的示例性应用。

本申请实施例可以应用于各种游戏的渲染场景,例如对抗游戏、赛车游戏、变装游戏等。在一些实施例中,服务器获取虚拟对象的第一纹理数据、以及对应虚拟对象的第二纹理数据的转换参数,其中,第一纹理数据的数据量小于第二纹理数据的数据量,且第一纹理数据的图像信息范围小于第二纹理数据的图像信息范围;基于转换参数以及第一纹理数据,进行拟合渲染处理,得到包括虚拟对象的拟合渲染图;确定拟合渲染图与标准渲染图之间的渲染损失,并基于渲染损失更新转换参数以及第一纹理数据,标准渲染图是基于第二纹理数据进行标准渲染处理得到的包括虚拟对象的渲染图,至此均是游戏开始前所执行的渲染处理,例如,游戏的开发打包阶段,终端接收由服务器发送的更新的第一纹理数据以及更新的转换参数(可以封装在安装包中),在游戏安装运行过程中,终端基于更新的转换参数以及更新的第一纹理数据,对所述虚拟对象进行实时渲染处理,得到包括所述虚拟对象的目标渲染图,并基于目标渲染图进行虚拟场景的人机交互,例如游戏对抗。需要说明的是,本申请实施例适用于大部分手游项目,尤其能满足对渲染性能和渲染效果都要求较高的手游项目,本方案能大幅提高渲染效率,并做到和原始效果差异很小。

参见图 4,图 4 是本申请实施例提供的渲染示意图,本申请实施例的游戏中的渲染效果只需要一张纹理和对应的 UV2 顶点数据即可,可以看作是将拟合出来的纹理 401 采用第 2 个纹理坐标(UV2)贴在模型 402(即待渲染的虚拟对象)上。

下面具体说明本申请实施例提供的基于可微分渲染的计算框架和机器学习拟合算法。参见图 5,图 5 是本申请实施例提供的拟合渲染的流程示意图,如图 5 所示,渲染过程是一个非常复杂的函数 f ,它的输入是 x , x 是参数集(待拟合参数),包含了诸如模型顶点位置、材质参数、纹理参数等数据, f 的输出 y 就是渲染结果。

本申请实施例不会去优化顶点位置等数据, x 中只有第一纹理数据以及转换参数需要进行优化,目标是在给定 HDRR 渲染结果的情况下,计算 x 来使得 $f(x)$ 尽量逼近 HDRR 渲染结果,逼近的衡量程度用损失函数(Loss)来计算。通过以下公式 $\frac{\partial y}{\partial x} = \frac{\partial}{\partial x} f(x)$ 计算拟合渲染图对待拟合参数的偏微分,通过以下公式 $\frac{\partial z}{\partial y} = \frac{\partial}{\partial y} Loss(y)$ 计算损失函数对拟合渲染图的偏微分,通过以下公式 $\frac{\partial z}{\partial x}$ 计算损失函数对待拟合参数的偏微分。

优化算法采用梯度下降算法。不过 f 是一个非常复杂的函数,正常的渲染流程是不可导的,所以为了用梯度下降算法来求最优的 x 需要 f 可导,本申请实施例提供一个可求导(即可微分)的光栅化渲染框架。需要说明的是, f 是完整的光栅化渲染流程,为了让其可微,本申请实施例对光栅化渲染进行了改进,参见图 8,改造后的整个光栅化渲染流程通过统一电子设备架构(CUDA, Compute Unified Device Architecture)来实现,并且作为机器学习库(例如 PyTorch)的一个模块来运行,整个渲染过程是可以逆向传播梯度的,所以整个过程是可微的。将所有相关资源(Assets)通过可传递梯度的计算连接(Connection with Gradients),获取模型切线/法线数据(Compute Tangent/Normal)、生成光照图纹理坐标(Lightmap UV Generation)、计算动画和蒙皮(Animation & Skinning),将模型切线/法线数据、光照图纹理坐标、动画和蒙皮通过可传递梯度的计算连接(Connection with Gradients),获取裁剪空间中的坐标(Clip Space Pos),将所有相关资源(Assets)通过常规的计算连接(Ordinary Connection),获取三角形索引数据(Index Buffer),基于裁剪空间中的坐标以及三角形索引数据进行光栅化(Rasterization),分别得到坐标 (u,v) 以及 $\frac{\partial u}{\partial x}$ 、 $\frac{\partial u}{\partial y}$ 、 $\frac{\partial v}{\partial x}$ 、 $\frac{\partial v}{\partial y}$,基于坐标 (u,v) 、 $\frac{\partial u}{\partial x}$ 、 $\frac{\partial u}{\partial y}$ 、 $\frac{\partial v}{\partial x}$ 、 $\frac{\partial v}{\partial y}$ 以及顶点属性数据(Vertex Attributes)进行插值(Interpolation)处理,得到插值后的属性数据(Interpolate Attributes)以及像素级属性数据的导数(Attribute Pixel Derivatives),

