(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 107833262 B
(45)授权公告日 2023.06.23

(21)申请号 201710790958.9

(22)申请日 2017.09.05

(65)同一申请的已公布的文献号
申请公布号 CN 107833262 A

(43)申请公布日 2018.03.23

(30)优先权数据
1615012.0 2016.09.05 GB

(73)专利权人 ARM 有限公司
地址 英国剑桥

(72)发明人 塞缪尔·马丁

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
专利代理师 李晓冬

(51)Int.Cl.
G06T 15/00(2011.01)
G06T 1/20(2006.01)

(56)对比文件
US 2004233219 A1,2004.11.25

审查员 孟子山

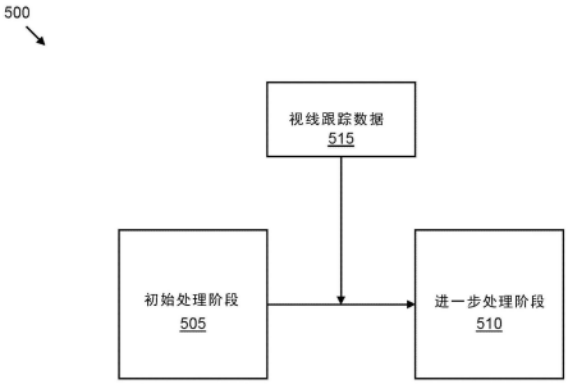
权利要求书3页 说明书15页 附图8页

(54)发明名称

图形处理系统和图形处理器

(57)摘要

本申请公开了图形处理系统和图形处理器。图形处理系统包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线。在初始处理阶段以第一分辨率处理场景的数据并以第二分辨率处理场景的数据。在初始处理阶段以第一分辨率和第二分辨率处理场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据。根据视线跟踪数据识别场景的至少一个子区域。在进一步处理阶段以第一分辨率处理场景的数据并以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据。通过组合在进一步处理阶段处理的场景的数据来渲染场景。



CN 107833262 B

1.一种操作图形处理系统的方法,该系统包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线,所述方法包括:

在所述初始处理阶段以第一分辨率处理场景的数据;

在所述初始处理阶段以第二分辨率处理所述场景的数据,其中所述初始处理阶段包括一个或多个几何操作;

在所述初始处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之后,接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据;

根据所接收的视线跟踪数据识别所述场景的至少一个子区域;

在所述进一步处理阶段以所述第一分辨率处理所述场景的数据;

在所述进一步处理阶段以所述第二分辨率仅处理与所识别的所述场景的至少一个子区域相对应的数据,其中所述进一步处理阶段包括一个或多个像素处理操作;以及

通过组合在所述进一步处理阶段以所述第一分辨率处理的所述场景的数据和在所述进一步处理阶段以所述第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

2. 如权利要求1所述的方法,其中所述图形处理流水线包括在所述初始处理阶段和所述进一步处理阶段之间的中间处理阶段,所述方法包括:

在所述中间处理阶段以所述第一分辨率处理所述场景的数据;以及

在所述中间处理阶段以所述第二分辨率仅处理与所识别的所述场景的至少一个子区域相对应的数据;

其中所述视线跟踪数据在所述中间处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之前被接收,并且与在所述中间处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之前所述虚拟现实用户设备的用户的至少一只眼睛的当前位置有关。

3. 如权利要求1所述的方法,其中所述图形处理流水线包括在所述初始处理阶段和所述进一步处理阶段之间的中间处理阶段,所述方法包括:

在所述中间处理阶段以所述第一分辨率处理所述场景的数据;以及

在所述中间处理阶段以所述第二分辨率处理所述场景的数据;

其中所述视线跟踪数据在所述中间处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之后被接收,并且与在所述中间处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之后所述虚拟现实用户设备的用户的至少一只眼睛的当前定位有关。

4.如权利要求1所述的方法,其中所述进一步处理阶段的处理具有比所述初始处理阶段的处理更大的计算负担。

5. 如权利要求1所述的方法,其中所述第一分辨率与所述第二分辨率相比相对较高。

6.如权利要求1所述的方法,其中所接收的视线跟踪数据与所述虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点有关;以及

其中所识别的所述场景的至少一个子区域与以所述虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点为中心的子区域有关。

7.如权利要求1所述的方法,其中所述场景的数据包括与左眼视图相关联的数据和与

右眼视图相关联的数据，并且所述图形处理流水线的每个阶段的数据处理包括处理与所述左眼视图相关联的数据和处理与所述右眼视图相关联的数据；以及

其中所接收的视线跟踪数据与所述虚拟现实用户设备的用户的左眼的中央凹的当前定位以及所述虚拟现实用户设备的用户的右眼的中央凹的当前定位有关。

8.如权利要求1所述的方法，包括：

在所述初始处理阶段以第三分辨率处理所述场景的数据；

其中所述视线跟踪数据在所述初始处理阶段以所述第三分辨率处理所述场景的数据之后被接收，并且与在所述初始处理阶段以所述第三分辨率处理所述场景的数据之后所述虚拟现实用户设备的用户的至少一只眼睛的当前定位有关，所述方法包括：

在所述进一步处理阶段以所述第三分辨率处理所述场景的数据；以及

在所述进一步处理阶段以所述第三分辨率仅处理与所述场景的另一子区域相对应的数据；

其中渲染所述场景包括进一步组合在所述进一步处理阶段以所述第三分辨率处理的所述场景的数据。

9.一种包括适配为执行操作图形处理系统的方法的软件代码的非暂态计算机可读存储介质，所述系统包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线，所述方法包括：

在所述初始处理阶段以第一分辨率处理场景的数据；

在所述初始处理阶段以第二分辨率处理所述场景的数据，其中所述初始处理阶段包括一个或多个几何操作；

在所述初始处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据；

根据所接收的视线跟踪数据识别所述场景的至少一个子区域；

在所述进一步处理阶段以所述第一分辨率处理所述场景的数据；

在所述进一步处理阶段以所述第二分辨率仅处理与所识别的所述场景的至少一个子区域相对应的数据，其中所述进一步处理阶段包括一个或多个像素处理操作；以及

通过组合在所述进一步处理阶段以所述第一分辨率处理的所述场景的数据和在所述进一步处理阶段以所述第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

10.一种包括图形处理流水线的图形处理系统，该图形处理流水线至少包括初始处理阶段和进一步处理阶段，所述图形处理系统被布置为：

在所述初始处理阶段以第一分辨率处理场景的数据；

在所述初始处理阶段以第二分辨率处理所述场景的数据，其中所述初始处理阶段包括一个或多个几何操作；

在所述初始处理阶段以所述第一分辨率和所述第二分辨率处理所述场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据；

根据所接收的视线跟踪数据识别所述场景的至少一个子区域；

在所述进一步处理阶段以所述第一分辨率处理所述场景的数据；

在所述进一步处理阶段以所述第二分辨率仅处理与所识别的所述场景的至少一个子

区域相对应的数据,其中所述进一步处理阶段包括一个或多个像素处理操作;以及

通过组合在所述进一步处理阶段以所述第一分辨率处理的所述场景的数据和在所述进一步处理阶段以所述第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

图形处理系统和图形处理器

技术领域

[0001] 本公开涉及图形处理系统和图形处理器。

背景技术

[0002] 注视点渲染(foveated rendering)是一种渲染技术,其中被显示的图像的一部分以较高的分辨率被渲染,并且该图像的一个或多个其他部分以较低的分辨率被渲染。这是因为用户直接看到的图像的部分可能需要以较高的分辨率被渲染以便视觉上可接受,而用户不直接看到的图像的外围区域可以以较低的分辨率被渲染但仍然看起来是视觉上可接受的。通过以较低分辨率渲染图像的外围区域,而不是以最高分辨率渲染整个图像,注视点渲染可以用于减小图形处理单元(GPU)上的渲染负担。

[0003] 注视点渲染可以包括识别一个或多个注视点(fixation point),其中在该一个或多个注视点处将渲染图像的较高分辨率版本,而远离该一个或多个注视点的区域以较低分辨率被渲染。具有不同的方式来确定图像的最高分辨率区域(即,一个或多个注视点)的位置。例如,可以使用头部跟踪或眼部跟踪系统来尝试识别图像中用户正在观看的地方。

[0004] 注视点渲染的一种使用是当为虚拟现实(VR)显示器渲染图像时,例如为虚拟现实头戴式显示器(VR HMD)渲染图像时。高分辨率VR HMD可能使用具有严重桶形畸变的镜头。这样做的效果是,朝向每只眼睛的显示器的中心的渲染图像被放大,而外围区域都被缩小。因此,外围区域可能以比中央放大区域更低的质量被渲染,但不会对用户的整体视觉效果造成任何显著损失。

[0005] 然而,仍然存在例如在执行注视点渲染时改进图形处理器和图形处理系统的操作的空间。

发明内容

[0006] 根据本公开的第一方面,提供了一种操作图形处理系统的方法,该系统包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线,所述方法包括:

[0007] 在初始处理阶段以第一分辨率处理场景的数据;

[0008] 在初始处理阶段以第二分辨率处理场景的数据;

[0009] 在初始处理阶段以第一分辨率和第二分辨率处理所述场景的数据之后,接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据;

[0010] 根据所接收的视线跟踪数据识别场景的至少一个子区域;

[0011] 在进一步处理阶段以第一分辨率处理场景的数据;

[0012] 在进一步处理阶段以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据;以及

[0013] 通过组合在进一步处理阶段以第一分辨率处理的场景的数据和在进一步处理阶段以第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

[0014] 根据本公开的第二方面，提供了一种操作包括图形处理流水线的图形处理系统的方法，所述方法包括在图形处理流水线中使用后期锁定的眼部跟踪数据来识别，与场景的一个或多个其他区域的分辨率相比，以相对高的分辨率被渲染的场景的一个或多个区域。

[0015] 根据本公开的第三方面，提供了一种操作图形处理系统的方法，该系统包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线，所述方法包括：

[0016] 在第一时间将用于图形处理工作项的输入数据提供给图形处理流水线；以及

[0017] 当图形处理流水线正在处理图形处理工作项时，在第二稍后时间将与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据提供给图形处理流水线。

[0018] 根据本公开的第四方面，提供了一种操作图形处理器的方法，该图形处理器与至少包含初始处理阶段和进一步处理阶段的图形处理流水线相关联，所述方法包括：

[0019] 在初始处理阶段以第一分辨率处理场景的数据；

[0020] 在初始处理阶段以第二分辨率处理场景的数据；

[0021] 在初始处理阶段以第一分辨率和第二分辨率处理场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据；

