[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200680019715.5

[43] 公开日 2008 年 5 月 28 日

[11] 公开号 CN 101189601A

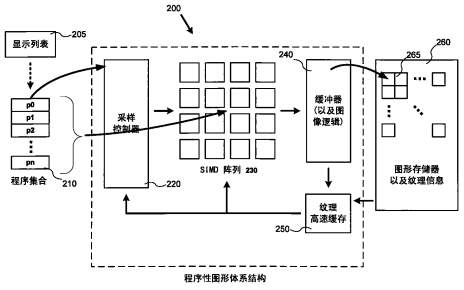
[22] 申请日 2006.6.28
[21] 申请号 200680019715.5
[30] 优先权
[32] 2005. 6.29 [33] US [31] 11/171,672
[86] 国际申请 PCT/US2006/025415 2006.6.28
[87] 国际公布 WO2007/002845 英 2007.1.4
[85] 进入国家阶段日期 2007.12.3
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 J·T·怀特德 J·T·卡吉亚
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 顾嘉运
权利要求书 3 页 说明书 21 页 附图 8 页

[54] 发明名称

程序性图形体系结构和技术

[57] 摘要

描述了呈现程序性图形的技术和工具。 例如，提供了允许为给定的程序参数值集合本地估计几何图形、变换、纹理和着色程序的体系结构。 该估计是在单指令多数据阵列上为不同参数值并行执行的，以允许并行处理程序集合。 在另一个例子中，描述了一个采样控制器，它基于标签映射、比率映射和参数映射中的信息为估计选择参数点集。



1．一种系统，包括：

被配置为本地估计程序性几何图形和至少一个其他类型的程序性图形描述的多个处理元件。

2．如权利要求 1 所述的系统，其中所述多个处理元件是单指令多数据算术逻辑部件阵列的一部分。

3．如权利要求 1 所述的系统，其中一个或多个图形程序描述所述程序性几何图形以及一表面的程序性变换、程序性着色和表面的程序性纹理。

4．如权利要求 1 所述的系统，其中所述多个处理元件还被配置为程序性地估计无源几何图形元素。

5．如权利要求 1 所述的系统，还包括被配置为控制所述程序性几何图形的自适应采样的采样控制器。

6．如权利要求 5 所述的系统，其中所述采样控制器是可完全编程的。

7．如权利要求 5 所述的系统，其中所述采样控制器被配置为在第一轮估计参数空间的至少一部分以便确定要在第二轮中估计的参数点集。

8．如权利要求 1 所述的系统，还包括：

贴片缓冲器，被配置为在图像样本被传送到图形存储器之前临时存储所述图像样本的一个或多个贴片。

9．如权利要求 8 所述的系统，其中通过分散方法将所述一个或多个贴片中一贴片的图像样本传送到所述贴片缓冲器。

10．如权利要求 8 所述的系统，其中：

所述多个处理元件被配置为每次估计一个贴片的图形程序；

所述多个处理元件被配置为估计所述多个处理元件将为其估计程序的贴片的所选程序子集；以及

为所述多个处理元件将为其估计程序的贴片选择参数样本。

11．如权利要求 1 所述的系统，其中所述多个处理元件还被配置成为估计所述程序性几何图形和参数空间中的多个参数点并行的至少一个其他类型的程序性图形描述。

12．一种呈现示出一个或多个表面的图像的方法，所述表面至少部分由图形估计例行程序阵列定义，所述方法包括：执行所述图形估计例行程序集合，在呈现期间进行估计时，这将产生所述图像的图像数据的点，其中，所述执行包括在呈现期间为所述图形估计例行程序集合中的每一个程序估计所选参数集中的每一个参数。

13．如权利要求 12 所述的方法，还包括：

审阅所述图形估计例行程序的阵列；

为所述图形估计例行程序中的每一个确定参数集，当估计时所述参数集将允许足够的呈现；以及

选择所确定的参数作为将要估计的参数的集合。

14．如权利要求 12 所述的方法，还包括：

将所述图像的空间划分为一个或多个贴片；

选择一贴片以用于呈现；

在呈现期间进行估计之前，

为估计选择在所选贴片中定义、且当被估计时产生所选贴片中的图像数据的点的一个或多个图形估计例行程序；以及

对为估计所选择的一个或多个估计例行程序中的每一个，选择映射到所述贴片中图像数据的点的一个或多个参数。

15．如权利要求 12 所述的方法，其中所述图形估计例行程序的集合为所述表面中的至少某些描述了表面形状和表面着色。

16．如权利要求 12 所述的方法，其中所述图形估计例行程序的集合为所述表面中的至少某些表面描述了非照片现实主义表面样式。

17．一种被配置为呈现由一系列图形程序定义的图像的图形处理单元，所述图形处理单元包括：

采样控制器，被配置为分析所述一系列图形程序并确定要估计的参数样本；

处理器阵列，被配置为根据所述一系列图形程序对由所述采样控制器确定的所述多个参数样本中的每一个并行地进行计算；以及

缓冲器，被配置为接收通过根据所述一系列图形程序的计算产生的像空间

数据。

18．如权利要求 17 所述的图形处理单元，其中所述采样控制器是可完全编程的。

19．如权利要求 18 所述的图形处理单元，其中所述采样控制器使用标签映射、比率映射和参数映射中的一个或多个来确定要估计的所述参数样本。

20．如权利要求 17 所述的图形处理单元，其中所述处理器阵列被配置为以随机存取的方式写入所述缓冲器。

程序性图形体系结构和技术

背景技术

图形呈现技术的发展导致了描述呈现过程中各步骤的程序性技术的发展。程序性几何图形用作根据紧凑描述产生任意复杂的几何图形的机制。举一个简单的例子，立方体可以无源（passively）地表示为包括 8 个顶点和 6 条边的列表的多边形表示。然而，可以开发更加紧凑的程序性呈现，其中所述立方体成为立方体生成程序的结果，这只需要输入位置坐标和大小。因此，几何程序往往提供有效的、紧凑的方式来表示形状，避免访问和传送多点数据。更复杂的程序，诸如旋转或样条（spline），对数据提供甚至更大的压缩。其它过程，诸如着色和纹理也利用程序性技术。实际上，可编程的程序性着色器被一些人看作是表示复杂材料特性的最有效的方法。

然而，常规的图形显示或图形处理器单元(“GPU”)体系结构借助于在固定、消极的多边形的图元上操作的处理链对程序性几何图形和程序性外观(诸如程序性着色器和纹理)进行了划分。常用方法是将程序性几何图形归类到预光栅化（prerasterization）极以将程序扩展到多边形中，并投入大量带宽将多边形馈送至图形处理器的变换和设置级。

图 1 中示出了这些传统技术的一个例子，其举例说明了传统的 GPU 体系结构的简化概观。显示列表 100 可以包含无源描述的几何图形和程序性几何图形的组合，它被输入顶点着色器（vertex shader）110 中，生成几何形状的多边形表示。然后这些几何形状被输入到光栅器中，后者对多边形进行内插并对其采样以导出像空间中的样本点集，然后可以对其着色和添加纹理。然后这些点被传递到一系列像素着色器 130，所述着色器是可以或不可以编程的，且利用并行计算技术来执行所述点的着色。另外，像素着色器将不时向给像点添加纹理，纹理被保存在图形存储器 150 中并且被高速缓存在纹理高速缓存 140 中。一旦这些过程完成，则在像点被传送到图形存储器 150 之前它们可被放置在帧缓冲器 160 中。

然而如上所述，因为几何程序被转换为多边形然后由光栅器转换为图像中的点的列表，所以紧凑的程序性图形信息往往在早期被暴露，而代替扩展的表面点集。当处理能力很低的时候，从资源和时间的角度来看，期望对于每个形状只计算一次几何程序，这样是有好处的。然而，当 GPU 处理器不断提高能力和效率时，这些处理问题会变得越来越小。

发明内容

本文描述了用于图形处理的多种技术和系统。当 GPU 的处理能力增加时，GPU 体系结构性能往往受到外部数据传送带宽对该体系结构的限制，而非受到处理能力的限制。首先不仅程序性图形表示是有益的，而且也期望将几何程序的扩展延续到后来的呈现过程中。当有更紧凑的程序性形式可以使用时，这可以帮助阻止体系结构中表面点的不必要的增长。本文描述的多种技术和系统在 GPU 体系结构内较长时间维护了图形物体的程序性表示。本文描述的多种技术和系统帮助减少对 GPU 的外来数据传送带宽。所述工具和技术包括，但是不局限于以下。

包括多个处理元件的一种工具，被配置为本地地估计程序性几何图形和至少一个其它类型的程序性图形描述。在一个例子中，一个或多个图形程序描述了程序性几何图形以及对表面的程序性变换、程序性着色和程序性纹理。在另一例子中，所述多个处理元件还被配置为对于参数空间中的多个参数点并行估计程序性几何图形和至少一个其它类型的程序性图形描述。

或者，一种工具呈现了示出一个或多个表面的图像，这些表面至少部分地由一组图形估计例行程序定义，如下所述。所述工具执行一组图形估计例行程序，当其在呈现期间估计时，产生该图像的图像数据点。当所述工具执行该图形估计例行程序集合时，它也是在呈现期间为该图形估计例行程序集合中的每一个估计所选参数集中的每个参数。

或者，图形处理单元被配置为呈现由一系列图形程序定义的图像。所述图形处理单元包括被配置为解析该一系列图形程序和确定要估计的参数样本的采样控制器。所述图形处理单元也包括被配置为根据一系列图形程序对由采样控制器确定的多个参数样本中的每一个并行进行计算的处理器阵列。所述图形