基于差值后的属性数据、像素级属性数据的导数以及纹理 (Texture) 进行纹理采样 (Texture Sampling) 处理, 得到过滤后的采样数据 (Filtered Samples), 基于光照 (Lights Camera)、材质参数 (Material Parameters)、插值后的属性数据以及过滤后的采样数据进行待拟合的目标渲染 (GroundTruth Shading), 得到有锯齿的渲染图像 (Aliased Image), 对有锯齿的渲染图像进行抗锯齿 (Antialiasing) 处理, 得到最终的渲染图像 (Final Image)。

下面具体说明本申请实施例提供的图像渲染方法中高动态范围空间转换函数的应用原理。为了减少包体占用以及内存占用, 最终存放的纹理数据是 LDRT, 存放范围是 [0-1] 之间的数据, 在渲染的时候会将其转换到高动态范围空间, 然后基于转换得到的数据进行渲染, 高动态范围空间的数据范围更广, 包含了更多的人类视觉系统能感知的数据, 这个转换的过程所使用的函数是高动态范围空间转换函数, 可以看作是色调映射的逆过程。

作为示例, 色调映射函数 Reinhard 可以参见公式 (4), 它的函数曲线如图 6A 中的虚线曲线所示:

$$c_L = \frac{c_H}{1.0 + c_H} \quad (4);$$

其中, c_H 是高动态范围空间的颜色值, c_L 是低动态范围空间的颜色值, 它相应的高动态范围空间转换函数 (Inverse Reinhard) 可以参见公式 (5), 它的函数曲线如图 6B 中的虚线曲线所示:

$$c_H = \frac{c_L}{1.0 - c_L} \quad (5);$$

作为示例, 色调映射函数 Aces 还可以参见公式 (6), 它的函数曲线如图 6A 中的实线曲线所示

$$c_L = \frac{c_H(2.51c_H + 0.03)}{c_H(2.43c_H + 0.59) + 0.14} \quad (6);$$

其中, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值, 它相应的高动态范围空间转换函数 (Inverse Aces) 可以参见公式 (7), 它的函数曲线如图 6B 中的实线曲线所示:

$$c_H = \frac{0.0062 - 0.1213c_L - 0.0021\sqrt{9 + 13702c_L - 10127c_L^2}}{c_L - 1.03292} \quad (7);$$

本申请实施例中针对不同 HDRR 的不同渲染效果均需要拟合出 LDRT, 使得基于 LDRT 进行不受光材质渲染之后逼近 HDRR 的渲染效果。对于不同的渲染效果来说, 它们的亮度范围其实是不一样的, 单一的高动态范围空间转换函数的参数均不是最优解, 不论是对应 Reinhard 的高动态范围空间转换函数还是对应 Aces 的高动态范围空间转换函数, 它们的参数都是固定的, 并不能很好的适配各种不同的渲染亮度范围。

参见图 7A-图 7B, 图 7A-图 7B 是本申请实施例提供的图像渲染方法的渲染示意图, 图 7A 的虚拟对象的颜色亮度范围更多的集中于暗部, 图 7B 的虚拟对象的颜色亮度范围更多的集中于亮部, 所以理想情况下它们的高动态范围空间转换函数的曲线应该是图 6C 所示, 图 7A 中虚拟对象应该使用图 6C 示出的虚线曲线, 即分配较多的纹理数据范围给较暗的颜色空间, 虚线曲线分配纹理数据范围 [0-0.9] 给亮度区间 [0-0.2], 并分配较少的纹理数据范围给较亮的颜色空间, 虚线曲线分配了纹理数据范围 [0.9-1.0] 给亮度区间 [0.2-1.6], 并且控制最大亮度为 1.6 左右。这样的数据分配可以让渲染结果较暗的区域实现较高的精度范围以及最佳拟合效果。图 7B 中虚拟对象应该使用图 6C 中的实线曲线, 即分配较多的纹理数据范围给较亮的颜色空间, 实线曲线分配了纹理数据范围 [0.44-1.0] 给亮度区间 [0.2-4.34], 分配较少的纹理数据范围给较暗的颜色空间, 实线曲线分配了纹理数据范围 [0-0.44] 给亮度区间 [0-0.2], 并且控制最大亮度为 4.34 左右。这样的数据分配可以让渲染结果较亮的区域实现较高的精度范围以及最佳拟合效果。