[0022] 根据所接收的视线跟踪数据识别场景的至少一个子区域；

[0023] 在进一步处理阶段以第一分辨率处理所述场景的数据；

[0024] 在进一步处理阶段以第二分辨率仅处理与所识别的所述场景的至少一个子区域相对应的数据；以及

[0025] 通过组合在进一步处理阶段以第一分辨率处理的所述场景的数据和在进一步处理阶段以第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

[0026] 根据本公开的第五方面，提供了一种计算机程序，该计算机程序包括当程序在图形处理系统中运行时适于执行一个或多个此种方法的软件代码。

[0027] 根据本公开的第六方面，提供了一种包括图形处理流水线的图形处理系统，该图形处理流水线至少包括初始处理阶段和进一步处理阶段，所述图形处理系统被布置成：

[0028] 在初始处理阶段以第一分辨率处理场景的数据；

[0029] 在初始处理阶段以第二分辨率处理场景的数据；

[0030] 在初始处理阶段以第一分辨率和第二分辨率处理场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据；

[0031] 根据所接收的视线跟踪数据识别场景的至少一个子区域；

[0032] 在进一步处理阶段以第一分辨率处理场景的数据；

[0033] 在进一步处理阶段以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据；以及

[0034] 通过组合在进一步处理阶段以第一分辨率处理的场景的数据和在进一步处理阶段以第二分辨率处理的与所识别的场景的至少一个子区域相对应的数据来渲染场景。

[0035] 根据本公开的第七方面，提供了一种包括图形处理流水线的图形处理系统，所述图形处理系统被配置成在图形处理流水线中使用后期锁定的眼部跟踪数据来识别，与场景的一个或多个其他区域的分辨率相比，以相对高的分辨率被渲染的场景的一个或多个区域。

[0036] 根据本公开的第八方面，提供了一种包括至少包含初始处理阶段和进一步处理阶段的图形处理流水线的图形处理系统，所述图形处理系统被布置成：

[0037] 在第一时间将用于图形处理工作项的输入数据提供给图形处理流水线；以及

[0038] 当图形处理流水线正在处理图形处理工作项时，在第二稍后时间将与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据提供给图形处理流水线。

[0039] 根据本公开的第九方面，提供了一种包括一个或多个此种图形处理系统的虚拟现实用户设备。

[0040] 根据本公开的第十方面，提供了一种图形处理器，该图形处理器与至少包含初始处理阶段和进一步处理阶段的图形处理流水线相关联，所述图形处理器被布置成：

[0041] 在初始处理阶段以第一分辨率处理场景的数据；

[0042] 在初始处理阶段以第二分辨率处理场景的数据；

[0043] 在初始处理阶段以第一分辨率和第二分辨率处理场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据；

[0044] 根据所接收的视线跟踪数据识别场景的至少一个子区域；

[0045] 在进一步处理阶段以第一分辨率处理场景的数据；

[0046] 在进一步处理阶段以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据；以及

[0047] 通过组合在进一步处理阶段以第一分辨率处理的场景的数据和在进一步处理阶段以第二分辨率处理的与所识别的场景的至少一个子区域相对应的数据来渲染所述场景。

[0048] 根据本公开的第十一方面，提供了一种包括此种图形处理器的虚拟现实用户设备。

[0049] 根据本公开的第十二方面，提供了基于本公开的第一方面的操作图形处理系统的方法，其中图形处理流水线包括在初始处理阶段和进一步处理阶段之间的中间处理阶段，该方法包括：

[0050] 在中间处理阶段以第一分辨率处理场景的数据；以及

[0051] 在中间处理阶段以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据；

[0052] 其中视线跟踪数据在中间处理阶段以第一分辨率和第二分辨率处理场景的数据之前被接收，并且与在中间处理阶段以第一分辨率和第二分辨率处理场景的数据之前虚拟现实用户设备的用户的至少一只眼睛的当前定位有关。

[0053] 根据本公开的第十三方面，提供了基于本公开的第一方面的操作图形处理系统的方法，其中图形处理流水线包括在初始处理阶段和进一步处理阶段之间的中间处理阶段，该方法包括：

[0054] 在中间处理阶段以第一分辨率处理场景的数据；以及

[0055] 在中间处理阶段以第二分辨率处理场景的数据；

[0056] 其中视线跟踪数据在中间处理阶段以第一分辨率和第二分辨率处理完场景的数据之后被接收，并且与在中间处理阶段以第一分辨率和第二分辨率处理场景的数据之后虚拟现实用户设备的用户的至少一只眼睛的当前定位有关。

[0057] 根据本公开的第十四方面，其中进一步处理阶段的处理具有比中间处理阶段的处

理更大的计算负担。

[0058] 根据本公开的第十五方面，其中中间处理阶段包括一个或多个几何操作。

[0059] 根据本公开的第十六方面，其中进一步处理阶段的处理具有比初始处理阶段的处理更大的计算负担。

[0060] 根据本公开的第十七方面，其中进一步处理阶段包括一个或多个像素处理操作。

[0061] 根据本公开的第十八方面，其中初始处理阶段包括一个或多个数据缓冲操作。

[0062] 根据本公开的第十九方面，其中在图形处理流水线的每个阶段中处理场景的数据包括在图形处理流水线的每个阶段中处理场景的数据帧。

[0063] 根据本公开的第二十方面，其中第一分辨率与第二分辨率相比相对较高。

[0064] 根据本公开的第二十一方面，其中第一分辨率和第二分辨率具有预定的关系。

[0065] 根据本公开的第二十二方面，其中该预定的关系包括2的幂。

[0066] 根据本公开的第二十三方面，其中该预定的关系包括1/4。

[0067] 根据本公开的第二十四方面，其中所接收的视线跟踪数据与虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点有关；以及

[0068] 其中所识别的场景的至少一个子区域与以虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点为中心的子区域有关。

[0069] 根据本公开的二十五方面，其中场景的数据包括与左眼视图相关联的数据和与右眼视图相关联的数据，并且图形处理流水线的每个阶段的数据处理包括处理与左眼视图相关联的数据和处理与右眼视图相关联的数据；并且

[0070] 其中所接收的视线跟踪数据与虚拟现实用户设备的用户的左眼的中央凹的当前定位以及虚拟现实用户设备的用户的右眼的中央凹的当前定位有关。

[0071] 根据本公开的二十六方面，其中所识别的场景的至少一个子区域与以虚拟现实用户设备的用户的左眼的中央凹的当前定位为中心的左眼子区域以及以虚拟现实用户设备的用户的右眼的中央凹的当前定位为中心的右眼子区域有关，并且

[0072] 其中渲染场景包括：

[0073] 组合在进一步处理阶段以第一分辨率处理的场景的左眼数据和在进一步处理阶段以第二分辨率处理的与所识别的场景的左眼子区域相对应的数据；以及

[0074] 组合在进一步处理阶段以第一分辨率处理的场景的右眼数据和在进一步处理阶段以第二分辨率处理的与所识别的场景的右眼子区域相对应的数据。

[0075] 根据本公开的第二十七方面，其中所识别的左眼子区域的至少一部分与所识别的右眼子区域的至少一部分不同。

[0076] 根据本公开的第二十八方面，提供了基于以上本公开的第一方面和第十二方面至第二十七方面中任一方面的操作图形处理系统的方法，该方法包括：

[0077] 在初始处理阶段以第三分辨率处理场景的数据；

[0078] 其中视线跟踪数据在初始处理阶段以第三分辨率处理场景的数据之后被接收，并且与在初始处理阶段以第三分辨率处理场景的数据之后虚拟现实用户设备的用户的至少一只眼睛的当前定位有关，该方法还包括：

[0079] 在进一步处理阶段以第三分辨率处理场景的数据；以及

[0080] 在进一步处理阶段以第三分辨率仅处理与场景的另一子区域相对应的数据；

[0081] 其中渲染该场景包括进一步组合在进一步处理阶段以第三分辨率处理的场景的数据。

[0082] 根据本公开的第二十九方面，提供了基于本公开的第二十八方面的操作图像处理系统的方法，该方法包括根据所接收的视线跟踪数据识别场景的另一子区域。

[0083] 根据本公开的第三十方面，提供了基于本公开的第二十八方面的操作图像处理系统的方法，该方法包括基于所识别的场景的至少一个子区域计算场景的另一子区域。

[0084] 根据本公开的第三十一方面，提供了一种本文中参考附图大体上描述的方法。

[0085] 根据本公开的第三十二方面，提供了一种本文中参考附图大体上描述的图形处理系统。

[0086] 根据本公开的第三十三方面，提供了一种本文参考附图大体上描述的图形处理器。

[0087] 根据本公开的第三十四方面，提供了一种本文参考附图大体上描述的图形处理流水线。

[0088] 根据下面参考附图仅通过示例方式给出的描述，其他特征和优点将变得显而易见。

附图说明

[0089] 图1示意性示出了根据本公开的实施例的图形处理系统的示例；

[0090] 图2示意性示出了根据本公开的实施例的图形处理流水线的示例；

[0091] 图3示意性示出了根据本公开的实施例的图形处理流水线的另一示例；

[0092] 图4示意性示出了根据本公开的实施例的帧流水线的示例；

[0093] 图5示意性示出了根据本公开的实施例的图形处理流水线的另一示例；

[0094] 图6示意性示出了根据本公开的实施例的图形处理流水线的另一示例；

[0095] 图7示意性示出了根据本公开的实施例的图形处理流水线的另一示例；和

[0096] 图8示意性示出了图形处理流水线的另一示例。

具体实施方式

[0097] 现在将在处理用于显示的计算机图形的上下文中描述根据本公开的实施例的多个示例。

[0098] 参考图1，其示意性示出了图形处理系统100的示例。

[0099] 在该示例中，在主处理器110上执行的应用程序105(诸如游戏)请求要由相关联的GPU 115执行的图形处理操作。为此，应用程序105生成应用程序编程接口(API)调用，其由驱动器120解析以用于GPU 115。驱动器120在主处理器110上运行。驱动器120生成适当的命令给GPU 115，以生成应用程序105所请求的图形输出。响应于来自应用程序105的命令，一组命令被提供给GPU 115。该命令可以用于生成要在显示器上显示的帧。

[0100] 参考图2，其示意性示出了图形处理流水线200的示例。图形处理流水线200指示可由GPU执行的一系列动作。

[0101] 在该示例中，GPU是基于拼块(tile-based)的渲染器。因此，GPU产生要生成的渲染输出数据数组的拼块。渲染输出数据数组可以是输出帧。基于拼块的渲染与即时模式渲染

不同，因为整个渲染输出不是被一次性处理，而是渲染输出被分成多个较小的子区域(或“区”)。这些子区域在本文中被称为拼块。每个拼块被单独渲染。例如，每个拼块可以被一个接一个地渲染。然后经渲染的图块被重新组合以提供用于显示的完整渲染输出。在基于拼块的渲染中，渲染输出可以被分成规则大小和形状的拼块。拼块可以是正方形或另一种形状。