处理单元也包括被配置为接收由根据这一系列图形程序计算得到的像空间数据的缓冲器。

所述多种技术和系统可组合或者独立使用。

根据如下参照附图进行的实施例的详细描述，本发明的其他特征和优势将是显而易见的。

附图说明

图 1 是现有技术的图形处理单元体系结构的方块图。

图 2 是程序性图形体系结构的方块图。

图 3 是将参数从参数域映射到物体空间然后映射到像空间的表示。

图 4 是示出在图 2 的程序性图形体系结构中执行的用于呈现图像的过程的流程图。

图 5 是示出在图 2 的程序性图形体系结构中执行的用于估计图形程序的过程的流程图。

图 6 是图 2 的采样控制器创建和使用的、用于参数自适应采样的映射的表示。

图 7 是示出由图 2 的采样控制器执行的用以创建用于自适应采样参数的信息的过程的流程图。

图 8 是用于实现图 2 的系统的适当的计算环境的方块图。

具体实施方式

如下描述针对用于呈现图形的系统和技术。在一些系统和技术的体系结构中，表现在显示处理器链内较后阶段也保持程序性。例如，所述体系结构被配置为将紧凑的几何程序的数据放大延迟到它们可以与变换，纹理和着色计算合并的时刻。以此方式，几何，变换，纹理和着色组合且本地执行，利用了多数图形处理器的并行性。另外，通过保持计算的本地性并且延迟程序性几何形状的扩展，降低了外部数据带宽成本。在另一个例子中，程序性图形的采样是通

过可编程采样控制器执行的，所述控制器分析程序以用于产生表示关于所述程序的信息的映射，其又可以被用来产生在呈现其本身时使用的采样参数的集合。

本文所述的技术和系统总体上有时是参照硬件或者软件或者程序描述的，而其他时候是参照方法或功能描述的。这种用法是为了简化描述，而不应该被认为是隐含了对本文所述的技术和系统的任何特定的限制。将认识到，本文所述的技术和系统可以应用于各种级别的软件和硬件抽象，包括程序、方法、功能、例行程序、或者软件系统的其他部分、电路或者硬件系统的其他部分、或者组合硬件/软件系统的各部分。而且，本文所述的任何给定技术或者系统不必解决背景技术中提到的任何或者所有缺陷，也不必满足发明内容中提到的任何或者所有特征。

I．示例的程序性图形体系结构

图 2 示出了程序性图形体系结构 200 的方块图，根据本文所述的技术，在一些实施例中其与包含程序集合 210 的显示列表 205 和图形存储器 260 交互。正如所要详细说明的，GPU 体系结构 200 和传统体系结构之间的一个主要区别是单个指令多数据(“SIMD”)处理器阵列 220，它用于估计几何程序以及其它图形程序(例如，着色和纹理(和可能的变换)图形程序)作为一组合的过程，而不是在体系结构的两个或多个部分之间将工作分开。传统 GPU 体系结构和程序性图形体系结构 200 之间的另一个显著的差异在于体系结构 200 不包含光栅器。相反，常规光栅器的两个功能，内插和采样，是在所示体系结构 200 中分开。常规光栅器的采样部分实际上由可编程采样控制器 220 替代。以下参照图 6 和 7 描述了这样的一个过程示例，其中可编程采样控制器 220 选择了足够映射到像空间的采样位置。与此相对，常规光栅器的内插实际上是由所估计的每个采样点处的程序的估计(例如，利用如下所述的 eval()方法)而替代。

图 2 示出了显示列表 205，其包括多个表面的信息以便显示。所述显示列表 205 可以包含简单的无源形状，例如那些预定义为多边形的简单列表，乃至如需要使用纹理映射硬件转换为几何图形的那些编码在纹理内的复杂的无源几何表示。在图 2 中，显示列表 205 也包括程序集 210。例如，程序集 210 表

示为代码阵列的阵列，其中每个代码表示物体的一部分，并且每个代码阵列表示一个物体。或者，使用某种其它数据结构来表示程序集 210。在一个实施例中，所述代码被实现为如下所述的 eval()方法。或者，根据某种其它的组织或者形式来实现所述代码。所述 eval()方法可以用 C#编码，或者它们可以用某种其它解释或者编译的软件语言来编码。一般来说，程序 210 具有图形程序员所熟知的限制，例如不允许循环但允许条件执行，或者不允许处理器间通信的限制；或者不具有这种限制。

一旦有所述体系结构接收到，则遍历程序阵列并将其广播给 SIMD 阵列 230 和可编程采样控制器 220 两者。在一个实施例中，SIMD 阵列 230 包含很多算术逻辑部件(“ALU”)，类似于常规分段着色器。所述阵列 230 被描述为 SIMD 阵列，意味着在给定时间，SIMD 阵列中的每个 ALU 可用于对于其所分配的参数值执行与 SIMD 阵列的其它 ALU 相同的图形程序指令。下面将更为详细地描述参数值。这有效地允许所述体系结构对多个数据并行地执行相同的图形程序。在图 2 所示的体系结构中，SIMD 阵列 230 也能够从纹理高速缓存 250 中拉出被包含在图形存储器 260 中的纹理信息。替换实施例可以使用多指令多数据(“MIMD”)阵列。另外，SIMD 阵列的 ALU 可以被配置为自发地动作以减少数据传送和伴随的带宽问题。

当利用 SIMD 阵列估计 eval()方法时，程序集 210 的 eval ()方法在 SIMD 阵列中相继地执行。通常，这些方法将参数值作为自变量，不返回任何值，并且将它们的结果留在寄存器中以减少数据带宽使用。举例来说，完成的程序性物体可以像多轮着色器（multipass shader）一样执行，除了中间结果保存在寄存器中而不写入外存储器以外。在执行完成时，预定义的寄存器数值被传送到缓存器 240。

可编程采样控制器 220 也分析程序集 210 中的程序，然后自适应地为每个程序性表面单步调试参数域并将参数值馈送给 SIMD 阵列 230。要在可编程采样控制器中执行的代码可以被预先处理到从程序集 210 获得的 eval()方法的阵列中。在另一实施例中，采样控制器使用硬编码到其逻辑中的采样方法。采样控制器可以经由纹理高速缓存 250 使用来自图形存储器 260 的纹理信息，以产生样本集。当使用双轮采样时，在呈现轮采样控制器使用在早先的采样轮中产

生的采样速率信息。或者，采样控制器在呈现轮中使用其它采样信息。

在使用贴片（tile）的体系结构中，图像平面被划分为贴片，一般来说，这易于成块地向外部图形存储器 260 移动数据和从中移出。采样控制器 220 可以被配置为创建样本集并且另外根据图像贴片估计图形程序以使得一次可以呈现单个贴片，便于从缓存器 240 到图形存储器 260 的块传送。或者，采样可以被配置为对于整个图像处理图形程序(例如，采样、呈现)。

正如图 2 所示出的，在计算之后，输出图像在被放入图形存储器 260 之前，被本地地缓存在缓存器 240 中。在使用图像贴片的体系结构中，缓存器 240 是贴片缓存器，并且图形存储器 260 将图像组织为贴片 265，每个贴片表示完整图像的矩形子集。在一个实施例中，如贴片阵列中表示的每个贴片映射到最终图像的 64x64 像素部分，并且在每个维度中都按照 4:1 的系数进行统一的过采样，使得阵列是 256x256 的（不考虑边缘的填补的地址)。另外，缓存器可以包含填补物以适应缓存器 240 的图像逻辑中执行的抗混叠处理程序。在替换实施例中，也可以在缓冲器 240 中实现不同的贴片配置以及不同的图像逻辑。通常利用该结构时，因为采样点不必在像素中央生成并且因为它们是根据参数点生成的，所以为了如下所述的原因，它们不必以任何相干顺序写入图像存储器中。因此，在本地存储器中提供了用于将样本路由到目的地的某些手段。例如，SIMD 阵列中的每个处理器能够写入任意的像素(通常被称为分散操作），并且缓存器 240 包括用于写操作的路由网络以及用于传统图形呈现处理（诸如 z 缓冲、滤波和抽取、抗混叠处理以及纹理高速缓存）的其他硬连线像素逻辑。

诸如图像贴片大小以及 SIMD 阵列中 ALU 的数目之类的细节取决于实现方式。在一例中，图像贴片大小以及 SIMD 阵列中 ALU 数目是根据在 90nm 的 CMOS 工艺中什么可能侧立装配在 15mm 的管芯上这样的估算得出的，其中控制器 220、SIMD 阵列 230、缓存器 240 以及高速缓存 250 装配在一个单片上。或者，也可以使用其它限制或规定。

就缓存器大小而言，在一些实施例中缓存器包括足够用于两个贴片的存储。虽然 SIMD 阵列的 ALU 写入一个贴片的值，但是另一贴片可以被滤波并读取到外部存储器。或者，缓存器具有另一大小。

II．参数点的示例使用

图3是示出为什么程序性体系结构200被配置为写入贴片内的任意位置的一个例子的方块图。图3示出了参数域300，其示出了示例几何程序的一对参数(u，v)的参数空间的二维表示的例子，其中每个参数被标准化为0到1的范围。例如，如果程序是定义圆柱的各边，那么那些边上的任一点可以定义为一对数字，其中一个数字是圆柱该边的高度，而另一个数字是绕圆柱中心线的旋转量。（圆柱的其它细节，诸如总高和半径将在几何程序本身中设置。）虽然不同的几何程序可以具有少于或多于两个的参数，但一般来说，由程序描述的物体表面上的点可以被描述为2个或以上参数值的元组。因为这些数值在物理世界中可能具有或不具有确切的含义，并且因为参数空间中的移动在物体或像空间中可以是非线性的，所以使用术语“参数”而不是更几何学的术语，诸如“座标”。