通过上述可知若想实现最佳拟合效果, 高动态范围空间转换函数的曲线都需要定制, 而且需要自动拟合得到, 申请人在实施本申请实施例时从色调映射函数 Reinhard 和色调映射函数 Aces 的公式中得到启发, 实现了如下两种形式的高动态范围空间转换函数的参数自动拟合, 并且本申请实施例提供的图像渲染方法不仅仅局限于下面形式的高动态范围空间转换函数, 其他游戏可以根据需求的不同和游戏计算量开销的不同进行个性化配置。

作为示例, 一阶色调映射函数如公式 (8) 所示:

$$c_L = \frac{ac_H + b}{cc_H + d} \quad (8);$$

其中, a, b, c, d 均是一阶色调映射函数的参数, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值。

作为示例, 一阶色调映射函数对应的一阶高动态范围空间转换函数如公式 (9) 所示:

$$c_H = \frac{ec_L + f}{gc_L + h} \quad (9);$$

其中, e, f, g, h 均是一阶高动态范围空间转换函数的参数, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值。

通过后续的机器学习拟合算法, 对于 HDRR 的每个渲染效果都会拟合出来最优的 e, f, g, h 参数, 并且, 由于 c_L 为 0 的时候 c_H 也为 0, 所以 f 参数恒为 0, 再进行化简可以得到公式 (10):

$$c_H = \frac{c_L}{pc_L + q} \quad (10);$$

其中, p, q 均是一阶高动态范围空间转换函数的需要拟合的参数, 不同的 p, q 对应不同的曲线, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值, 图 6D 中示出了不同 p, q 对应的不同曲线, 横坐标是原始的 HDR 颜色值, 纵坐标是 LDR 颜色值。

作为示例, 二阶色调映射函数如公式 (11) 所示:

$$c_L = \frac{ac_H^2 + bc_H + c}{dc_H^2 + ec_H + f} \quad (11);$$

其中, a, b, c, d, e, f 均是二阶色调映射函数的参数, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值。

作为示例, 二阶色调映射函数对应的二阶高动态范围空间转换函数如公式 (12) 所示:

$$c_H = \frac{-\sqrt{pc_L^2 + qc_L + r + sc_L + t}}{uc_L + v} \quad (12);$$

其中, p, q, r, s, t, u, v 均是二阶高动态范围空间转换函数的参数, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值。

通过后续的机器学习拟合算法, 对于 HDRR 的每个渲染效果都会拟合出来最优的 p, q, r, s, t, u, v 参数。另外因为 c_L 为 0 的时候 c_H 也为 0, 所以 u, v 参数恒为 0, 再进行化简可以得到公式 (13):

$$c_H = \frac{-\sqrt{pc_L^2 + qc_L + r + sc_L + t}}{c_L + t} \quad (13);$$

其中, p, q, r, s, t 均是二阶高动态范围空间转换函数的需要拟合的参数, 不同的 p, q, r, s, t 对应不同的曲线, c_H 是原始的 HDR 颜色值, c_L 是 LDR 颜色值, 图 6E 中示出了不同 p, q 对应的不同曲线, 横坐标是原始的 HDR 颜色值, 纵坐标是 LDR 颜色值。

二阶高动态范围空间转换函数能给出比一阶高动态范围空间转换函数更多的曲线形状, 支持更多的亮度范围的压缩方式, 一阶高动态范围空间转换函数所对应的计算量小于二阶高动态范围空间转换函数。

接下来会详细介绍如何使用可微渲染框架和机器学习算法来拟合出最优的转换参数以及第一纹理数据 (LDRT)。

如图 9 所示, 输入部分包括: 1) PBR 渲染的原始虚拟对象, 包括模型 901、材质球、光照 902 (可以包括平行光、点光、面光源等); 2) 摄像机 903, 在整个拟合过程中, 摄像机保持不动。

输出部分包括: 1) 自动化生成的模型 UV2, 用来映射纹理空间到模型渲染结果; 2) 一张基于 UV2 展开的纹理 1001, 如图 10 所示, 当拟合结束时这张纹理里面存放的就是烘焙出来的 LDRT 数据, 其中, 这张纹理的像素参数在下面描述中用 ϕ 来表示; 3) 拟合得到的最优的高动态范围空间转换函数 (Inverse Tone Mapping) 的参数。

如图 11 所示的完整的拟合过程, 在给定的输入输出条件下, 完整的拟合过程主要包含 8 个步骤, 下面将具体说明每个步骤:

在步骤 801 中, 初始化场景及各项参数。

在进行机器学习拟合之前, 先初始化场景, 包括加载模型、纹理、材质等等, 然后设置相机的位置、朝向 (不优化), 设置模型的位置、朝向, 设置各种灯光的参数 (不优化), 自动化生成的模型 UV2。另外需要初始化纹理图每个像素的数据 (第一纹理数据) 以及高动态范围空间转换函数 (Inverse Tone Mapping) 的参数 θ 。一般可以初始化为如图 12 所示的灰色常量。其中, 在机器学习拟合的过程中不优化相机的位置、朝向、灯光的参数 (不优化)。

在步骤 802 中, 设置角色的朝向, 具体而言, 在机器学习拟合的过程中设置虚拟对象 (即模型) 的朝向。

在实际的游戏运行中, 虚拟对象不会一直处于同一个朝向, 所以需要拟合角色各个朝向的渲染效果, 这样才能保证虚拟对象各个状态下的拟合效果都是比较准确的。如图 13 所示的虚拟对象的朝向设置实例, 虚拟对象 1301 的三个不同朝向, 在每一次迭代的时候都会随机设置一个虚拟对象的朝向。

在步骤 803 中, 进行标准渲染处理, 得到 HDRR 渲染结果。

HDRR 渲染结果是需要进行拟合的标准渲染效果, 需要首先用 HDRT 进行 PBR 渲染, 得到如图 14 所示的虚拟对象 1401 某个朝向的渲染图 (即标准渲染图)。

在步骤 804 中, 进行拟合渲染处理, 得到包括虚拟对象的拟合渲染图。

然后采用同样的虚拟对象朝向, 以第一纹理数据作为资源, 用不受光材质 (Unlit) 的渲染方式进行渲染, 得到如图 15 所示的相同朝向的渲染图 (即拟合渲染图)。

在步骤 805 中, 计算渲染损失 Loss。

为了让拟合渲染图逼近标准渲染图, 需要对拟合纹理进行修改, 首先计算拟合渲染图与标准渲染图之间差距, 图 16 为两者在屏幕空间的像素差异值, 本申请实施例采用 L1 Loss 函数, 如公式 (14) 所示:

$$L1(\text{Img1}, \text{Img2}) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W |\text{Img1}_{i,j} - \text{Img2}_{i,j}| \quad (14);$$

其中, Img1 和 Img2 分别表示标准渲染图与拟合渲染图, H 和 W 分别表示 Img1 (或 Img2) 的长宽, $\sum_{i=1}^H \sum_{j=1}^W |\text{Img1}_{i,j} - \text{Img2}_{i,j}|$ 表示标准渲染图与拟合渲染图在屏幕空间的像素值差异, (i, j) 表示标准渲染

图中在屏幕空间中任一像素。

在步骤 806 中,判断渲染损失是否小于阈值,如果小于,转入步骤 807 和步骤 808,如果不小于,转入步骤 809。

在步骤 807 中,计算第一纹理数据的梯度(Gradient)以及转换参数的梯度(Gradient)。

- 5 在计算 Loss 之后,就可以通过 PyTorch 框架以及可微分光栅化渲染器计算出渲染损失 Loss 对第一纹理数据的梯度以及对转换参数的梯度。

在步骤 808 中,用相应的梯度分别更新第一纹理数据以及转换参数,并继续执行步骤 802 至步骤 805。

- 10 在计算第一纹理数据的梯度后,使用 PyTorch 的优化器对第一纹理数据进行更新,在计算转换参数的梯度后,使用 PyTorch 的优化器对转换参数进行更新,然后转入步骤 802 以进入下一次迭代,不停地重复前面步骤 802-步骤 807 的迭代过程,第一纹理数据以及对转换参数就会逐渐收敛逼近到最佳值。

在步骤 809 中,将第一纹理数据以及转换参数的格式进行转换并输出。

当渲染损失 Loss 的值小于损失阈值时,表征拟合渲染图已经非常接近标准渲染图,即可以退出整个迭代过程,保存第一纹理数据以及转换参数,在游戏中就可以采用 UV2 来采样所得到的纹理图,以获取接近 HDRR 的渲染结果。

- 15 下面详细介绍步骤 804,它包括 2 个子步骤,解决了如何使用第一纹理数据来进行 Unlit 渲染,首先,使用公式(10)或者公式(13)将 LDR 空间的第一纹理数据转换到 HDR 空间,算法刚开始的时候,公式的参数是默认的初始值,后来在每次迭代的过程中,公式的参数(转换参数)均是基于前一次迭代更新得到的,在算法迭代的过程中通过 Gradient 逐步更新到最优的参数。接着,使用 UV2 纹理坐标将采样得到的 HDR 空间的纹理数据渲染到世界空间中得到最后的屏幕空间渲染结果。