[0102] 可以用于“平铺(tiling)”和“基于拼块的”渲染的其他术语包括“组块”，其中渲染拼块被称为“组块”和“桶状”渲染。为了方便，在下文中将使用术语“拼块”和“平铺”，但是应当理解，这些术语旨在包括所有替代和等同的术语和技术。

[0103] 渲染输出数据数组可以是用于在显示设备(诸如屏幕或打印机)上显示的输出帧。渲染输出也可以例如包括旨在用于稍后的渲染通道(rendering passes)的中间数据。其一个示例是“渲染到纹理”输出。

[0104] 当计算机图形图像要被显示时，它可以首先被定义为一组几何图形，例如一系列图元(primitives)。图元的一个示例是多边形。然后几何图形在光栅化过程中被分割成图形片元(fragment)。这之后是图形渲染。在图形渲染操作期间，渲染器可以修改与每个片元相关联的数据，使得片元能够被正确地显示。此种数据的示例包括颜色和透明度。一旦片元完全遍历渲染器，则其相关的数据值被存储在存储器中，准备输出。

[0105] 图2示出了与GPU相关联的各种元件和流水线阶段，该各种元件和流水线阶段与本文所述的实施例的操作相关。然而，还可以存在未在图2中示出的图形处理流水线的其他元件和阶段。还应该注意的是，图2仅是示意性的，并且例如，在实践中，所示出的功能单元和流水线阶段可以共享重要的硬件电路，尽管它们在图2中被示意性地示出为分开的阶段。还将理解，图形处理流水线200的每个阶段、元件和单元等可以根据需要被实现，并且因此将包括例如用于执行相关联的操作和功能的合适电路和/或处理逻辑等。

[0106] 如图2所示，GPU执行的图形处理流水线220包括多个阶段，包括顶点着色器205、外壳着色器210、镶嵌器215、域着色器220、几何着色器225、平铺器230、光栅化阶段235、早期Z(或“深度”)和模板测试阶段240、片元着色阶段形式的渲染器245、后期Z(或“深度”)和模板测试阶段250、混合阶段255，拼块缓冲器260和降采样和回写阶段265。然而，GPU的其它布置也是可能的。

[0107] 顶点着色器205接收与为要生成的输出定义的顶点相关联的输入数据值。顶点着色器205处理这些数据值以生成一组相应的顶点着色的输出数据值，以供图形处理流水线200的后续阶段使用。

[0108] 要处理的每个图元可以由一组顶点定义和表示。图元的每个顶点可以具有与其相关联的一组属性。一组属性是针对顶点的一组数据值。这些属性可以包括位置数据和其他非位置数据(或“变化”)。非位置数据可以定义例如所讨论的顶点的颜色、光、简正和/或纹理坐标。

[0109] 为要由GPU生成的给定输出定义一组顶点。针对该输出要被处理的图元包括该组顶点中的给定顶点。顶点着色操作将每个顶点的属性转换为后续图形处理操作所需的形式。这可以包括例如将顶点位置属性从最初所定义用于的世界或用户空间转换到其中将要显示图形处理系统的输出的屏幕空间。这还可以包括例如修改输入数据以考虑要渲染的图像中的光照的效果。

[0110] 外壳着色器210对多组贴片(patch)控制点执行操作，并生成被称为贴片常量的附加数据。

[0111] 镶嵌阶段215将几何形状细分以产生外壳的高阶表示。

[0112] 域着色器220以与顶点着色器205类似的方式对镶嵌阶段输出的顶点执行操作。

[0113] 几何着色器225处理全部图元，例如三角形、点或线。

[0114] 顶点着色器205、外壳着色器210、镶嵌器215、域着色器220和几何着色器225，响应于提供给GPU的命令和顶点数据，执行片元前端操作(诸如转换和光照操作)以及图元设置，来设置要被渲染的图元。

[0115] 一旦要被渲染的所有图元已经被适当地设置，则平铺器230确定对于为了处理的目的而将渲染输出分割得到的每个拼块而言有哪些图元要被处理。为此，平铺器230将要处理的每个图元的位置与拼块位置进行比较，并且将图元添加到确定图元可能落入其内的每个拼块的相应图元列表。用于将图元排序和合并(binning)到拼块列表中的任意合适和期望的技术(诸如精确合并，或边界框合并或介于之间的任意合并)，可以用于平铺过程。

[0116] 一旦以这种方式为每个渲染拼块准备了要被渲染的图元的列表(或“图元列表”)，则图元列表被存储以供使用。当所考虑的拼块被渲染时，图元列表允许系统识别哪些图元要被考虑和渲染。

[0117] 一旦平铺器(tiler)230已经准备好所有拼块列表，则每个拼块可以被渲染。为此，每个拼块由平铺器230之后的图形处理流水线阶段处理。

[0118] 当给定的拼块正在被处理时，针对该拼块要被处理的每个图元被传递到光栅器235。图形处理流水线200的光栅化阶段235操作以将图元光栅化成单独的图形片元以进行处理。为此，光栅器235将图元光栅化为采样点，并生成具有用于渲染图元的合适位置的图形片元。然后由光栅器235生成的片元被向前发送到流水线200的其余部分以进行处理。

[0119] 早期Z和模板测试阶段240对从光栅器235接收的片元执行Z(或“深度”)测试，以查看在该阶段是否能够丢弃(或“剔除”)任意片元。为此，早期Z和模板测试阶段240将由光栅器235发布的片元的深度值与已经被渲染的片元的深度值进行比较。已经被渲染的片元的深度值被存储在作为拼块缓冲器260的一部分的深度缓冲器中。由早期Z和模板测试阶段240执行的比较用于确定新的片元是否将被已经被渲染的片元遮挡。同时，进行早期模板测试。通过早期Z和模板测试阶段240的片元被发送到片元着色阶段245。片元着色阶段245对通过早期Z和模板测试阶段240的片元执行合适的片元处理操作以生成经适当渲染的片元数据。该片元处理可以包括任意合适的片元着色过程，诸如在片元上执行片元着色程序生成合适的片元数据，将纹理应用到片元，对片元应用起雾或其他操作等等。片元着色阶段245可以是可编程片元着色器。

[0120] 然后是后期片元Z和模板测试阶段250，其对经着色的片元执行流水线末端的深度测试等，以确定在最终的图像中是否将实际看到经渲染的片元。该深度测试使用针对存储在拼块缓冲器260中的Z缓冲器中的片元的位置的Z缓冲器值，来确定新的片元的片元数据是否应该代替已经被渲染的片元的片元数据。这可以包括将由片元着色阶段245发布的片元的深度值与存储在Z缓冲器中的已经被渲染的片元的深度值进行比较。该后期片元深度和模板测试阶段250还可以对片元执行后期阿尔法和/或模板测试。

[0121] 然后，通过后期片元测试阶段250的片元在混合器255中被进行与已被存储在拼块

缓冲器260中的片元的任意混合操作。所需要对片元进行的任何其他剩余操作(例如抖动等(未示出))也在该阶段进行。

[0122] 最后,输出片元数据(或“值”)被写入到拼块缓冲器260。然后可以将输出片元数据输出到帧缓冲器270进行显示。输出片元的深度值还适当地被写入到拼块缓冲器260内的Z缓冲器。拼块缓冲器260存储颜色和深度缓冲器,颜色和深度缓冲器分别存储针对缓冲器所表示的每个采样点的合适的颜色等或Z值。这些缓冲器存储片元数据的数组,该片元数据数组表示整个渲染输出的一部分(在该示例中是整个渲染输出中的拼块),其中缓冲器中的采样值的各个集合对应于整个渲染输出的各个像素。例如,采样值的每个2×2集合可以对应于输出像素,其中使用4x多采样。

[0123] 拼块缓冲器260被提供作为图形处理流水线200的本地随机存取存储器(RAM)的一部分。换句话说,拼块缓冲器260被提供在片上存储器中。

[0124] 来自拼块缓冲器260的数据被输入到降采样写出单元265,并且然后输出(或“回写”)到外部存储器输出缓冲器,例如显示设备(未示出)的帧缓冲器270。显示设备可以包括例如包括像素阵列的显示器,诸如计算机监视器或打印机。

[0125] 降采样写出单元265将存储在拼块缓冲器260中的片元数据降采样到用于输出缓冲器和设备的合适分辨率,以便生成与输出设备的像素对应的像素数据数组。这产生用于输出到输出缓冲器270的像素形式的输出值。

[0126] 一旦渲染输出的拼块已被处理并且其数据被导出到进行存储的主存储器,例如导出到主存储器中的帧缓冲器270,则下一个拼块被处理,以此类推,直到足够的拼块被处理以生成整个渲染输出。然后,为下一个渲染输出重复该过程,依此类推。

[0127] 从图2可以看出,图形处理流水线200包括多个可编程处理或“着色器”阶段,即顶点着色器205、外壳着色器210、域着色器220、着色器225和片元着色器245。这些可编程着色器阶段执行具有一个或多个输入变量并生成输出变量集合的各个着色器程序。可以针对要处理的每个工作项执行所讨论的着色器程序,例如在顶点着色器205的情况下针对每个顶点。可以针对要处理的每个工作项发布执行线程,并且然后线程执行着色器程序中的指令以产生所需的着色的输出数据。

[0128] 应用程序(诸如上面参照图1所述的应用程序105)提供要执行的着色器程序,该着色器程序使用高级着色器编程语言,例如OpenGL®着色语言(GLSL)、高级着色语言(HLSL)、开放计算语言(OpenCL)等。然后,这些着色器程序由着色器语言编译器转换成用于目标图形处理流水线200的二进制代码。这可以包括在编译器内创建程序的一个或多个内部中间表示。例如,编译器可以是驱动器120的一部分,其中存在使编译器运行的专门API调用。因此,编译器执行能够被看作是驱动器响应于应用程序105生成的API调用所作出的绘图调用准备的一部分。

[0129] 现在回到注视点渲染,视线跟踪(也被称为眼部跟踪)有很大可能会影响VR的若干区域,包括注视点渲染。

[0130] 人眼的大部分受体在中央凹(fovea)处,该中央凹是眼睛负责敏锐中央视力的区域。与全视野相比,中央凹较小。对于诸如VR系统中所使用的近眼显示而言,眼睛在任何时刻可能只能感知显示器上的信息的子集。视线跟踪的注视点渲染将渲染适配为在中央凹能够看到的显示区域中保持完整的计算,并减少其他地方的计算。

[0131] 参考图3,其示意性示出了图形处理流水线300的示例。图形处理流水线300被包括在图形处理系统中。图形处理流水线300至少包括初始处理阶段305和进一步处理阶段310。图形处理流水线300可以用于执行视线跟踪的注视点渲染。