如图3所示，参数域中示出的两个点映射到物体空间中物体表面上的3D表面点，物体空间的一部分被显示为物体空间310。(在全局坐标空间中，该物体空间将是这样的：其中圆柱位于由所呈现图像示出的场景中。）然后物体空间中的点被呈现为图像中的2D点，如所通过映射到像空间320中的贴片示出。所述两个点也示出了参数点和所映射的物体或像点之间的差别，因为两个参数点在参数域300中不是非常接近，但是映射到物体空间和像空间中相对接近的点。

当使用图像贴片时，整个像空间划分为多个贴片，如上所述，这又可以影响物体空间和参数域的哪些部分映射到特定贴片。因此，图3示出只有一部分物体空间落入像空间320中的贴片内，因此该部分被示出了。并且因为在呈现贴片时，只有部分物体空间有意义，所以追踪参数域的哪些部分映射到图像域中的该部分是十分有用的。因此，图3也示出了参数域305的着色部分，它表示映射到图3中所示的那部分物体空间310中的点的参数域中的点。这由所示处于着色部分305之外的参数点进一步表明。如图3所示，该参数点未映射到物体空间中处于被映射到贴片320内的部分的点。下面将参照采样程序更详细地描述参数域中已知映射到贴片的点和参数域中已知未映射到贴片的点之间的差别。

III. 示例呈现过程

图4是示出用于利用程序性图形体系结构200来呈现图形的过程的一个例子的流程图。在各个实施例中，图4中示出的块可被组合、分为子块、或被省略的，并且块的顺序可以被修改。图2的程序性图形体系结构200执行所述过程。或者，另一工具执行所述过程。

所述过程从块410开始，其中由所述体系结构接收到图形程序列表。在一个实施例中，如上所述，该列表被接收为图形程序阵列的阵列。另外，也在块410接收无源图形学表示。

然后在块420，像空间被分成多个贴片。例如，这是根据缓存器240的硬件能力来完成的；或者，可以在呈现的时候确定贴片的大小，其中贴片缓存器足够大以能够处理最有可能的贴片大小。在一些场景中，参考像空间的大小来选择贴片，以确保像空间被正确地分割。

然后在块430，分析程序列表中的程序以确定将用于呈现的参数点集。下面将参照图6和7更详细描述这种程序的一个例子。一般来说，选择所述参数点集，使得产生的几何图形点足够用于所期望的细节水平，但不是太密集的(这将意味着对于该细节水平的计算的浪费)。因此，可以选择许多靠近的参数点以在参数空间的一个区域中呈现，选择更加宽间距的参数点以在参数空间的另一区域中呈现。

然后，在块440选择贴片来呈现。取决于所使用的方法，可以顺序地、根据分层结构、或者根据可利用的处理器资源来选择贴片。在可替换实现方式中，贴片是在块430的过程之前选择的，在这种情况下为每个贴片重复块430的采样过程。

所述过程在块450继续，其中为特定贴片估计图形程序。下面将参照图5描述该估计的一个例子。

然后在块460，所述过程将贴片写入图形存储器。例如，一旦估计了所述贴片的程序之后，则将得到的图像样本信息写入贴片缓存器中的图像位置。

然后所述过程在判定块470确定是否有其他的贴片要呈现。倘若有，则所述过程返回到块440并重复。如果没有，则所述过程结束。

虽然图 4 示出了在逐个贴片基础上进行的程序性估计和呈现，或者，GPU 体系结构为整个非贴片化图像估计程序。

图 5 是示出利用程序性图形体系结构 200 来估计图形程序的过程的一个例子的流程图。因而图 5 的过程是用于实现图 4 的块 450 的过程的一个例子。在各个实施例中，图 5 中示出的块可被组合、分为子块、或被省略，并且块的顺序可以被修改。图 2 的 SIMD 阵列 230 被用来执行所述过程。或者，另一工具执行所述过程。在图 4 的上下文中，为在图 4 的过程期间所选定的每个贴片执行图 5 的过程。

在根据图形程序估计的过程中，程序的指令被传送到 SIMD 阵列。当使用贴片时，这些程序可以被这样选择，使得只将与贴片中所显示的形状相关联的那些程序的指令发送给 SIMD 阵列，以阻止无关的执行。

在块 520，由采样控制器 220 将所采样的参数点发送给 SIMD 阵列。例如，这些参数点是鉴于采样轮中预先计算的比率映射和参数映射来选择的。或者，参数点是按其它机制选择的。

然后，块 530-570 示出了由 SIMD 阵列连续执行的图形程序。如上所述，程序性图形体系结构允许对所采样的每个参数点连续地估计图形程序的完整集合(例如，用于几何图形以及变换，和着色)，并且这些估计是在 SIMD 阵列上为不同参数点并行执行的。这允许大量（brunt）计算工作被并行执行而无需执行所述估计的 ALU 之间的通信，移除了传统的 GPU 体系结构从几何图形扩展级到纹理映射级再到着色级等等所需要的大部分数据带宽需求。

在本章节的示例程序中，用于几何图形、纹理、着色和变换的程序是级联（concatenated）的代码片段。分开的代码片段可以简化编程并便于重新使用。或者，各个代码片段被组合成单个(例如，经编译和优化的)程序。在任何情况下，级联程序可以通过依靠在代码片段结束执行时以无源的中间表示保存在处理器寄存器中的简单的数据结构将由下一个片段使用，而无需存储器读写来进一步改善。参数 u 和 v 以及表面上的点的一个这种结构的示例是：

```
struct point {
  float u, v;
  float x, y, z, w;
  float nx, ny, nz;
  float r, g, b, a;
}
```

在上述例子中，采样参数被保存在 **u** 和 **v** 寄存器中， **x**、**y**、**z** 和 **w** 表示 4-D 空间中的坐标(如变换程序中通常用的)，**nx**、**ny** 和 **nz** 表示物体空间中的点，**r**、**g** 和 **b** 表示 RGB 色彩空间中的颜色输出，**a** 表示不透明度。因此在以上块 520 的过程中，在每个 ALU 上，以参数值填充 **u** 和 **v** 寄存器。在其他实现方式中，如果程序需要，可以为参数值和/或得到的采样值提供更多或者更少的寄存器。

该简单结构提供了图形程序之间进行交换的一般但可能冗长的媒介。实际上，采用中间结果的无源表示不必然限制程序的灵活性。例如，有时将取景变换（viewing transform）移动到执行链结束处是方便的。由于几何图形、着色和变换全部都集中在本地处理器，这种移动变成了代码细微的重新排序。然而更一般地，使用中间表示的固定无源的结构是与级联代码片段的使用相关的实现细节，并且它不必是图 2 中示出的 GPU 体系结构的特征。

参照(简化的)示例执行块 530-570 的过程。所述示例程序描述了通过沿 v 的第二 2D 函数的路径扫描 u 的 2D 三次函数而形成的双变量表面。对于不同表面来说(例如回转面、任意扫描面和 3D 矩形多边形)，图形程序依此改变。

对于 SIMD 阵列的每个 ALU，u 和 v 值逐一地被传送。然后示例程序的指令被传送到 ALU。尽管传统的片段着色渐变器使用向量操作，但是为简单起见以下显示的操作是标量的，而它们相反也可以是向量操作。而且，在示例程序中，在赋值语句左边的任何变量被本地存储在 ALU 寄存器中，而如上所述 point struct 的元素被表示为 p.var(例如，p.x)。在赋值语句右边的变量或者是本地寄存器值或者被嵌入程序本身中。

起始于块 530，执行几何程序以确定参数映射到物体空间中的哪里。这种程序的一个例子是：

```
// 初始部分：双三次曲面
// 计算 u 曲线和法线
// 80 次乘法，44 次加法
ucmp = 1.0 - u;
var0 = u*u*u;
var1 = 3.0*u*u*ucmp;
var2 = 3.0*u*ucmp*ucmp;
var3 = ucmp*ucmp*ucmp;
xu = var0*x0 + var1*x1 + var2*x2 + var3*x3;
yu = var0*y0 + var1*y1 + var2*y2 + var3*y3;
nxu = 3*(y0 - 3*y1 + 3*y2 - y3)*u*u + 6*(y0 - 2*y1 + x2)*u + 3*(y1 - y0)
nyu = 3*( - x0 + 3*x1 - 3*x2 + x3)*u*u +6*(x0 - 2*x1 + x2)*u +3*(x1 - x0);
scln = recipsqrt(nxu*nxu + nyu*nyu);
nxu = nxu*scln;
nyu = nyu*scln;
// 计算 v 曲线和法线
... 与 u 曲线相同 ...
// 计算扫描面上的点
p.x = xu + nxu*xv;
p.y = yv;
p.z = yu + nyu*xv;
```

在下一个块 540，可以包括建模（modeling）变换以用于在全局坐标中定位该物体。在很多情况下该变换也许已应用于嵌入程序的系数。

在下一个块 550 增加纹理。这种程序的一个例子是：

```
// 第二部分：合成纹理
nse = nseTab2[ p.x , p.z ];
tb += abs(nse*scale);
scale /= 2.0;
// 为还有两个八元组（octave）重复
. . .
idx = thrs sin(px/period + const);
if (idx)
  p.r = red0;
  p.g = green0;
  p.b = blue0;
else
  p.r = red1;
  p.g = green1;
  p.b = blue1;
```

在块 560，执行着色程序。这种着色程序的一个例子是常规的 Phong 着色器。在此例子中，所述 Phong 着色器不同于典型的实现方式，因为它在全局坐标中操作，视点在程序内指定。