- 20 参见图 17,图 17 是本申请实施例提供的图像渲染方法的损失效果示意图,1701 是标准渲染图,1702 是使用步骤 808 输出的第一纹理数据,然后通过拟合得到高动态范围空间转换函数进行渲染得到的目标渲染图,拟合得到高动态范围空间转换函数如图 18 所示,高动态范围空间转换函数中的转换参数是步骤 808 输出的,1703 是目标渲染图与标准渲染图两者之间的像素差异值,可以看出两者之间的像素差异值是很小的。

- 25 本申请实施例提供了一种图像渲染方法,能够全自动对 HDRR 这种渲染方式进行拟合,并且能够拟合得到 LDRT,以代替 HDRT 作为渲染基础,大大降低纹理数据量,能以很低的开销实现逼近 HDRR 的渲染效果,大幅提升了游戏的帧率,降低了对电量的消耗。

- 30 可以理解的是,在本申请实施例中,涉及到用户信息等相关的数据,当本申请实施例运用到具体产品或技术中时,需要获得用户许可或者同意,且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。

- 下面继续说明本申请实施例提供的图像渲染装置 455 的实施为软件模块的示例性结构,在一些实施例中,如图 2 所示,存储在存储器 450 的图像渲染装置 455 中的软件模块可以包括:获取模块 4551,配置为获取虚拟对象的第一纹理数据、以及对应虚拟对象的第二纹理数据的转换参数,其中,第一纹理数据的数据量小于第二纹理数据的数据量,且第一纹理数据的图像信息范围小于第二纹理数据的图像信息范围;拟合模块 4552,配置为基于转换参数以及第一纹理数据,进行拟合渲染处理,得到包括虚拟对象的拟合渲染图;损失模块 4553,配置为确定拟合渲染图与标准渲染图之间的渲染损失,并基于渲染损失更新转换参数以及第一纹理数据,其中,标准渲染图是基于第二纹理数据进行标准渲染处理得到的包括虚拟对象的渲染图;渲染模块 4554,配置为基于更新的转换参数以及更新的第一纹理数据,进行实时渲染处理,得到包括虚拟对象的目标渲染图。

- 40 在一些实施例中,拟合模块 4552,还配置为:基于转换参数,对虚拟对象的第一纹理数据进行朝向第二纹理数据的空间转换处理,得到第三纹理数据,其中,第三纹理数据与第二纹理数据处于相同动态范围空间;基于第三纹理数据,进行对应虚拟对象的拟合渲染处理,得到包括虚拟对象的拟合渲染图。

- 45 在一些实施例中,第一纹理数据包括每个纹理像素的第一颜色值,第二纹理数据包括每个纹理像素的第二颜色值,转换参数包括第一一阶参数以及第二一阶参数;拟合模块 4552,还配置为:针对第一纹理数据中的每个纹理像素执行以下处理:确定纹理像素的第一颜色值与第一一阶参数之间的第一乘积结果;确定第一乘积结果与第二一阶参数的第一求和结果;将纹理像素对应的第一颜色值与第一求和结果之间的比值,确定为纹理像素的与第二颜色值处于相同动态范围空间的第三颜色值;将多个纹理像素的第三颜色值,组成第三纹理数据。

- 50 在一些实施例中,第一纹理数据包括每个纹理像素的第一颜色值,第二纹理数据包括每个纹理像素的第二颜色值,转换参数包括第一二阶参数、第二二阶参数、第三二阶参数、第四二阶参数以及第五二阶参数;拟合模块 4552,还配置为:针对第一纹理数据中的每个纹理像素执行以下处理:确定纹理像素的第一颜色值的平方与第一二阶参数之间的第二乘积结果、以及纹理像素的第一颜色值与第二二阶参数之间的第三乘积结果;对第二乘积结果、第三乘积结果以及第三二阶参数进行求和处理,得到第二求和

结果；确定纹理像素的第一颜色值与第四二阶参数之间的第四乘积结果；对第二求和结果的平方根、第四乘积结果以及第三二阶参数的平方根进行求和处理，得到第三求和结果；确定纹理像素的第一颜色值与第五二阶参数的第四求和结果；将第三求和结果与第四求和结果之间的比值，确定为纹理像素的与第二颜色值处于相同动态范围空间的第三颜色值；将多个纹理像素的第三颜色值，组成第三纹理数据。

5 在一些实施例中，第三纹理数据包括每个纹理像素的第三颜色值；拟合模块 4552，还配置为：获取虚拟对象的二维纹理坐标；获取对应拟合渲染处理的可微分渲染框架；将二维纹理坐标以及每个纹理像素的第三颜色值在可微分渲染框架中进行正向传播，得到包括虚拟对象的拟合渲染图。