[0132] 在初始处理阶段305或进一步处理阶段310中的任何数据处理之前接收视线跟踪数据315。视线跟踪数据315涉及虚拟现实用户设备的用户的至少一只眼睛的定位。根据接收的视线跟踪数据识别场景的至少一个子区域。所识别的场景的至少一个子区域可以例如对应于场景中用户正在观看的一个或多个区域。在初始处理阶段305和/或进一步处理阶段310以第一分辨率处理场景的数据。然而,在初始处理阶段305和/或进一步处理阶段310以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据。与第二分辨率相比,第一分辨率可以相对较高。然后,可以通过组合进一步处理阶段310所输出的数据来渲染场景。因此,可以在第一分辨率低于第二分辨率的情况下提供视线跟踪的注视点渲染,因为视线跟踪数据315被用来影响图形流水线300中的处理。

[0133] 虽然为视线跟踪的注视点渲染提供了此示例,但视线跟踪的注视点渲染的成功取决于各种因素。一个此种因素是眼部跟踪数据315的质量。另一个此种因素是由眼部跟踪器设备获得的眼睛跟踪信息被传递给渲染系统并由其使用的响应延迟。眼部跟踪数据的准确性越差(由于延迟使得结果不够实时和/或来自较低质量的跟踪数据),注视点渲染的结果越不成功。

[0134] 在一些示例中,鉴于延迟,GPU具有两帧或三帧流水线,这取决于例如几何形状是否在初始帧中被提交,或几何形状是否被延缓到下一帧的开始。GPU可以在延缓模式下正常操作,在这种情况下,流水线将是三帧。

[0135] 参考图4,其示意性示出了帧流水线400的示例。在该示例中,GPU具有三帧流水线,其中中央处理单元(CPU)缓冲405在帧0中执行,几何阶段410在帧1中执行,并且像素阶段415在帧2中执行。按照60Hz的显示速率,这在新数据通过图形API被提供给图形流水线的时间和结果被准备好输出到例如显示器的时间之间产生50ms的延迟。真正的延迟可能显著地大于这个时间,因为它包括图像传感器捕获、处理和传输。

[0136] 在针对VR系统假定目标感知阈值为20ms的运动光子延迟的情况下,超过50ms的延迟就用户体验而言,可能太大了。

[0137] 再次回到注视点渲染,在VR上下文中渲染内容包括将场景绘制到多个视图,其中每只眼睛至少有一个视图。扩展的OVR_multiview族(以下称为"多视图")可以用于OpenGL®嵌入式系统(OpenGL®ES),以产生场景的多个视图。原则上,多视图可以用于处理使用四个视图的注视点渲染,即较高分辨率的左眼视图、较高分辨率的右眼视图、较低分辨率的左眼视图和较低分辨率的右眼视图。例如,多视图可以用于同时渲染四个视图;每只眼睛的内部和外部视图。

[0138] 然而,当前版本的多视图中的所有视图必须具有相同的分辨率。当能够为内部和外部区域找到单个值时,这例如对于非视线跟踪实现方式可能是可接受的。然而,当前版本的多视图不允许使用用于视线跟踪实现方式中的紧密聚焦的中央凹区域。

[0139] 此外,当前版本的多视图只允许多达四个视图。可能可以扩展能够与新的多视图扩展结合使用的视图的数目,但是可能附加的视图也将受制于单一分辨率规则。

[0140] 此外,每个视图的配置(包括与焦点位置相关联的配置)由图形流水线开始处的应

用程序提供。因此，该配置将具有至少与图形流水线一样长的延迟。因此，在上述示例中，在跟踪观看者的眼睛位置和具有相关联的图形输出之间的延迟将总是至少50ms。考虑到此延迟，图形流水线可以采取保守方法来确定输出的高分辨率区域的大小。例如，图形流水线可以使用中心在图形流水线开始处的观看者的焦点位置的相对较大的区域，期望当渲染完成时，观看者可能正在观看该相对较大的区域中的某处，并因此可能正在观看输出的高分辨率部分。然而，这种保守方法涉及以较高分辨率渲染场景中的相对较大的部分，因为当输出被显示时，不确定观看者将会在看哪里。

[0141] 期望提供与多视图兼容的措施，但是该措施扩展多视图以允许渲染到可以低延迟地改变的中央凹区域和/或允许中央凹区域的分辨率与多视图分辨率不同。

[0142] 现在将描述图形处理器被用来通过以不同分辨率渲染表示场景的同一视图的多个图像来执行注视点渲染的示例。不同分辨率的视图然后被组合以提供输出中央凹图像，在该图像上分辨率是变化的。在这些示例中，上述流水线延迟和以不同分辨率进行渲染的开销也可以被减少。

[0143] 在这些示例中，眼部跟踪数据在图形流水线开始之后被使用，例如在图形流水线的中间。如此，与在图形流水线开始处使用的眼部跟踪数据相比，能够使用更加当前(即最近接收或获得的)的眼部跟踪数据。通过减少在渲染流水线中使用眼部跟踪数据的延迟，可以增加注视点渲染的优势。此外，以本文所述的方式实现此类措施可以容易地与现有架构集成，从而得到特别高效的实现方式。

[0144] 参考图5，其示意性示出了图形处理流水线500的示例。

[0145] 图形处理流水线500被包括在图形处理系统中。图形处理流水线500至少包括初始处理阶段505和进一步处理阶段510。进一步处理阶段510的处理可能具有比初始处理阶段505的处理更大的计算负担。

[0146] 在一些示例中，进一步处理阶段510包括一个或多个像素处理操作，例如光栅化和/或片元着色；并且初始处理阶段505包括一个或多个几何操作，例如顶点着色。在一些示例中，进一步处理阶段510包括一个或多个像素处理操作，例如光栅化和/或片元着色，并且初始处理阶段505包括一个或多个数据缓冲操作，例如CPU缓冲。在一些示例中，进一步处理阶段510包括一个或多个几何操作，例如片元着色，并且初始处理阶段505包括一个或多个数据缓冲操作，例如CPU缓冲。

[0147] 在初始处理阶段505以第一分辨率处理场景的数据并且以第二分辨率处理该场景的数据。在初始处理阶段505以第一分辨率和第二分辨率处理完该场景的数据之后，接收与虚拟现实用户设备的用户的至少一只眼睛的当前定位有关的视线跟踪数据515。根据所接收的视线跟踪数据515识别该场景的至少一个子区域。

[0148] 所接收的视线跟踪数据515可以涉及虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点。所识别的该场景的至少一个子区域可以涉及以虚拟现实用户设备的用户的至少一只眼睛的中央凹的当前注视点为中心的子区域。

[0149] 在进一步处理阶段510以第一分辨率处理该场景的数据。然而，在进一步处理阶段510以第二分辨率仅处理与所识别的该场景的至少一个子区域相对应的数据。

[0150] 在图形处理流水线500的每个阶段中处理场景的数据可以包括在图形处理流水线500的每个阶段中处理场景的数据帧。

[0151] 场景的数据可以包括与左眼视图相关联的数据和与右眼视图相关联的数据。图形处理流水线的每个阶段的数据处理可以包括处理与左眼视图相关联的数据和处理与右眼视图相关联的数据。所接收的视线跟踪数据515可以涉及虚拟现实用户设备的用户的左眼的中央凹的当前定位以及虚拟现实用户设备的用户的右眼的中央凹的当前定位。

[0152] 所识别的场景的至少一个子区域可以涉及以虚拟现实用户设备的用户的左眼的中央凹的当前定位为中心的左眼子区域和以虚拟现实用户设备的右眼的中央凹的当前定位为中心的右眼子区域。

[0153] 通过组合在进一步处理阶段510以第一分辨率处理的所述场景的数据和在进一步处理阶段510以第二分辨率处理的与所识别的所述场景的至少一个子区域相对应的数据来渲染所述场景。

[0154] 渲染场景可以包括首先将在进一步处理阶段510以第一分辨率处理的场景的左眼数据和在进一步处理阶段510以第二分辨率处理的与所识别的场景的左眼子区域相对应的数据组合在一起，其次，组合在进一步处理阶段510以第一分辨率处理的场景的右眼数据和在进一步处理阶段510以第二分辨率处理的与所识别的场景的右眼子区域相对应的数据。所识别的左眼子区域的至少一部分可以与所识别的右眼子区域的至少一部分不同。

[0155] 在一些示例中，在初始处理阶段505以第三分辨率处理场景的数据。在初始处理阶段505以第三分辨率处理完场景的数据之后，接收视线跟踪数据520，并且该视线跟踪数据520涉及在初始处理阶段505以第三分辨率处理完所述场景的数据之后虚拟现实用户设备的用户的至少一只眼睛的当前定位。在进一步处理阶段510以第三分辨率处理场景的数据。然而在进一步处理阶段510以第三分辨率仅处理与场景的另一子区域相对应的数据。渲染场景包括进一步组合在进一步处理阶段510以第三分辨率处理的场景的数据。根据所接收的视线跟踪数据515识别所述场景的另一子区域。场景的另一子区域可以基于所识别的场景的至少一个子区域来计算。

[0156] 参考图6，其示意性示出了图形处理流水线600的示例。图形处理流水线600被包括在图形处理系统中。图形处理流水线600包括初始处理阶段605和进一步处理阶段610。

[0157] 图形处理流水线600类似于上述图形处理流水线500，因为它包括初始处理阶段605和进一步处理阶段610。

[0158] 然而，在该示例中，图形处理流水线600还包括在初始处理阶段605和进一步处理阶段610之间的中间处理阶段620。

[0159] 进一步处理阶段610的处理可能具有比中间处理阶段620的处理更大的计算负担。例如，进一步处理阶段610可以包括一个或多个像素处理操作，例如光栅化和/或片元着色，并且中间处理阶段620可以包括一个或多个几何操作，例如顶点着色。

[0160] 在中间处理阶段620以第一分辨率处理场景的数据并且以第二分辨率处理场景的数据。在该示例中，在中间处理阶段620以第一分辨率和第二分辨率处理完场景的数据之后，接收视线跟踪数据615。在该示例中，视线跟踪数据615涉及在中间处理阶段620以第一分辨率和第二分辨率处理完场景的数据之后的虚拟现实用户设备的用户的至少一只眼睛的当前定位。视线跟踪数据615在图形处理流水线600中被相对较晚地接收，因此更可能准确地反映观看者的当前的一个或多个焦点。然而，因为它在图形处理流水线600中被相对较晚地接收，所以它对图形处理流水线600中进行的整体处理具有较小的影响。

[0161] 参考图7，其示意性示出了图形处理流水线700的示例。图形处理流水线700被包括在图形处理系统中。图形处理流水线700包括初始处理阶段705、中间处理阶段720和进一步处理阶段710。