接着在块 570，在先前部分计算出的样本的位置被变换为图像坐标。这种程序的一个例子是：

```
// 最后程序：取景
// 17 次乘法，1 次除法，12 次加法
tx = m00*p.x + m01*p.y + m02*p.z + m03*p.w;
ty = m10*p.x + m11*p.y + m12*p.z + m13*p.w;
tz = m20*p.x + m21*p.y + m22*p.z + m23*p.w;
tw = m30*p.x + m31*p.y + m32*p.z + m33*p.w;
winv = 1.0/tw;
p.x = tx*winv;
p.y = ty*winv;
p.z = tz*winv;
```

此时每个 ALU 的本地寄存器的最终图像样本内容准备好在块 580 被传送到贴片缓存器中。数据传送代码的一个例子是:

```
img[p.x,p.y].rgbaz = p.rgbaz;
```

其中 p.x 和 p.y 是像空间中样本点的坐标并用于在贴片缓存器中寻址，其中 rgbaz 表示样本的 RGB 值以及不透明度 a 和 z 顺序（z-order）。

图 5 的处理只计算单个图像样本。然而，块 530-570 的估计是对块 520 向 SIMD 阵列给出的参数值样本并行执行的。

下面是关于示例性程序的若干观察结果。

第一，如上所述的示例程序缺少裁剪功能。如果编译该程序性描述的预处理器确定所产生形状没有任何部分比近裁剪面更近，那么不必执行裁剪代码。

第二，图形状态的各个方面被嵌入程序本身之内。这些包括取景参数和建模变换参数。GPU 体系结构可以包括对所估计的程序性描述执行运行时处理的预处理器，以便产生嵌入到程序内的隐含图形状态。

第三，尽管以上未示出，但还可以包括无源的表示，例如经编码的多边形网格，并将其发送到 GPU。

第四，不需要多边形网格的无源表示的显式转换。例如无源表示可以被转换为几何图像，其作为纹理映射被馈送到 SIMD 阵列。该例子的几何程序是简单的双线性内插，这是纹理映射硬件的内建函数，并且可以使用形状的平凡（trivial）

规定：

```
// 几何图形图像示例
// 取回位置和法线
p.x = txtrX[u,v] + translateX;
p.y = txtrY[u,v] + translateY;
p.z = txtrZ[u,v] + translateZ;
p.w = 1.0;
p.nx = txtrNX[u,v];
p.ny = txtrNY[u,v];
p.nz = txtrNZ[u,v];
```

第五，当使用建立轮并且预先确定执行顺序时，可以预取诸如例如纹理映射或者几何图像的列表数据（tabulated data），且能够可靠地预测片外存储器延迟的补偿。

第六，尽管示例程序是以可预测的方式使用中间无源记录的寄存器的级联代码，但是由编译器生成的程序可以以任意方式重新分配寄存器。每一 ALU 的寄存器数目仍然是对这种重新分配的物理限制。或者，所述编译器可以产生其中寄存器被分页到外部存储器的代码。

第七，尽管一些程序性体系结构反映了像素中心的方法，其中程序性几何图形产生像素对齐的内插值，但上述的示例程序是参数中心的。或者，本文所述的 GPU 体系结构中的采样和内插可以同时适于像素中心的和参数中心的算法。

IV．示例采样程序

虽然采样控制器 220 可以采用朴素方法并将来自整个参数空间的均匀间隔的参数元组发送到 SIMD 阵列 230，但是这种方法通常很浪费，因为它导致对最终图像中未示出的参数样本进行处理。它还可以导致图像域中一些区域的采样不足。因此在一些实施例中，采样控制器利用自适应采样。与几何样本的任何正向映射一样，自适应采样的任务是选择参数域中将足够，但不非常密集地，映射到像空间的采样位置。虽然存在不同的自适应采样程序，但是本文所述的是这种程序的一个新颖的例子。

对于自适应采样来说，从程序估计部分进行反馈的某些手段很有用。本文所述的过程使用双轮方法。在第一轮中，以通过每个物体的参数域的稀疏循环遍历该情形。下面示出了这种稀疏轮的一个(简化的)例子：

```
foreach(object)
{
  foreach(sparse s,t)
  {
    if ( P(s,t) in tile_{i,j} )
    {
      tag[obj,tile_{i,j}] = true
      prmMap[obj,tile_{i,j},s,t] = true
    }
    rateMap[s,t] = local sampling rate
  }
}
```

其中 P(s,t)给出了为输入参数值 s 和 t 计算的图像样本，并且 $tile_{i,j}$ 是贴片集合中坐标 i,j 处的贴片。如果参数值(s,t)的估计得出 $tile_{i,j}$ 中的图像样本，则“prmMap”映射中的该参数的相应条目被置为真。一旦任何图像样本被映射到贴片中，则“tag”映射中的 $tile_{i,j}$ 条目被置为真。例如如下所述，也计算出参数值(s, t)的 rateMap 条目。

利用示出的“tag（标签）”和“prmMap”结构，采样控制器执行类似于传统流水线中裁剪的功能。图 6 示出了贴片集合 600 在稀疏扫描中生成的标签（tag）映射 610 和参数映射 620 的集合。标签映射 610 指示特定形状投影到哪些图像贴片中。当呈现贴片时，这允许控制器跳过其像空间投影完全落到贴片之外的物体。(注意，图 6 中示出的标签映射在未被所示物体覆盖的贴片(右侧中间的贴片)中具有“1”。这是因为所述映射表示过度保守的覆盖估算。)虽然标签映射 610 仅仅示出了取决于形状是否处于相关联的贴片中的 1 或者 0，或者，标签映射包含更多信息，诸如哪些形状位于哪些贴片中，以便更高效地驱

动呈现。

参数映射 620 示出了贴片在参数空间中的裁剪映射。它指示对投影到给定贴片的给定物体，参数域中在该贴片内产生投影的区域。因此，参数映射 620 可用于避开参数域中将在贴片外产生像空间投影的区域。跳过参数映射 620 中的参数域的空区域帮助使采样器保持远离当前贴片之外的不必要的处理。或者，可为每个贴片保存简单的 minmax（最小最大）参数值(表明参数值的简单范围)，这将更为紧凑。然而在测试时，参数映射 620 证实，在产生给定贴片之外的像空间投影的裁剪参数值处的效率高约 2.5 倍。在一些实现方式中，填补参数空间和屏幕空间中的边界以确保稀疏采样不会错过应该被覆盖的贴片。在某种意义上，参数映射类似于 Rhoads 等人在 *Computer Graphics（计算机图形）*(1992 *Symposium on Interactive 3D Graphics（关于交互式 3D 图形的专题）*)(1992 年 3 月)中的“Real-time Procedural Textures（实时程序性纹理）”中描述的“region-hit flags（区域命中标志）”，以及 1998 年 Chapel Hill 的北卡罗来纳州大学计算机科学系 Olano 哲学博士的论文“A Programmable Pipeline for Graphics Hardware（图形硬件的可编程流水线）”中提到的“暗示（hints）”。比率映射在图 6 中未示出，即以上列出的示例代码的“rateMap”结构。所述 rateMap 结构类似于 Rhoads 中描述的纹理比例因子，除了它反映从参数空间到像空间的缩放比例而不是相反的之外。参照图 7 描述用于计算参数空间和贴片的比率映射(以及计算贴片和参数映射)的示例程序。

图 7 是示出用于生成信息供程序性图形体系结构 200 中的自适应采样中使用的一个例子的流程图。因而，图 7 的过程是用于实现图 4 的块 430 的过程的一个例子。在各个实施例中，图 7 中示出的块可以被组合、分为子块或者被省略，且块的顺序可以被修改。图 2 的可编程采样控制器 220 执行所述过程。或者，另一个工具执行所述过程。如上所述，图 7 的过程可以在逐个贴片的基础上执行，或者一次对所有贴片执行。

处理从块 710 开始，其中选择了用于采样的稀疏参数比率。稀疏采样的程序性图形体系结构中的开销在某种程度上取决于对通过参数值的稀疏循环的分辨率。精细分辨率将产生高开销，虽然过于粗糙的循环会将事实上被像空间投影部分覆盖的贴片误标记为未命中（missed）的。可以填补参数空间和像空

间两者中的边界以避免它(或者至少减轻其概率)，但保守估算也引入开销。例如在一个场景中，用实验方法推导出稀疏采样率，所述稀疏采样轮生成最终样本数的约 25%，而稀疏采样轮的开销是总体呈现开销的大致 14%。在其他场景中可以观察到不同的开销结果。

如图 7 所示，可以顺序地估计将要在稀疏采样循环中估计的参数元组。或者，可以在 SIMD 阵列中并行估计将要在稀疏采样循环中估计的参数元组(使用比率的单独计算，如下所述)。

接着，在块 720，参数元组(在上述例子中，所述元组是 2 元组，或者一对)被选来采样测试。在块 725，计算所述参数元组所估计的像点。例如，可应用的几何图形程序是以参数元组作为输入而进行估计的，并且所得到的 3D 表面点被变换到全局坐标然后变换到像空间。

然后在块 730，采样控制器确定所述像点(在块 725 计算的)落入哪个贴片。当在块 740 使用标签映射时，对所述标签映射中与该贴片相对应的条目置位。当使用参数映射时，该过程接着继续到块 750，其中在与块 730 确定的贴片相关联的参数映射中对参数元组的条目置位。

接着，通过在块 760 估计相邻参数值处的几何图形，然后在块 770 计算像空间中所得到的本地邻居的半径来计算比率映射。所述半径信息被存储在比率映射中。例如，每个物体的二元组的两个参数中的每一个的半径值被存储在分辨率与稀疏参数循环的分辨率相同的小纹理映射中。换言之，所估计的参数元组在比率映射中有相关联的条目。用于采样比率确定的逻辑例如是在贴片缓存器中实现的。