在一些实施例中，损失模块 4553，还配置为：确定标准渲染图与拟合渲染图在屏幕空间的整体像素值差异；基于整体像素值差异、拟合渲染图的长度以及拟合渲染图的宽度，确定渲染损失。

10 在一些实施例中，损失模块 4553，还配置为：针对拟合渲染图以及标准渲染图在屏幕空间中任一相同像素执行以下处理：确定拟合渲染图中对应像素的第一像素值，并确定标准渲染图中对应像素的第二像素值；将第一像素值与第二像素值之间的差值的绝对值作为像素的像素值差异；对屏幕空间中多个像素的像素值差异进行求和处理，得到整体像素值差异。

15 在一些实施例中，损失模块 4553，还配置为：基于渲染损失对第一纹理数据进行偏导处理，得到对应第一纹理数据的梯度；基于渲染损失对转换参数进行偏导处理，得到对应转换参数的梯度；基于对应第一纹理数据的梯度，更新第一纹理数据，并基于对应转换参数的梯度，更新转换参数。

20 在一些实施例中，损失模块 4553，还配置为：将设定学习率与对应第一纹理数据的梯度相乘，得到第一纹理数据的数据变化值；将第一纹理数据的数据变化值与第一纹理数据相加，得到更新的第一纹理数据；将设定学习率与对应转换参数的梯度相乘，得到转换参数的数据变化值；将转换参数的数据变化值与转换参数相加，得到更新的转换参数。

在一些实施例中，渲染模块 4554，还配置为：当渲染损失小于损失阈值时，基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理；或者当更新次数达到更新次数阈值时，基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理。

25 在一些实施例中，渲染模块 4554，还配置为：基于更新的转换参数，对更新的第一纹理数据进行朝向第二纹理数据的空间转换处理，得到第四纹理数据，其中，所述第四纹理数据与所述第二纹理数据处于相同动态空间范围；确定虚拟对象的至少一个二维纹理坐标；针对每个二维纹理坐标执行以下处理：从第四纹理数据中采样出与二维纹理坐标对应的纹理图像，并对采样得到的纹理图像进行贴合处理；基于每个二维纹理坐标的贴合结果，生成包括虚拟对象的目标渲染图。

30 本申请实施例提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括计算机指令，该计算机指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机指令，处理器执行该计算机指令，使得该计算机设备执行本申请实施例上述的图像渲染方法。

本申请实施例提供一种存储有可执行指令的计算机可读存储介质，其中存储有可执行指令，当可执行指令被处理器执行时，将引起处理器执行本申请实施例提供的图像渲染方法，例如，如图 3A-3C 示出的图像渲染方法。

35 在一些实施例中，计算机可读存储介质可以是 FRAM、ROM、PROM、EPROM、EEPROM、闪存、磁表面存储器、光盘、或 CD-ROM 等存储器；也可以是包括上述存储器之一或任意组合的各种设备。

在一些实施例中，可执行指令可以采用程序、软件、软件模块、脚本或代码的形式，按任意形式的编程语言（包括编译或解释语言，或者声明性或过程性语言）来编写，并且其可按任意形式部署，包括被部署为独立的程序或者被部署为模块、组件、子例程或者适合在计算环境中使用的其它单元。

40 作为示例，可执行指令可以但不一定对应于文件系统中的文件，可以可被存储在保存其它程序或数据的文件的一部分，例如，存储在超文本标记语言（HTML，Hyper Text Markup Language）文档中的一个或多个脚本中，存储在专用于所讨论的程序的单个文件中，或者，存储在多个协同文件（例如，存储一个或多个模块、子程序或代码部分的文件）中。

45 作为示例，可执行指令可被部署为在一个电子设备上执行，或者在位于一个地点的多个电子设备上执行，又或者，在分布在多个地点且通过通信网络互连的多个电子设备上执行。

综上所述，通过本申请实施例对第一纹理数据以及第二纹理数据分别进行渲染处理，并基于渲染结果之间的损失更新第一纹理数据以及基于第一纹理数据进行渲染时所涉及到的转换参数，由于第二纹理数据的图像范围信息优于第一纹理数据，且第一纹理数据的数据量由于第二纹理数据，从而基于更新得到的第一纹理数据以及转换参数进行实时图像渲染时，可以达到对标第二纹理数据的渲染效果，仅消耗较少的存储空间以及计算资源，进而有效提高渲染资源利用率。