[0162] 图形处理流水线700类似于上面参考图6描述的图形处理流水线600，因为其包括初始处理阶段705、中间处理阶段720和进一步处理阶段710。

[0163] 然而，在图形处理流水线600中，在中间处理阶段620以第一分辨率和第二分辨率处理完所述场景的数据之后，接收视线跟踪数据615；在图形处理流水线700中，在中间处理阶段720以第一分辨率和第二分辨率处理所述场景的数据之前，接收视线跟踪数据715。此外，在该示例中，视线跟踪数据715涉及在中间处理阶段720以第一分辨率和第二分辨率处理场景的数据之前的虚拟现实用户设备的用户的至少一只眼睛的当前定位。如此，在该示例中，在中间处理阶段720以第一分辨率处理场景的数据。然而，在中间处理阶段720以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据。此外，如上所述，在进一步处理阶段710以第一分辨率处理场景的数据，并且以第二分辨率仅处理与所识别的场景的至少一个子区域相对应的数据。视线跟踪数据715仍在图形处理流水线700刚开始之后被接收，而不是像上述的视线跟踪数据615那样在图形处理流水线700中的后期被接收。因此，与视线跟踪数据615相比，视线跟踪数据715比较不可能准确地反映观看者的当前的一个或多个焦点。然而，因为与视线跟踪数据615相比，在图形处理流水线700中更早地接收视线跟踪数据715，所以它能够对图形处理流水线700中进行的整体处理具有更大的影响。

[0164] 参考图8，其示意性示出了图形处理流水线800的示例。图形处理流水线800被包括在图形处理系统中。图形处理流水线800包括初始处理阶段805、中间处理阶段820和进一步处理阶段810。

[0165] 图形处理流水线800类似于上面参考图6和图7描述的图形处理流水线600、700，因为其包括初始处理阶段805、中间处理阶段820和进一步处理阶段810。

[0166] 在该示例中，在中间处理阶段820以第一分辨率和第二分辨率处理场景的数据之前，接收第一视线跟踪数据815。根据第一视线跟踪数据815识别场景的至少一个第一子区域。在该示例中，视线跟踪数据815涉及在中间处理阶段820以第一分辨率和第二分辨率处理场景的数据之前虚拟现实用户设备的用户的至少一只眼睛的当前定位。如此，在该示例中，在中间处理阶段820以第一分辨率处理场景的数据。然而，在中间处理阶段820以第二分辨率仅处理与所识别的至少一个第一子区域相对应的数据。

[0167] 在该示例中，在中间处理阶段820以第一分辨率和第二分辨率处理场景的数据之后，接收第二视线跟踪数据825。根据第二视线跟踪数据825识别场景的至少一个第二子区域。在该示例中，第二视线跟踪数据825涉及在中间处理阶段820以第一分辨率和第二分辨率处理场景的数据之后虚拟现实用户设备的用户的至少一只眼睛的当前定位。

[0168] 场景的至少一个第二子区域可以被包含在场景的至少一个第一子区域中。例如，第一视线跟踪数据815可以用于将处理限制到观看者可能正在观看的场景的一个或多个第一部分(基于第一视线跟踪数据815)。这节省了在中间处理阶段820中以第二分辨率处理所有场景。第二视线跟踪数据825可以用于将进一步处理限制到观看者可能正在观看的场景的一个或多个第二部分(基于第二视线跟踪数据825)，并且其中场景的一个或多个第二部分比场景的一个或多个第一部分小。第二视线跟踪数据825更有可能指示当场景被输出时

用户将正在观看的场景的部分。

[0169] 系统可以被配置为同时渲染两组或更多组视图。这可以有助于高效渲染混合分辨率，并允许后期锁定(late-latched)的眼部跟踪数据。后期锁定的眼部跟踪数据涉及在图形流水线的初始阶段之后使用的眼部跟踪数据。在本文描述的示例中，在图形流水线被提交之后，执行眼部跟踪数据的后期锁定。

[0170] 现在将描述首先在没有眼部跟踪情况下的渲染多个分辨率的示例技术。如上所述，多视图使得能够渲染相同分辨率的单个视图集合。这是将OpenGL®ES API描述为在一个或多个附接点(例如颜色，深度和模板)处绑定了从GL_TEXTURE_2D_ARRAY到GL_DRAW_FRAMEBUFFER目标的连续范围。GL_DRAW_FRAMEBUFFER目标在本文中也称为绘图目标。

[0171] 例如，使用深度测试渲染两个视图可以涉及一种应用程序：该应用程序创建宽x高x num_views的深度纹理GL_TEXTURE_2D_ARRAY、创建宽x高x num_views的颜色纹理GL_TEXTURE_2D_ARRAY、创建帧缓冲器对象、将深度范围绑定到帧缓冲器绘图目标的深度附件，将颜色范围绑定到帧缓冲器绘图目标的颜色附件，然后使用被配置为输出num_views的着色器绘制几何形状，在此示例中，num_views＝2。

[0172] 这可以通过引入标记为GL_DRAW_ALT_FRAMEBUFFER并且在下文中被称为draw_alt的新帧缓冲器目标来扩展。该新目标类型用于以不同但相关的分辨率复制被绘制到绘图目标的任何结果。draw_alt帧缓冲器实际上映射(mirror)绘图帧缓冲器。绑定过程可以与上述的相同，但为简单起见，draw_alt纹理可以与原始绘图配置相匹配。例如，如果具有分辨率为X的深度和颜色的两个视图被绑定到绘图目标，则将存在具有相关分辨率Y(X)的深度和颜色的两个视图被绑定到draw_alt。在一些示例中，使用不止一个“draw_alt”目标。在一些此种示例中，每个“draw_alt”目标都有不同的相关分辨率。在这种情况下，可以使用一组目标。

[0173] 每个绘图目标可以被允许具有完全不同的分辨率。可替代地，目标可以具有相关的分辨率，例如，以允许二次幂缩放的方式。这将允许通过顶点和平铺阶段处理的几何形状在两个集合之间被重用，如现在将要描述的。如此，上述第一分辨率和第二分辨率可以具有预定的关系，例如包括2的幂，例如1/4。然而，这两个集合可以仍是片元着色的。

[0174] 当执行注视点渲染时，可以控制GPU来渲染场景的同一视图的多个不同分辨率版本。例如，GPU可以渲染场景的同一视图的三个不同分辨率版本，即最高分辨率视图、中间分辨率视图和最低分辨率视图。然后，视图的不同分辨率图像被适当地组合(或“合成”)以提供例如用于显示的输出中央凹图像。每个不同分辨率的图像可以显示正在被渲染的整个场景的不同视野(或“部分”)。在一个示例中，与注视点相关联的中央凹图像的区域具有最高分辨率，并且远离注视点的图像的一个或多个区域以较低分辨率显示。

[0175] 请求渲染的应用程序可以向图形处理器的驱动程序指示场景的几何形状将被渲染到多个视图并且一起指示与中心凹视图相关联的比例因子。响应于来自应用程序的此类命令，驱动器随后配置合适的平铺和渲染任务，发送到GPU，以本文所述的方式执行注视点渲染。

[0176] 在一个示例中，场景的几何形状被处理并平铺，换句话说，被分类成仅被渲染一次的图像的相应渲染拼块的列表。这提供了单组拼块几何列表，然后在渲染每个相应的分辨率图像时，即当执行用于渲染每个相应分辨率图像的片元处理时，共同使用(或“共享”)所

述单组拼块几何列表。具体地，GPU可以被控制用于以输出图像所要求的最高分辨率、针对正在被渲染的场景仅执行一次顶点位置处理和几何列表生成过程。一旦已经以这种方式准备了几何列表，则可以在渲染显示部分或全部视图的每个相应的不同分辨率图像时，使用所述单组几何列表。

[0177] 对于最高分辨率视图，所准备的几何列表可以被使用，因为这些列表将标识在渲染最高分辨率图像时针对每个渲染拼块将要处理的几何形状。对于其它较低分辨率图像，几何列表和那些列表中的已经以最高分辨率准备的几何形状被用来通过以合适的缩放因子对全分辨率几何列表中的几何形状进行缩减，来识别和定义针对较低分辨率图像将要渲染的几何形状。缩放因子例如使用线性缩放将所讨论的较低分辨率图像的分辨率与已经为其准备好几何列表的最高分辨率相关联。该缩减可以作为顶点位置加载操作的一部分来完成，但是如果需要，其他布置将是可能的。

[0178] 在一些示例中，只有一些而不是全部的最终图像以不同的相应分辨率被渲染。具体地，在最终组合中将不被使用的图像的区域可以不被渲染。仅仅对每个不同分辨率图像的被选择部分进行渲染可以通过向输出图像的每个渲染拼块指示针对该分块位置应该产生具有什么分辨率的一个或多个图像来实现。该信息可以与渲染拼块的几何列表一起被提供。其他布置可以作为替代被使用，例如GPU的驱动器被配置为当向图形处理器提供渲染任务时提供要被渲染的拼块的合适列表，和/或指示渲染排除区域，该区域标识其中拼块不应该被渲染以用于图像的区域。

[0179] 一旦每个不同分辨率图像的适当部分已经通过GPU被渲染，则这些图像被存储在图形处理系统的存储器中的适当帧缓冲器中。然后，这些图像被适当地组合以提供将被显示的最终输出的中央凹视图。

[0180] 合并视图的不同分辨率图像以提供最终的中央凹输出试图可以由GPU来执行，该GPU将每个不同的分辨率视图视为图形纹理进行处理并且然后使用纹理映射操作(该纹理映射操作可以被实现为片元着色器的一部分)来适当地采样并组合要被显示的最终输出的中央凹视图上的不同分辨率图像。

[0181] 当执行此种注视点渲染时，最高分辨率图像不一定位于被显示的最终输出图像的中心，而是可以偏移到视图中的任何位置，这取决于一个或多个注视点的位置，该一个或多个注视点可以例如通过使用视线跟踪传感器识别观看者实际观看的地方来确定。

[0182] 渲染最高分辨率集合并且降采样以产生其他集合将不会产生相同的结果。

[0183] 多采样着色率可以在集合之间变化。在多视图中，就像分辨率应该优选地在所有被绑定到目标的纹理之间匹配，因此多采样计数应该优选地匹配。然而，分辨率、多采样计数、纹理格式和布局可以在不同的集合之间变化。描述可以被渲染的纹理的任何内容都可以变化。

[0184] 如上所述，本文描述的示例涉及使用后期锁定的眼部跟踪数据的注视点渲染。

[0185] 在一些示例中，硬件片元作业或工作项描述符包括2D轴对齐边界框，该2D轴对齐边界框将片元着色限制在仅由边界框界定的区域内。这是一种将全分辨率视图的着色限制到仅中心凹区域的有效机制，该中央凹区域与边界框相关联。边界框数据只在片元着色开始之前使用，因此可以在图形流水线的后期更改。这意味着在最差的情况下，相对于视线跟踪数据被使用的当前时刻，存在一帧延迟，即按照60Hz为16.7ms。这与完整图形流水线的