在一些实现方式中，SIMD 阵列不允许处理器间通信，并且特定元组的邻值是在单个 ALU 中顺序计算的，结果在图像贴片逻辑中累加。或者，如果允许在 SIMD 阵列的 ALU 之间进行处理器间通信，那么可以从并行估计其他元组的 ALU 共享邻值。

在判定块 780，采样控制器确定是否存要检查的更多参数元组，并且如果有，则该过程在块 720 重复。

如果没有，那么则该过程然后在块 790 基于比率、标签和参数映射选择参数值集合供呈现期间的参数空间的自适应采样使用，然后结束。例如，在随后

的呈现轮期间，为每个物体读取比率映射(在图 2 的体系结构中，经由纹理高速缓存 250)，并且由采样控制器使用该比率映射以动态地计算通过参数域的循环的步长。

在某种意义上，采样率控制器是由常规光栅器执行的采样函数的泛化。所述采样控制器产生参数值阵列用作图形程序的自变量(例如，eval()方法)。在一些 GPU 体系结构中，采样控制器是可编程的，并且如上所述的采样程序是关于可如何使用控制器的非排他性的例子。一般地说，控制器能够在参数空间中在一表面上、或者沿着一条线或者笔触自适应地采样。取决于为图形程序所预备的采样过程，同一几何程序集合可以产生着色的呈现或者画线。

V．结果分析

取决于所估计的程序，图 2 中示出的 GPU 体系结构的计算要求通常在比常规流水线大一个至两个数量级的范围内。然而，图 2 的 GPU 体系结构的外部带宽需求通常比常规流水线小约一个数量级。部分地，较高的计算要求是由于形状程序的重复估计(而不是预处理)，对每个采样的取景变换相应的重复应用、以及使像空间中更有可能有足够的覆盖的过采样。还有与采样控制器的可编程性和缓冲器的路由网络相关联的电路面积（real estate）成本。然而，这些结果不必是排他的，并且本文所述的框架内的替换实现方式也可以产生改进的结果。

VI．替换方式

现在描述多个替换实施例。

(1)除了使用参照图 6 和 7 所述的自适应采样的双轮方法，GPU 体系结构可以使用另一机制。例如，GPU 体系结构可以使用具有独立的反馈数据路径的运行时速率控制机制。如果该体系结构不支持从 SIMD 阵列返回采样控制器的直接通信，那么这种反馈是间接的。

(2)除了过采样以使像空间中更有可能有足够的覆盖，GPU 体系结构也可以跳过或者减少过采样，然后在重构期间寻找间隙、缝隙或者其他呈现伪影。

(3)当使用贴片时，可以在逐个贴片的基础上向图 2 的 GPU 体系结构馈送程序。

或者，为了减少与在逐个贴片的基础上读取程序描述相关联的外部带宽，程序描述可以被高速缓存或者以其它方式存储。就写操作而言，如上所述，写操作可以被高速缓存到全帧缓存器而不是贴片缓存器，不管先前操作是逐个贴片的基础上还是在完整图像基础上执行的。

(4)虽然本文所述的技术特别针对程序性几何图形、纹理和着色而言的，但是或者，除此之外或作为其替换，还可使用产生图像样本的其他程序性描述。因此在一个例子中，可以描述和估计非超级现实主义（photorealistic）样式的程序，作为对常规着色的替换或加强。

VII. 计算环境

以上描述的技术和系统可以在任何多种计算设备和环境上实现，包括各种形式因素的计算机(个人、工作站、服务器、手持、膝上型、图形输入板、或者其他移动的)，分布式计算网络和网页服务，作为一些一般的例子。系统的各个部分可以以硬件电路以及以在例如图 8 中示出的计算机或者其他计算环境中执行的软件 880 实现。

图 8 示出了其中可以实现所述技术的适当的计算环境 800 的一般例子。所述计算环境 800 不旨在对所述技术和系统的使用范围或者功能提出任何限制，因为本发明可以在多种通用或者专用的处理环境中实现。

参照图 8 计算环境 800 包括至少一个处理单元 810 和存储器 820。在图 8 中，其最基本配置 830 被包括在虚线内。处理单元 810 执行计算机可执行指令。在多处理系统中，多个处理单元执行计算机可执行指令以增加处理能力。存储器 820 可以是易失性存储器(例如，寄存器、高速缓存、RAM)、非易失性存储器(例如，ROM、EEPROM、闪存等等)，或者这两者的组合。在一些实施方式中，处理器资源和存储器被配置为采样控制器、SIMD 阵列中的 ALU 和多个缓存器和高速缓存，如上参照图 2 所述。存储器 820 的一部分存储用于实现例如上述的程序性图形技术的软件 880。

计算环境可以具有其他特征。例如，计算环境 800 包括存储 840、一个或多个输入设备 850、一个或多个输出设备 860 以及一个或多个通信连接 870。互连机制(未示出)，诸如总线、控制器或者网络，使计算环境 800 的组件互连。

存储 840 可以是可移动的或不可移动的，并且包括可用于存储信息并可以在计算环境 800 内访问的任何介质。存储 840 可以存储软件 880 的指令。

输入设备 850(例如，作为控制点的设备)可以是触摸输入设备，或者是提供向计算环境 800 提供输入的另一个设备。所述输出设备 860 可以是提供来自计算环境 800 的输出的任何设备。

通信连接 870 允许通过通信介质与另一个计算实体通信。通信介质以已调制数据信号传送诸如计算机可执行指令、音频/视频或其它媒体信息、或者其他数据。已调制数据信号是以编码信号中的信息的方式设置或改变其一个或多个特征的信号。举例来说，而不是限制，通信介质包括以电、光、RF、红外、声学或者其他载波实现的有线或者无线技术。

本文所述的技术和工具可以在计算机可读介质的一般上下文描述。计算机可读介质是能够在计算环境内访问的任何可用的介质。举例来说，而不是限制，利用所述计算环境 800，计算机可读介质包括存储器 820、存储 840、通信介质，和任何上述的组合。

本文所述的技术和工具能够在计算机可执行指令的一般上下文描述，诸如程序模块中包括的、在目标真实或者虚拟处理器上的计算环境中执行的可执行指令。一般来说，程序模块包括例行程序、程序、方法、库、对象、类、组件、数据结构等等，它们执行特定的任务或实现特定的抽象数据类型。在多个实施例中，可根据需要，组合程序模块的功能或者在程序模块之间将功能分开。程序模块的计算机可执行指令可以在本地或者分布式计算环境内执行。

公开的技术和工具可以在各式各样的电路和系统(例如，专用集成电路(ASIC)、系统级芯片(SOC)、系统级封装(SIP)、系统上封装(SOP)、多芯片组件(MCM)、或者其他这种设备)中实现。所公开技术和工具的多个组件可以使用多种不同的半导体材料实现(单独地或者以不同组合地以及彼此的子组合)，这些材料包括但不限于：砷化镓(GaAs)和基于 GaAs 的材料(AlGaAs，InGaAs，AlAs，InGaAlAs，InGaP，InGaNP，AlGaSb 等等)；磷化铟(InP)和基于 InP 的材料(InAlP，InGaP，InGaAs，InAlAs，InSb，InAs 等等)；硅(Si)，应变硅，锗(Ge)和基于硅和锗的材料(SiGe，SiGeC，SiC，SiO2，高介电常数氧化物等等)，诸如互补金属一氧化物一半导体(CMOS)工艺；和氮化镓材料(GaN，

AlGaN，InGaN，InAlGaN，SiC，蓝宝石，Si 等)。在一个例子中，程序性图形结构是在单个芯片上实现的。所公开的技术和工具也可以使用这些工艺技术的组合来实现(例如，在多个芯片或者单个芯片上)。所公开的技术和工具还可以使用多种不同的芯片外工艺来实现，包括但不限于低频或者高频印刷电路板(PCB)工艺、厚膜或者薄膜混合工艺、多层有机工艺，和低温共烧陶瓷(LTCC)工艺。

类似地，多种晶体管技术可用于实现所公开的技术和工具。例如，所公开的程序性图形体系结构技术和工具可以使用双极晶体管(BJT)技术实现(例如，异质结双极晶体管(HBT))或者场效应晶体管(FET)技术(例如，假晶高电子迁移率晶体管(pHEMT))实现。还可以使用这些技术或者其他晶体管技术的组合或者子组合来实现所公开的技术和工具。这种组合可以在多个芯片或者单个芯片上实现。例如，在一个示例性的实施例中，一个或多个 pHEMT 晶体管是在与一个或多个异质结双极晶体管(HBT)相同的芯片上实现的。