50 以上所述，仅为本申请的实施例而已，并非用于限定本申请的保护范围。凡在本申请的精神和范围之内所作的任何修改、等同替换和改进等，均包含在本申请的保护范围之内。

权利要求书

- 1、一种图像渲染方法，所述方法由电子设备执行，所述方法包括：
获取虚拟对象的第一纹理数据、以及与所述虚拟对象的第二纹理数据对应的转换参数，其中，所述
第一纹理数据的数据量小于所述第二纹理数据的数据量，且所述第一纹理数据的图像信息范围小于所述
5 第二纹理数据的图像信息范围；
基于所述转换参数以及所述第一纹理数据，进行拟合渲染处理，得到包括所述虚拟对象的拟合渲染
图；
确定所述拟合渲染图与标准渲染图之间的渲染损失，并基于所述渲染损失更新所述转换参数以及所
述第一纹理数据，其中，所述标准渲染图是基于所述第二纹理数据进行标准渲染处理得到的包括所述虚
10 拟对象的渲染图；
基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标
渲染图。
- 2、根据权利要求1所述的方法，其中，所述基于所述转换参数以及所述第一纹理数据，进行拟合渲
染处理，得到包括所述虚拟对象的拟合渲染图，包括：
15 基于所述转换参数，对所述虚拟对象的第一纹理数据进行朝向所述第二纹理数据的空间转换处理，
得到第三纹理数据，其中，所述第三纹理数据与所述第二纹理数据处于相同动态范围空间；
基于所述第三纹理数据，进行拟合渲染处理，得到包括所述虚拟对象的拟合渲染图。
- 3、根据权利要求2所述的方法，其中，所述第一纹理数据包括每个纹理像素的第一颜色值，所述第
二纹理数据包括每个所述纹理像素的第二颜色值，所述转换参数包括第一一阶参数以及第二一阶参数；
20 所述基于所述转换参数，对所述虚拟对象的第一纹理数据进行朝向所述第二纹理数据的空间转换处
理，得到第三纹理数据，包括：
针对所述第一纹理数据中的每个所述纹理像素执行以下处理：
确定所述纹理像素的第一颜色值与所述第一一阶参数之间的第一乘积结果；
确定所述第一乘积结果与所述第二一阶参数的第一求和结果；
25 将所述纹理像素的第一颜色值与所述第一求和结果之间的比值，确定为所述纹理像素的与所述第二
颜色值处于相同动态范围空间的第三颜色值；
将多个所述纹理像素的第三颜色值，组成所述第三纹理数据。
- 4、根据权利要求2所述的方法，其中，所述第一纹理数据包括每个纹理像素的第一颜色值，所述第
二纹理数据包括每个所述纹理像素的第二颜色值，所述转换参数包括第一二阶参数、第二二阶参数、第
30 三二阶参数、第四二阶参数以及第五二阶参数；
所述基于所述转换参数，对所述虚拟对象的第一纹理数据进行朝向所述第二纹理数据的空间转换处
理，得到第三纹理数据，包括：
针对所述第一纹理数据中的每个所述纹理像素执行以下处理：
确定所述纹理像素的第一颜色值的平方与所述第一二阶参数之间的第二乘积结果、以及所述纹理像
35 素的第一颜色值与所述第二二阶参数之间的第三乘积结果；
对所述第二乘积结果、所述第三乘积结果以及所述第三二阶参数进行求和处理，得到第二求和结果；
确定所述纹理像素的第一颜色值与所述第四二阶参数之间的第四乘积结果；
对所述第二求和结果的平方根、所述第四乘积结果以及所述第三二阶参数的平方根进行求和处理，
得到第三求和结果；
40 确定所述纹理像素的第一颜色值与所述第五二阶参数的第四求和结果；
将所述第三求和结果与所述第四求和结果之间的比值，确定为所述纹理像素的与所述第二颜色值处
于相同动态范围空间的第三颜色值；
将多个所述纹理像素的第三颜色值，组成所述第三纹理数据。
- 5、根据权利要求2所述的方法，其中，所述第三纹理数据包括每个纹理像素的第三颜色值；
45 所述基于所述第三纹理数据，进行对应所述虚拟对象的拟合渲染处理，得到包括所述虚拟对象的拟
合渲染图，包括：
获取所述虚拟对象的二维纹理坐标；
获取对应所述拟合渲染处理的微分渲染框架；
将所述二维纹理坐标以及每个所述纹理像素的第三颜色值在所述微分渲染框架中进行正向传播，
50 得到包括所述虚拟对象的拟合渲染图。
- 6、根据权利要求1所述的方法，其中，所述确定所述拟合渲染图与标准渲染图之间的渲染损失，包
括：
确定所述标准渲染图与所述拟合渲染图在屏幕空间的整体像素值差异；