50ms形成对比。如果几何形状在单个帧内被提交、平铺和着色，则延迟可以被进一步减小。

[0186] 可能需要任何“后期”边界框数据被复制到之前提交的硬件片元作业描述符中，以便正确对其读取。给定新的数据，如果驱动器知道片元作业尚未开始，则这可以由驱动器完成，或由GPU计算着色器从共享缓冲器复制数据来完成。

[0187] 在流水线的后期提供数据在OpenGL®ES中并不常见，但手动同步访问数据是很好理解的。默认情况下，OpenGL®ES中的计算着色器是不同步的。用户使用同步图元来完成正确操作。这将提供足够的基础设施来及时更新片元描述符。一种实现方式可以利用原子和共享缓冲器或者通过自定义API入口点来实现等效功能。

[0188] Vulkan比OpenGL®ES更低级，除非明确要求，否则不会执行任何同步，因此与OpenGL®ES中的那些机制类似的机制在Vulkan中适用。Vulkan的一个特点是它的VkRenderPass对象已经具有定义2D轴对齐边界框的明确的“renderArea”属性，该2D轴对齐边界框将着色限制到指定的区域。开发Vulkan扩展可能比开发OpenGL®ES扩展更有效，因为Vulkan API中已经存在边界框概念。Vulkan可以引入vkRenderPass对象的扩展，其将允许用户定义“映射”其他内容的第二组目标。这可以允许等效于上述“GL_DRAW_ALT_FRAMEBUFFER”功能。

[0189] 在特定示例中，假设多视图具有两个视图，一种实现方式以完整分辨率为两个视图创建一组纹理数组(颜色、深度等)，并将这些数组绑定到帧缓冲区中的“draw(绘图)”目标。更新这些纹理数组的片元边界框以匹配最近报告的眼部跟踪数据。该系统以完整分辨率视图的分辨率的四分之一创建针对另外两个视图创建的匹配的一组纹理数组，并且将这些视图绑定到“draw_alt”目标。然后通过使用draw和draw_alt目标来绘制场景。如此，在该示例中，第一分辨率和第二分辨率之间的预定关系是1/4。

[0190] 然后在每个帧的开始，或以规则间隔，所有相关的已提交的和仍可修改的片元作业将被更新以使用最新的边界框。更新的理想时间是在接收到新的眼部跟踪数据之后，其可以与主渲染循环异步运行。

[0191] 该特定示例将产生针对所有立体图像的四分之一分辨率输出以及一对小但完整分辨率的区域以匹配眼睛位置。然后这些将被叠加以产生最终图像。该叠加过程将适于包括在随后的透镜校正或合成过程中，例如存在于低延迟VR系统中并且被称为“时间扭曲(timewarp)”、“异步时间扭曲”、“电子显示同步”等术语的那些过程。通过跟踪，它使用关于区域的位置和大小的已知信息。时间扭曲过程也可以在两个区域之间进行混合，以创建更柔和的转换，或选择将完整分辨率修剪为更小的替代形状，例如将框体修剪为球体。

[0192] 上述实施例应理解为说明性示例。可以设想其他实施例。应当理解，关于任何一个实施例描述的任何特征可以被单独使用或与所描述的其他特征组合使用，并且还可以与任何其他实施例或或任何其他实施例的任何组合的一个或多个特征组合使用。此外，在不脱离本公开的实施例的范围的情况下，也可以采用上面未描述的等同物和修改，其被限定在所附权利要求书中。

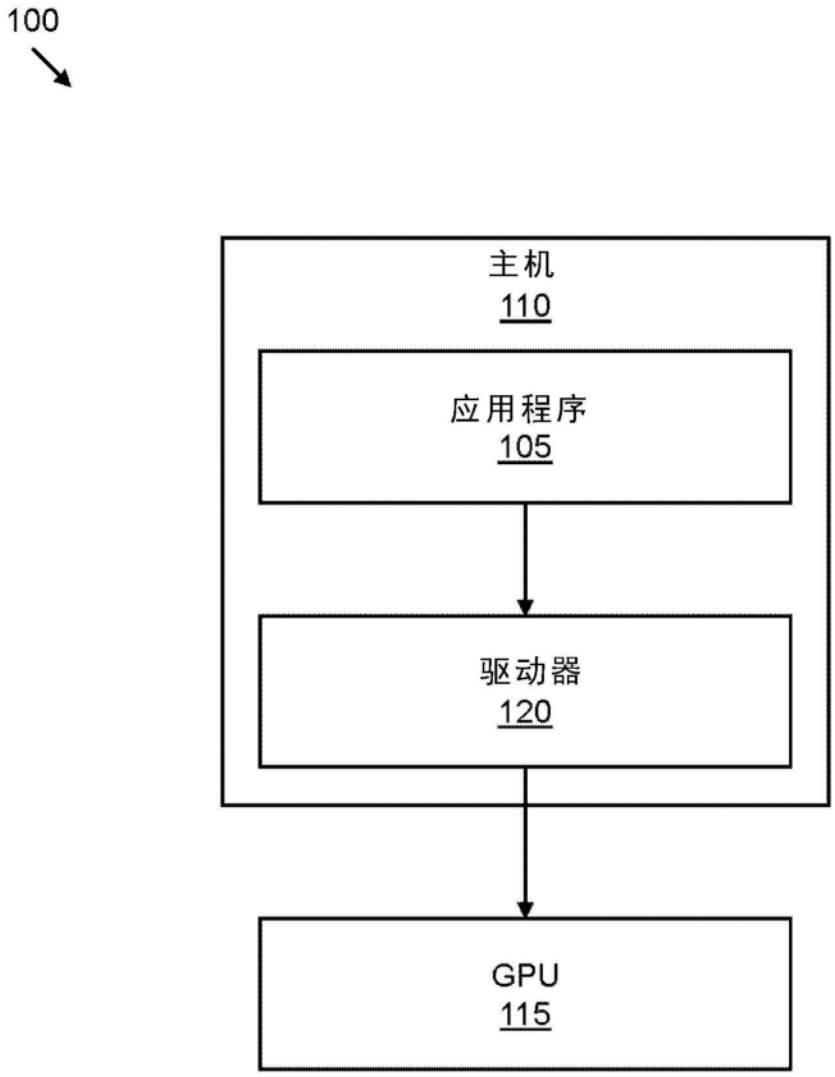


图1

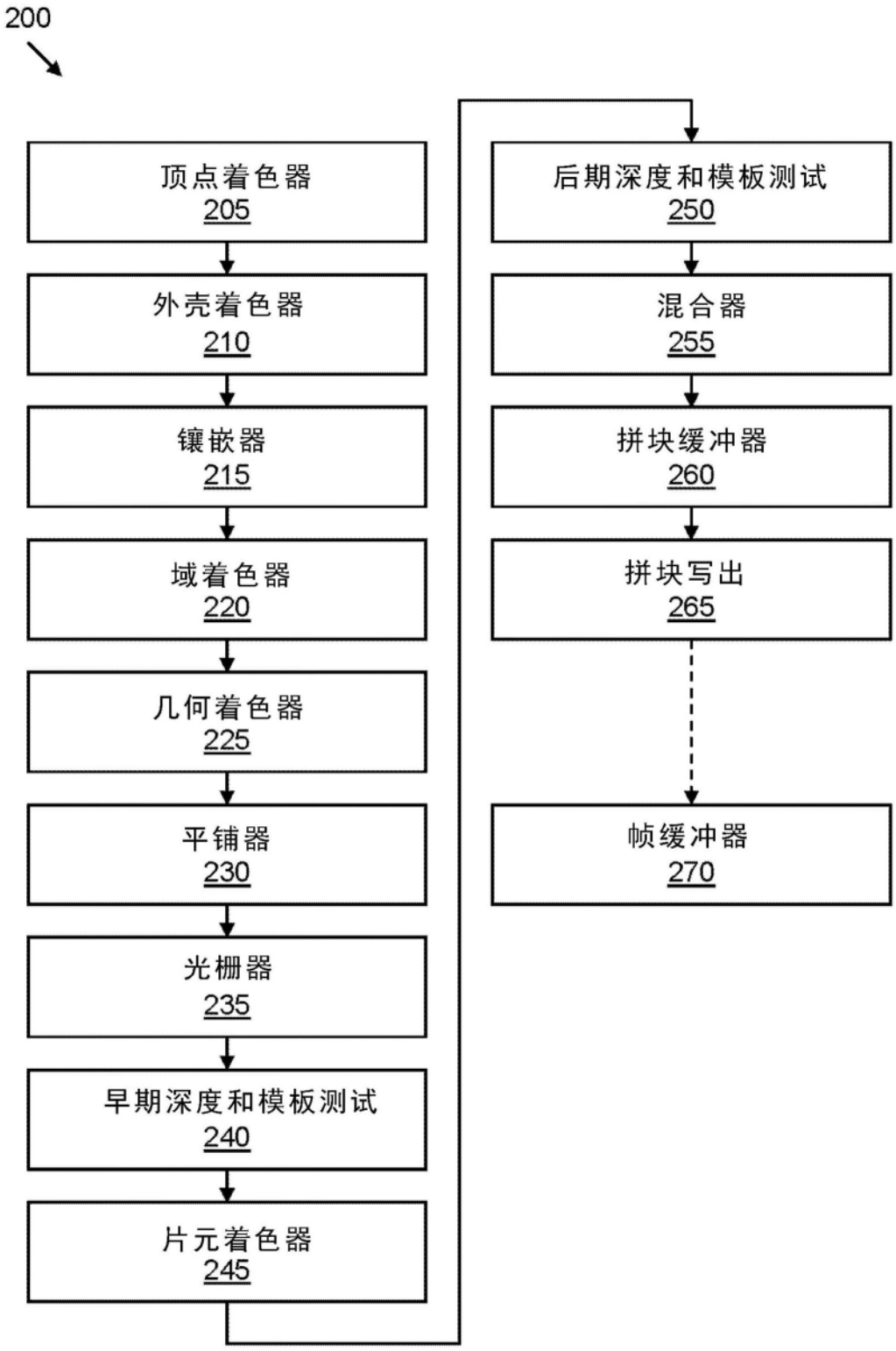

200
顶点着色器
205
外壳着色器
210
镶嵌器
215
域着色器
220
几何着色器
225
平铺器
230
光栅器
235
早期深度和模板测试
240
片元着色器
245
后期深度和模板测试
250
混合器
255
拼块缓冲器
260
拼块写出
265
帧缓冲器
270