鉴于可以本发明的原理可应用的许多可能的实施例，要求将本发明所有这种实施例都落入如下权利要求与其等效方式的范围和精神之内。

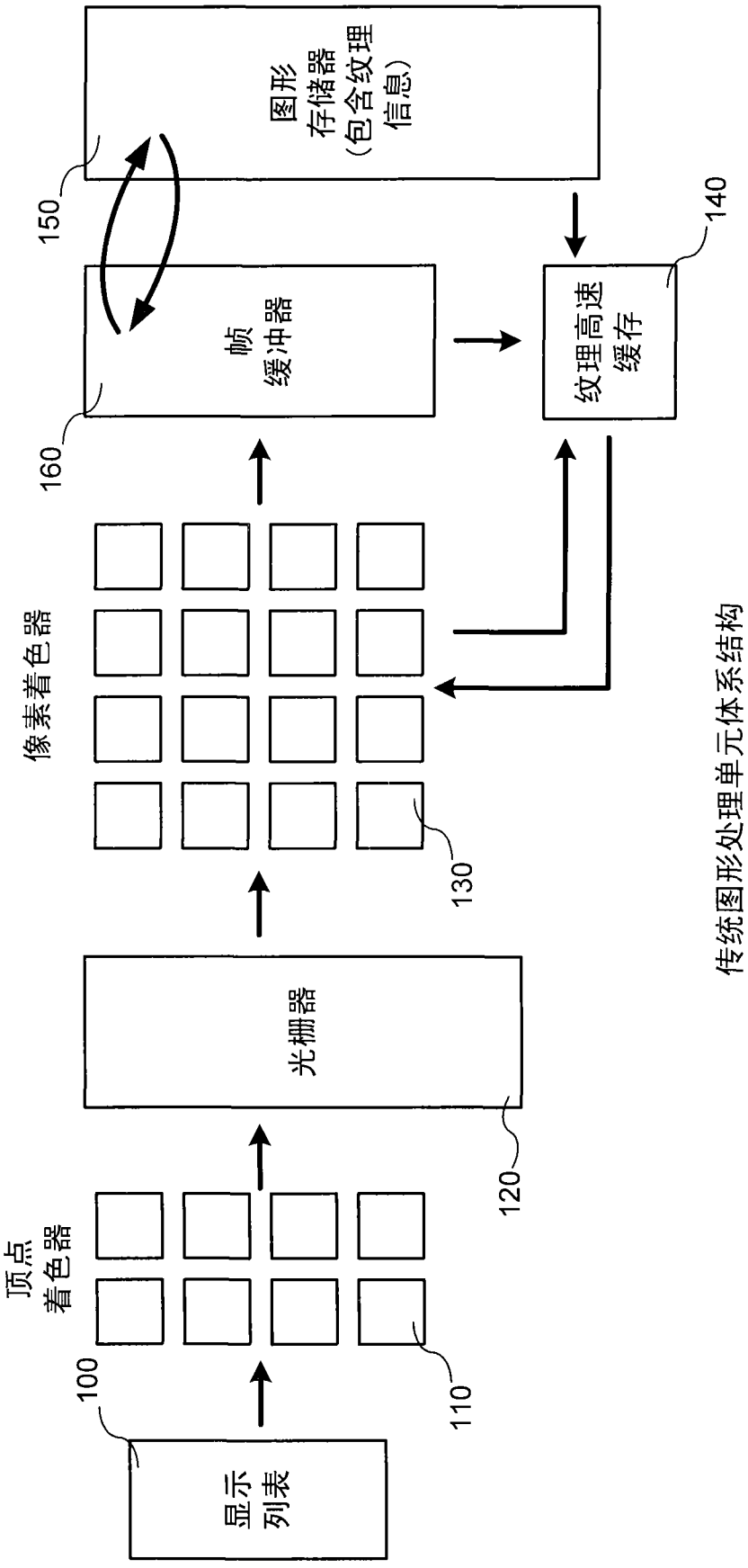

显示列表
顶点着色器
光栅器
像素着色器
帧缓冲器
图形存储器（包含纹理信息）
纹理高速缓存
100
110
120
130
140
150
160
传统图形处理单元体系结构

图 1
现有技术

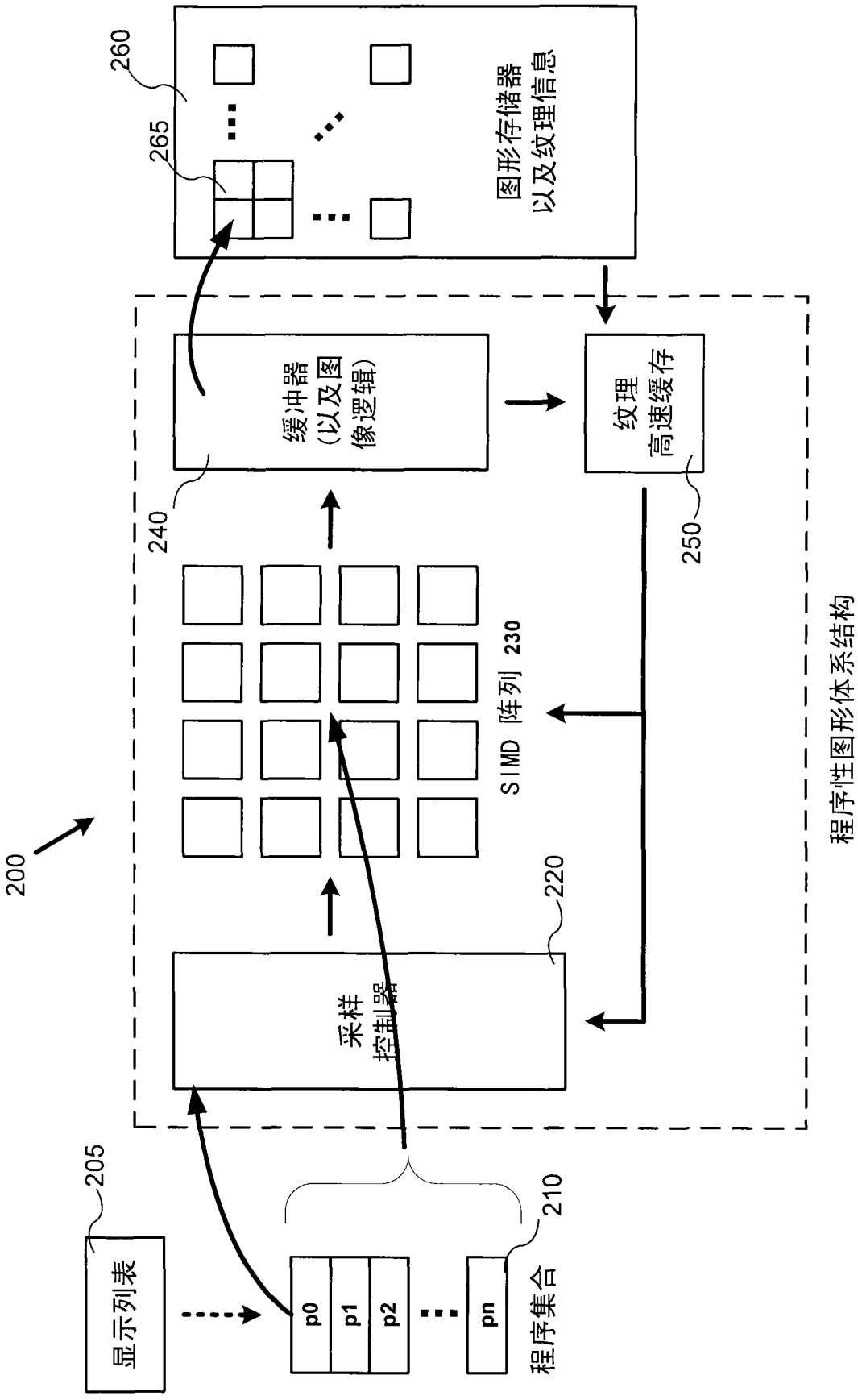

260
265
图形存储器
以及纹理信息
缓冲器
(以及图
像逻辑)
240
纹理
高速缓存
250
SIMD 阵列 230
200
220
采样
控制器
程序性图形体系结构
205
显示列表
p0
p1
p2
pn
210
程序集合