基于所述整体像素值差异、所述拟合渲染图的长度以及所述拟合渲染图的宽度，确定所述渲染损失。

7、根据权利要求6所述的方法，其中，所述确定所述标准渲染图与所述拟合渲染图在屏幕空间的整体像素值差异，包括：

针对所述拟合渲染图以及所述标准渲染图在所述屏幕空间中任一相同像素执行以下处理：

- 5 确定所述拟合渲染图中对应所述像素的第一像素值，并确定所述标准渲染图中对应所述像素的第二像素值；

将所述第一像素值与所述第二像素值之间的差值的绝对值作为所述像素的像素值差异；

对所述屏幕空间中多个所述像素的像素值差异进行求和处理，得到所述整体像素值差异。

- 8、根据权利要求1所述的方法，其中，所述基于所述渲染损失更新所述转换参数以及所述第一纹理数据，包括：

基于所述渲染损失对所述第一纹理数据进行偏导处理，得到对应所述第一纹理数据的梯度；

基于所述渲染损失对所述转换参数进行偏导处理，得到对应所述转换参数的梯度；

基于对应所述第一纹理数据的梯度，更新所述第一纹理数据，并基于对应所述转换参数的梯度，更新所述转换参数。

- 9、根据权利要求8所述的方法，其中，

所述基于对应所述第一纹理数据的梯度，更新所述第一纹理数据，包括：

将设定学习率与对应所述第一纹理数据的梯度相乘，得到所述第一纹理数据的数据变化值；

将所述第一纹理数据的数据变化值与所述第一纹理数据相加，得到更新的第一纹理数据；

所述基于对应所述转换参数的梯度，更新所述转换参数，包括：

- 20 将设定学习率与对应所述转换参数的梯度相乘，得到所述转换参数的数据变化值；

将所述转换参数的数据变化值与所述转换参数相加，得到更新的转换参数。

10、根据权利要求1所述的方法，其中，所述基于更新的转换参数以及更新的第一纹理数据，对所述虚拟对象进行实时渲染处理，包括：

- 25 当所述渲染损失小于损失阈值时，基于更新的转换参数以及更新的第一纹理数据，对所述虚拟对象进行实时渲染处理；或者

当更新次数达到更新次数阈值时，基于更新的转换参数以及更新的第一纹理数据，对所述虚拟对象进行实时渲染处理。

11、根据权利要求1所述的方法，其中，所述基于更新的转换参数以及更新的第一纹理数据，对所述虚拟对象进行实时渲染处理，得到包括所述虚拟对象的目标渲染图，包括：

- 30 基于所述更新的转换参数，对所述更新的第一纹理数据进行朝向所述第二纹理数据的空间转换处理，得到第四纹理数据，其中，所述第四纹理数据与所述第二纹理数据处于相同动态空间范围；

确定所述虚拟对象的至少一个二维纹理坐标；

针对每个所述二维纹理坐标执行以下处理：

- 35 从所述第四纹理数据中采样出与所述二维纹理坐标对应的纹理图像，并对采样得到的纹理图像进行贴合处理；

基于每个所述二维纹理坐标的贴合结果，生成包括所述虚拟对象的目标渲染图。

12、一种图像渲染装置，所述装置包括：

- 40 获取模块，配置为获取虚拟对象的第一纹理数据、以及对应所述虚拟对象的第二纹理数据的转换参数，其中，所述第一纹理数据的数据量小于所述第二纹理数据的数据量，且所述第一纹理数据的图像信息范围小于所述第二纹理数据的图像信息范围；

拟合模块，配置为基于所述转换参数以及所述第一纹理数据，进行拟合渲染处理，得到包括所述虚拟对象的拟合渲染图；

- 45 损失模块，配置为确定所述拟合渲染图与标准渲染图之间的渲染损失，并基于所述渲染损失更新所述转换参数以及所述第一纹理数据，其中，所述标准渲染图是基于所述第二纹理数据进行标准渲染处理得到的包括所述虚拟对象的渲染图；

渲染模块，配置为基于更新的转换参数以及更新的第一纹理数据，进行实时渲染处理，得到包括所述虚拟对象的目标渲染图。

13、一种电子设备，所述电子设备包括：

存储器，用于存储计算机可执行指令；

- 50 处理器，用于执行所述存储器中存储的计算机可执行指令时，实现权利要求1至11任一项所述的图像渲染方法。

14、一种计算机可读存储介质，存储有计算机可执行指令，所述计算机可执行指令被处理器执行时实现权利要求1至11任一项所述的图像渲染方法。

15、一种计算机程序产品，包括计算机程序或计算机可执行指令，所述计算机程序或计算机可执行指令被处理器执行时实现权利要求 1 至 11 任一项所述的图像渲染方法。