图2

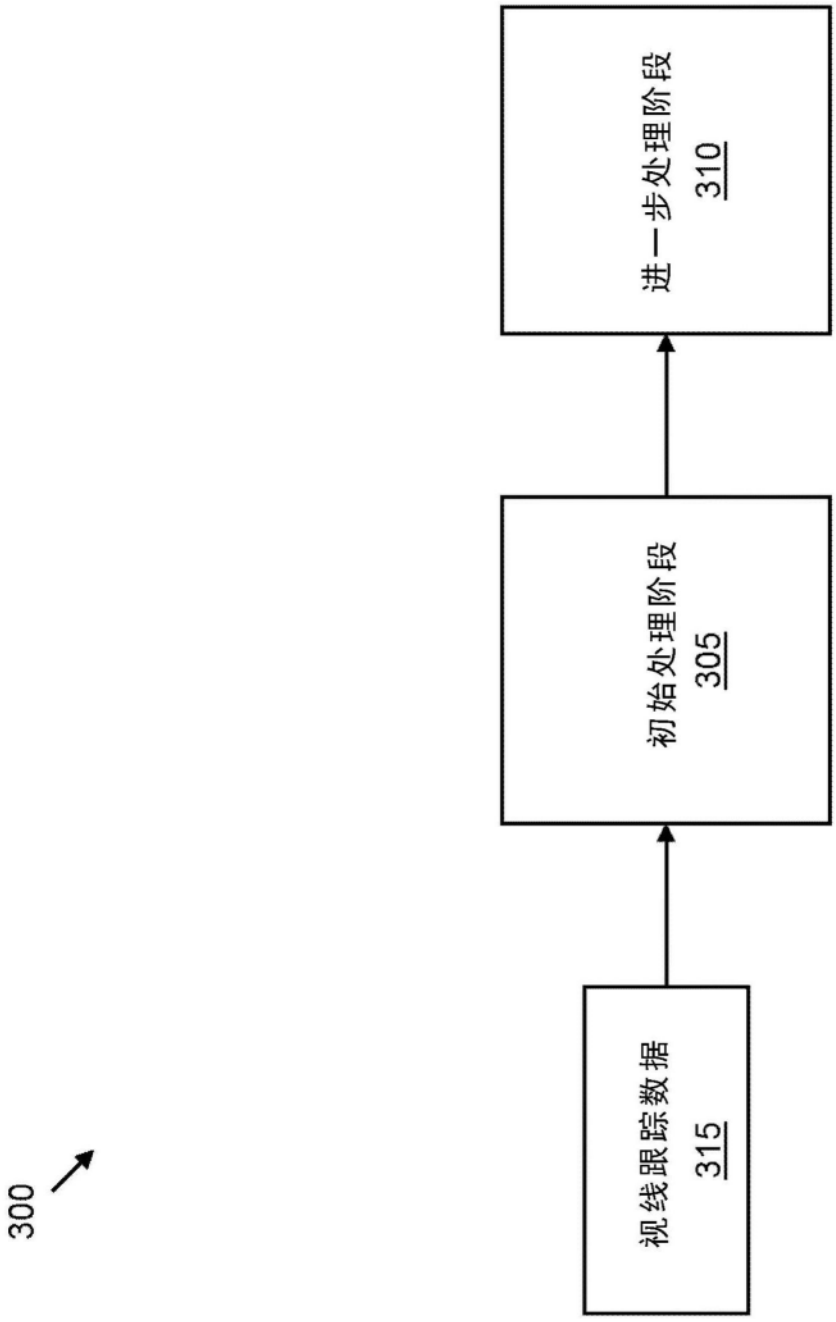


图3

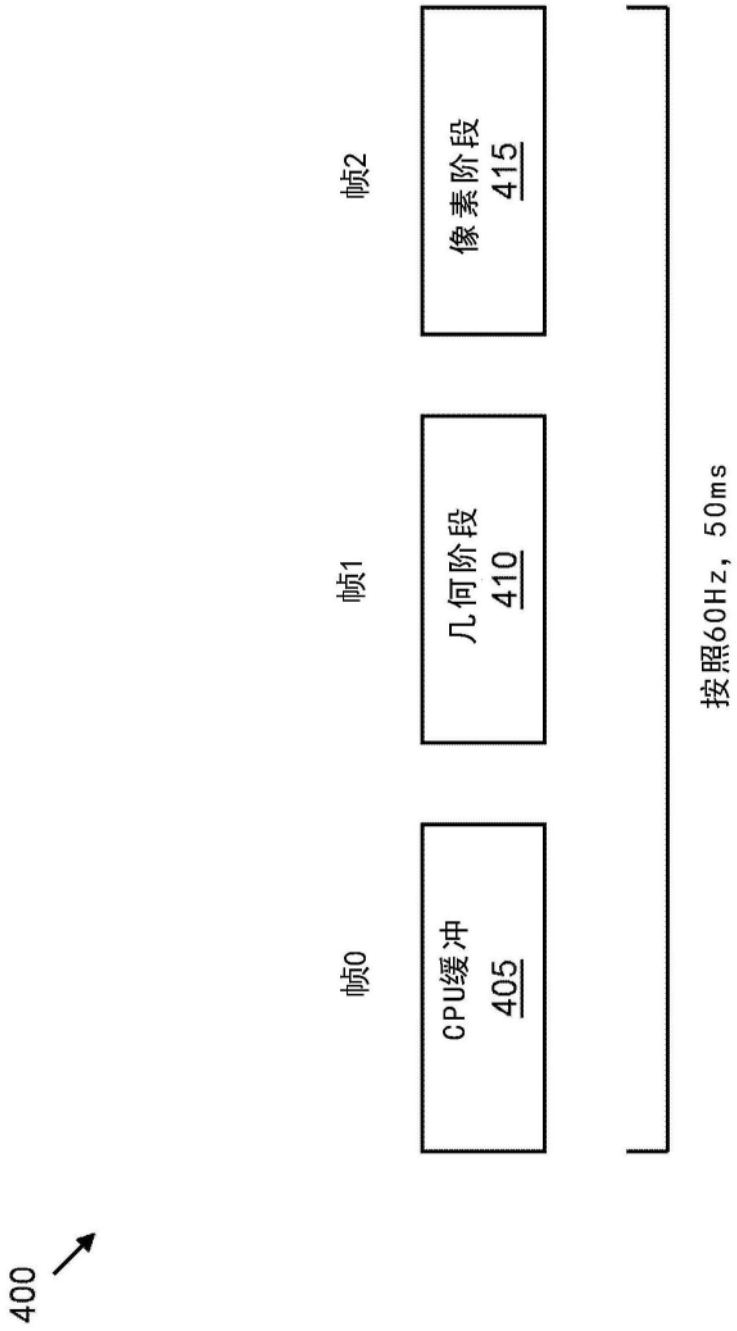


图4

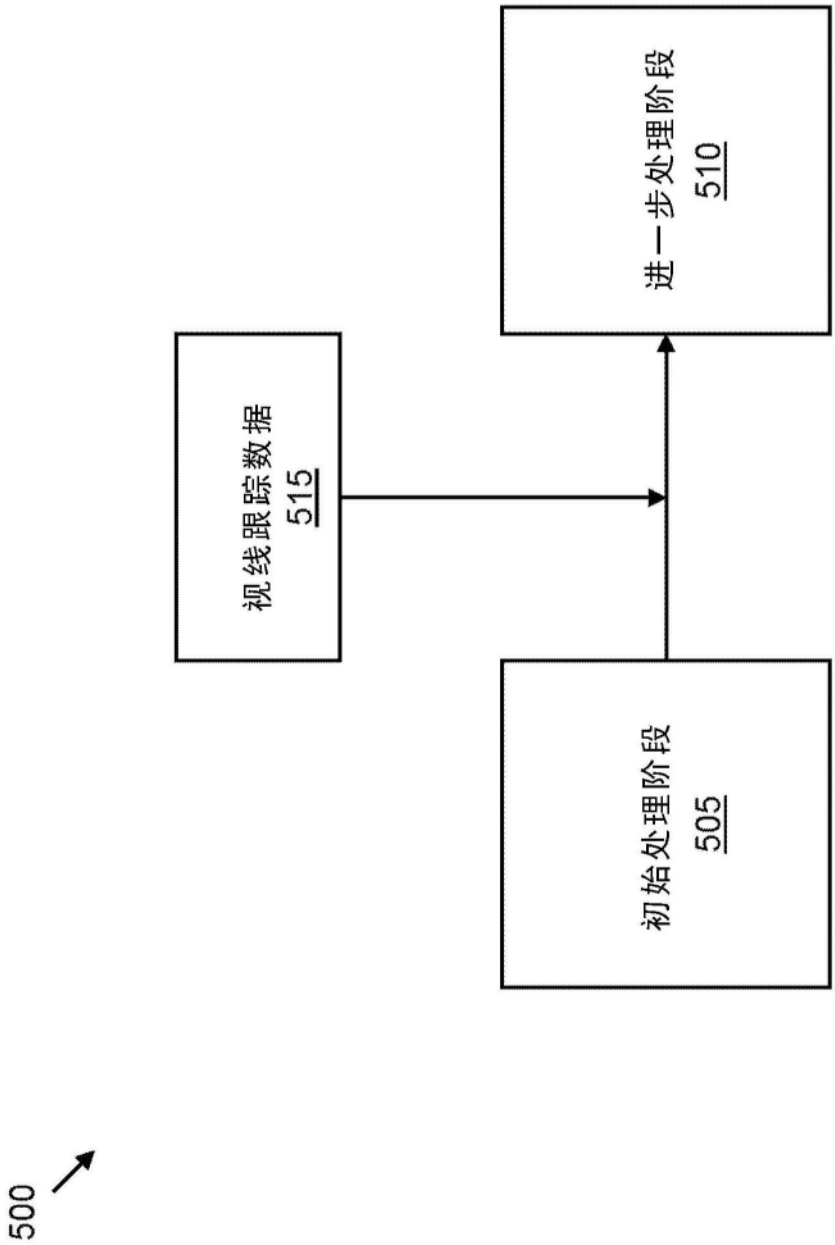


图5

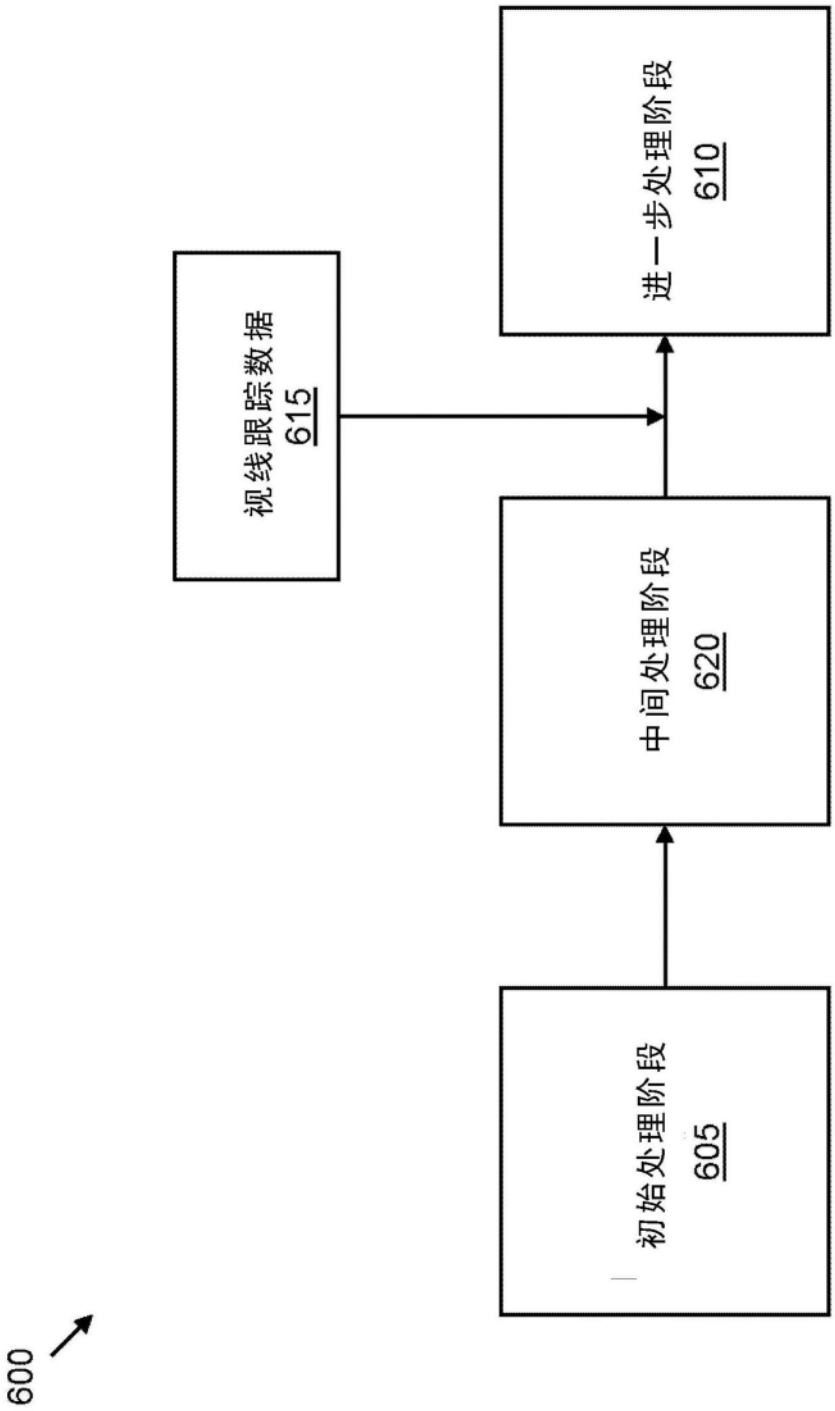