图 2

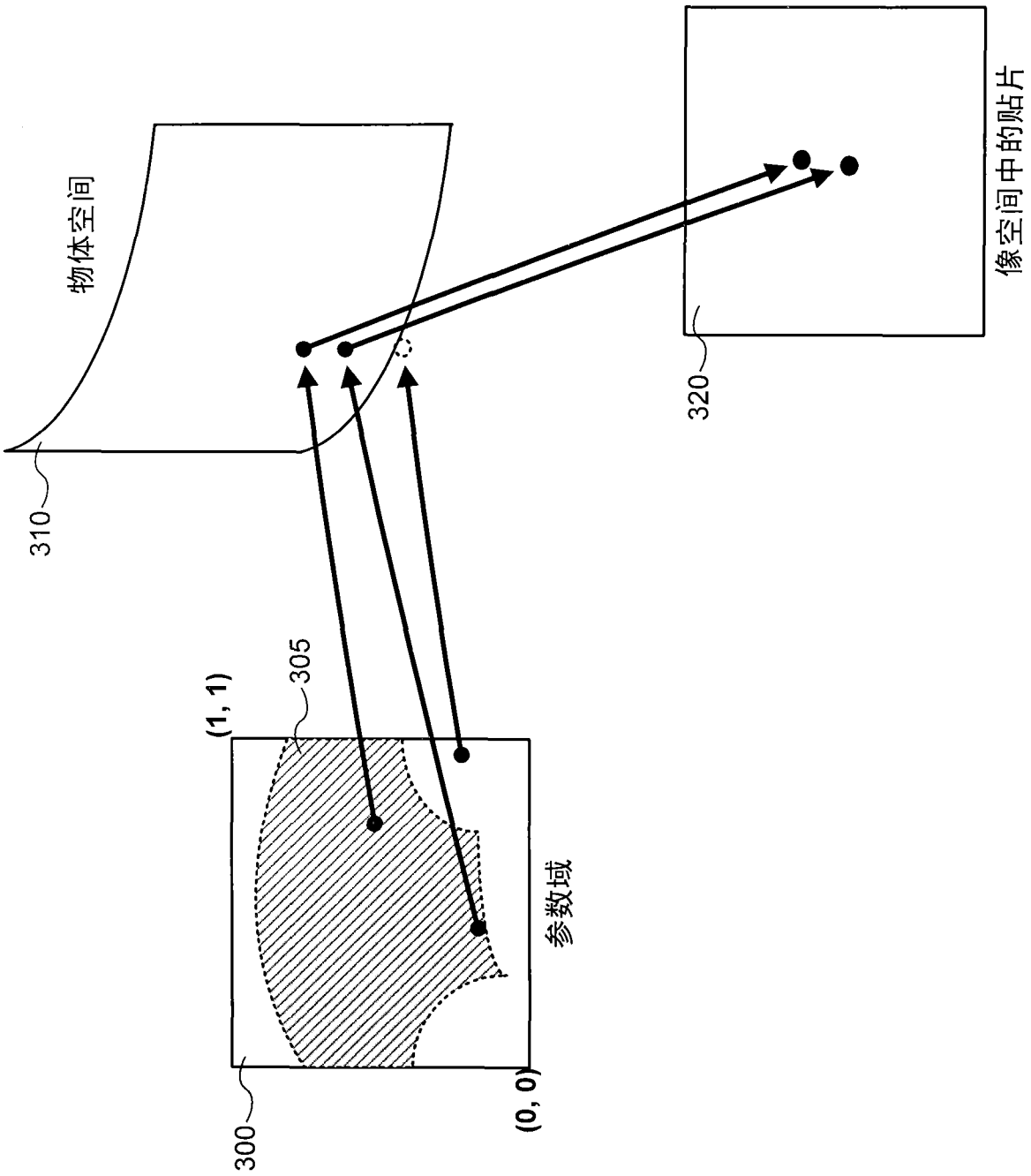


图 3

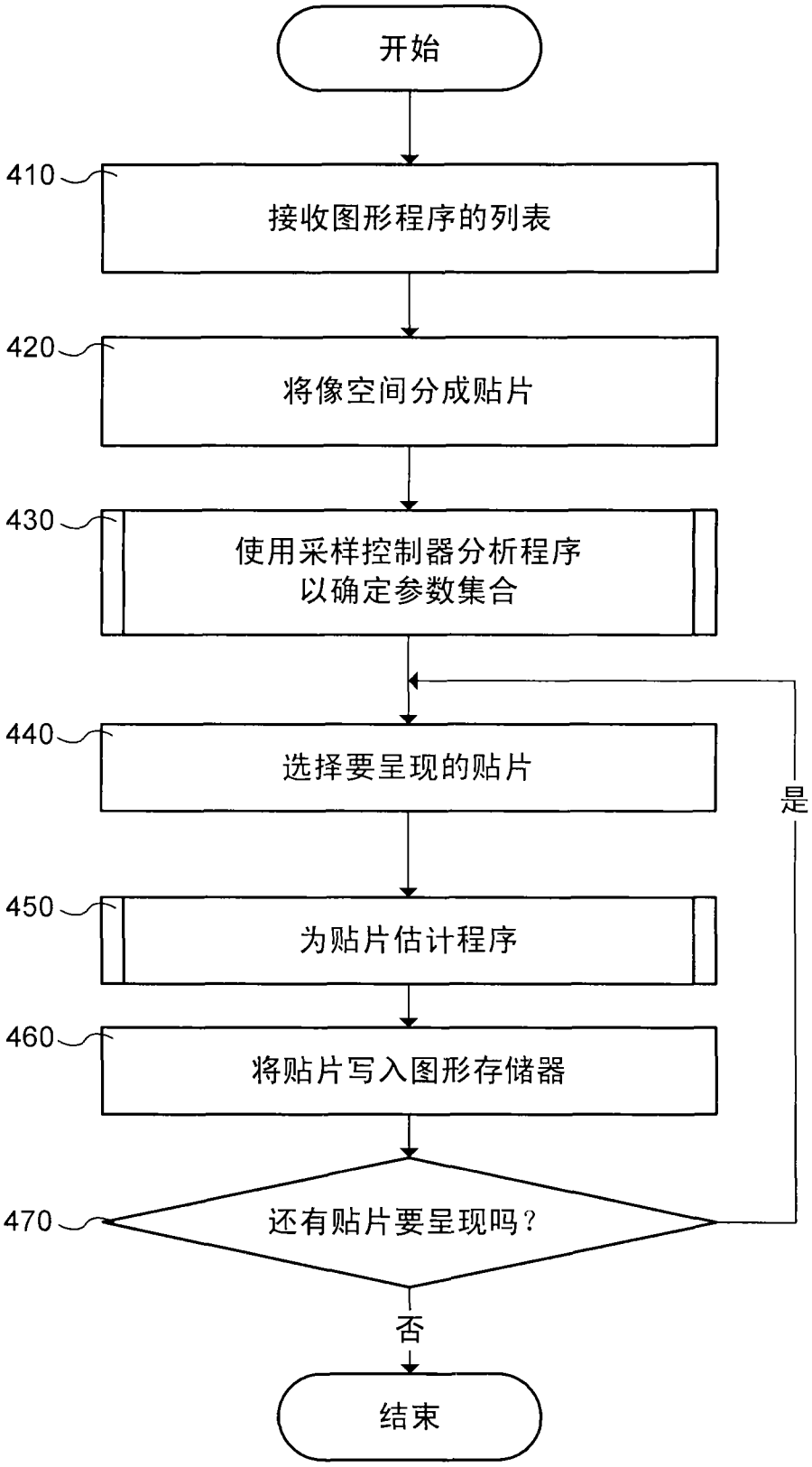

开始
410
接收图形程序的列表
420
将像空间分成贴片
430
使用采样控制器分析程序
以确定参数集合
440
选择要呈现的贴片
450
为贴片估计程序
460
将贴片写入图形存储器
470
还有贴片要呈现吗？
是
否
结束

图 4

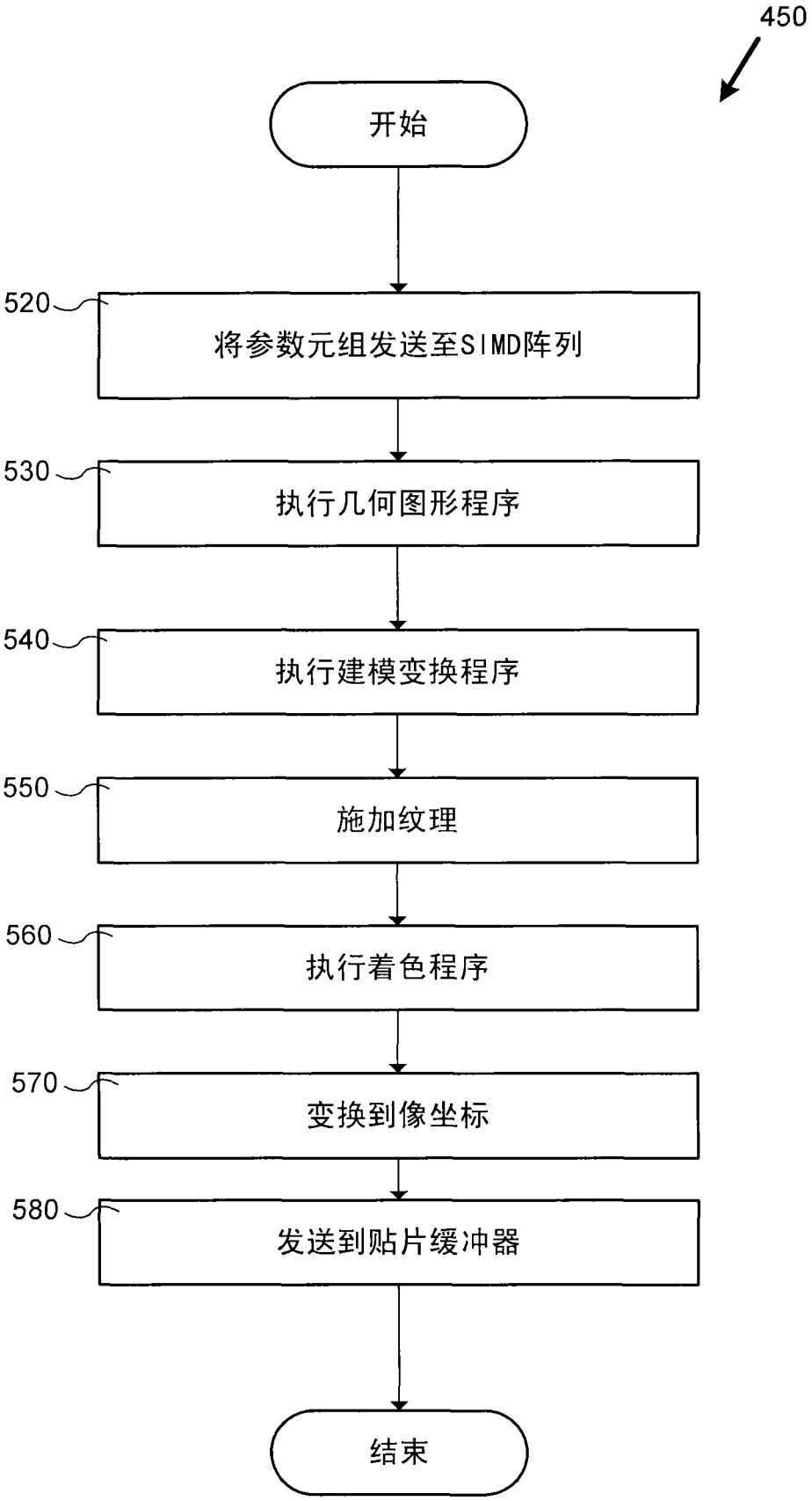

450
开始
520
将参数元组发送至SIMD阵列
530
执行几何图形程序
540
执行建模变换程序
550
施加纹理
560
执行着色程序
570
变换到像坐标
580
发送到贴片缓冲器
结束