图6

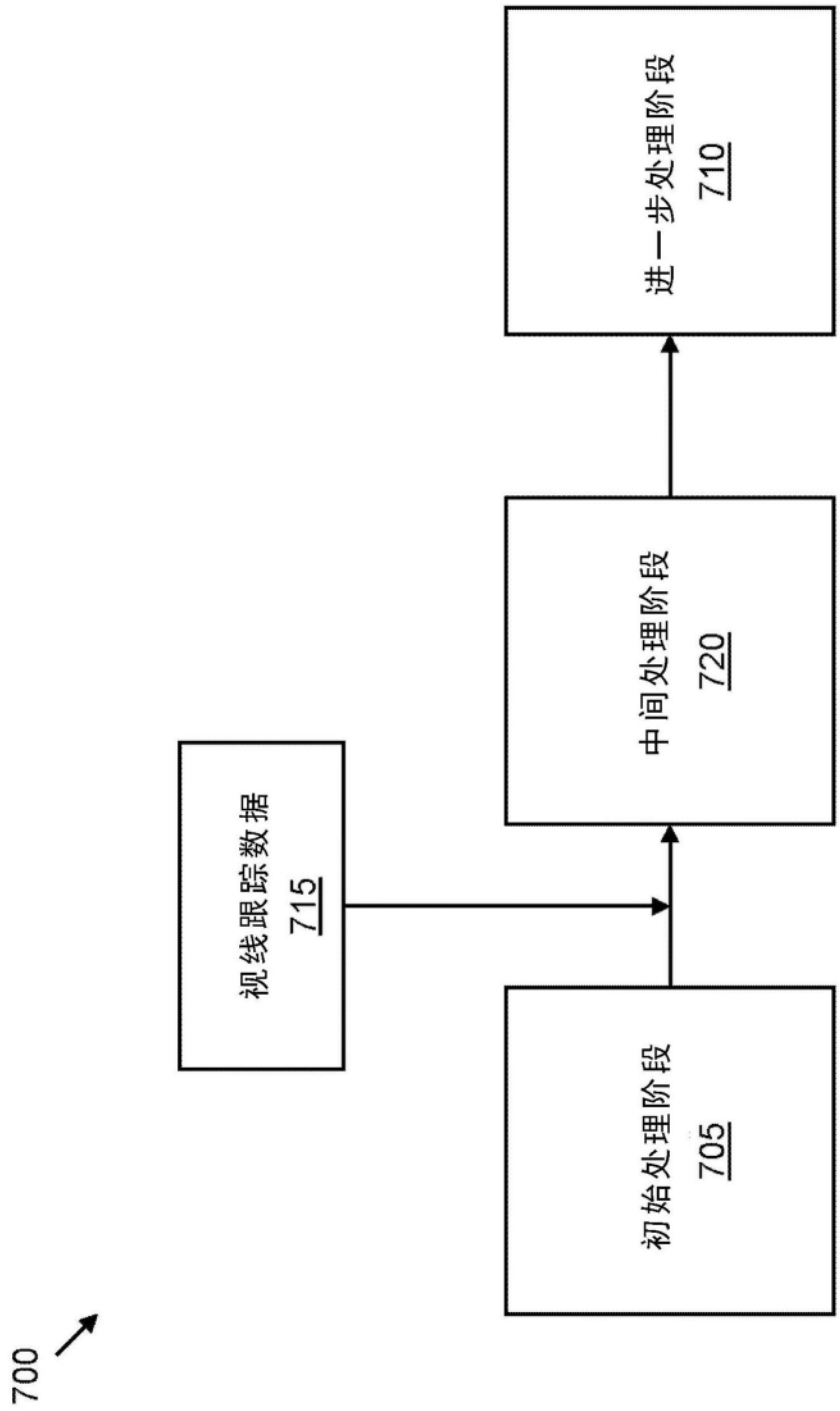


图7

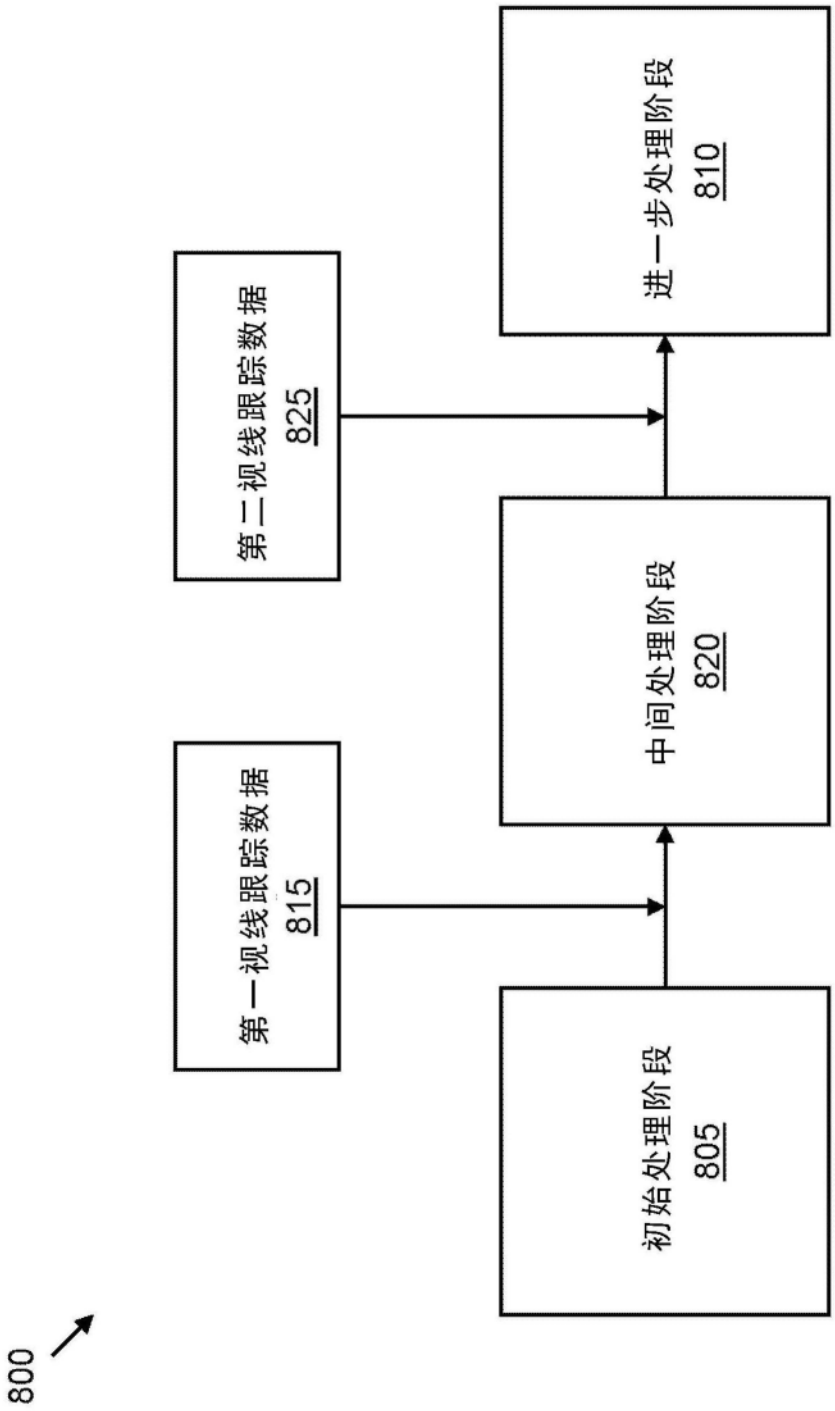

800
初始处理阶段
805
第一视线跟踪数据
815
中间处理阶段
820
第二视线跟踪数据
825
进一步处理阶段
810

图8