图 5

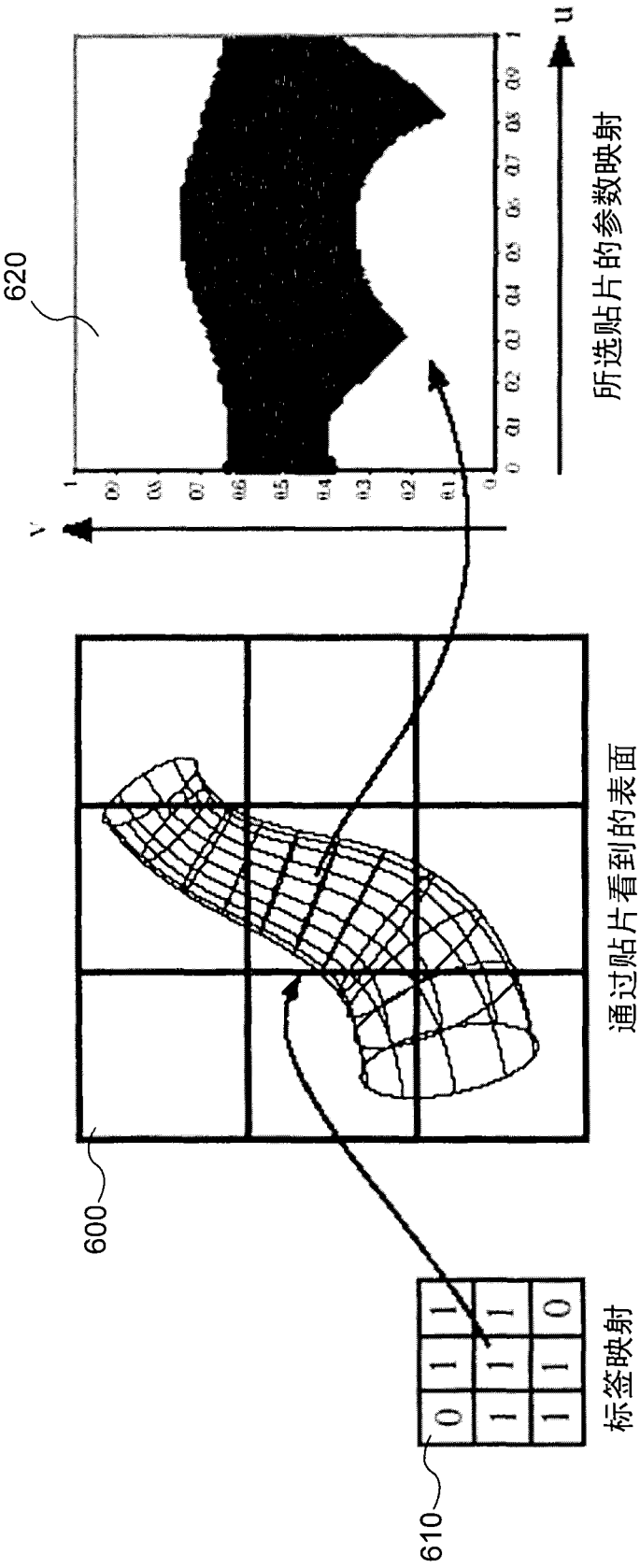


图 6

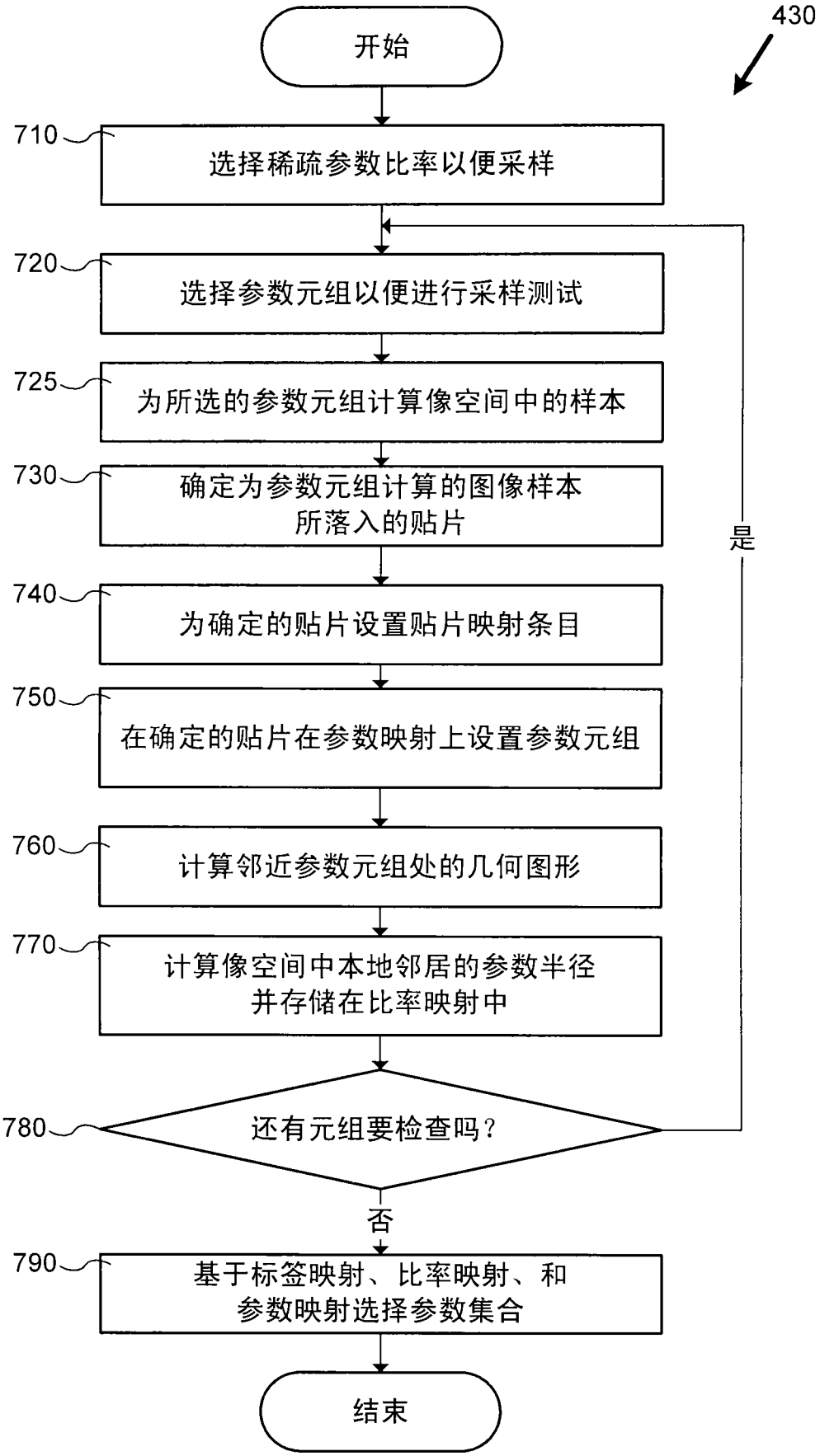

430
开始
710
选择稀疏参数比率以便采样
720
选择参数元组以便进行采样测试
725
为所选的参数元组计算像空间中的样本
730
确定为参数元组计算的图像样本
所落入的贴片
740
为确定的贴片设置贴片映射条目
750
在确定的贴片在参数映射上设置参数元组
760
计算邻近参数元组处的几何图形
770
计算像空间中本地邻居的参数半径
并存储在比率映射中
780
还有元组要检查吗？
是
否
790
基于标签映射、比率映射、和
参数映射选择参数集合
结束

图 7

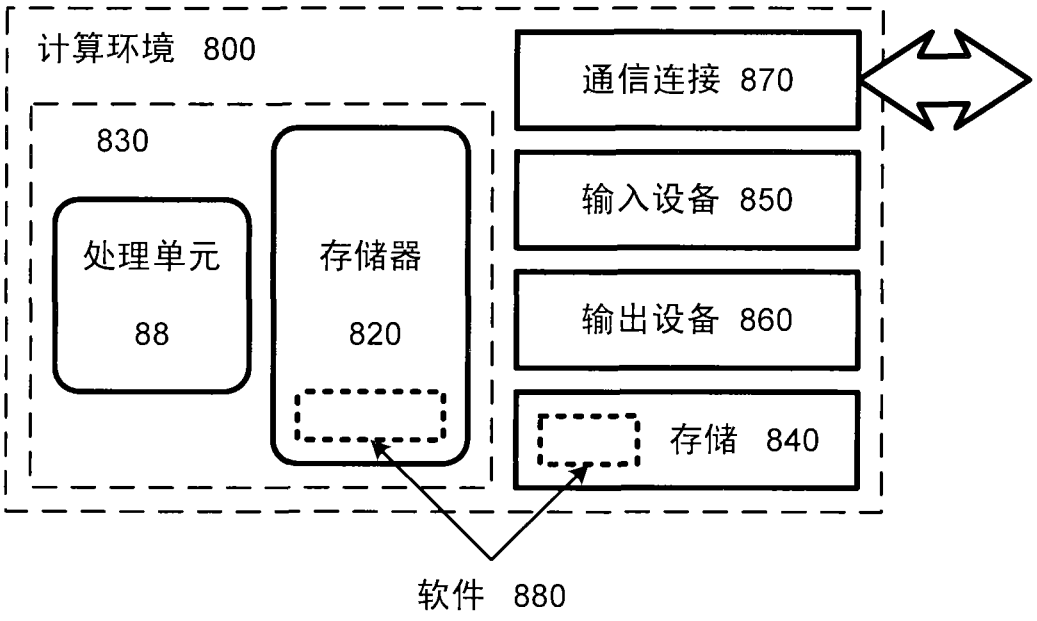

计算环境 800
830
处理单元
88
存储器
820
通信连接 870
输入设备 850
输出设备 860
存储 840
软件 880

图 8