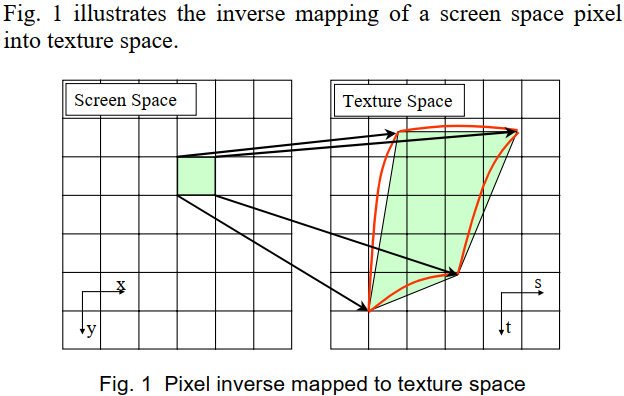
纹理映射是计算机图形图像生成的基本功能。 在当前基于PC的加速硬件中，具有双线性和三线性滤波的MIP映射是减少空间混叠伪像的常用滤波技术。 这些技术以减少模糊为代价减少图像混叠的有效性取决于MIP贴图级别选择以及屏幕空间到纹理空间像素缩放的相关计算。 本文介绍了针对每个像素和每个原始细节计算级别的实用方法的研究。 此调查是屏幕空间光栅化ASIC设计工作的一部分。 讨论了具有相当视觉质量的几种算法的实现，并提供了按原始值和按像素计算成本的比较。

1 介绍

纹理映射已成为当今计算机图形学中最重要的操作之一。使用纹理贴图可以将高度的视觉复杂性应用于3D场景，而无需花费过于复杂的几何模型。一旦纹理贴图过程到位，就可以以恒定的性能成本应用任意表面细节。纹理表面上的每个点都映射到纹理图中的某个位置，可以从该位置检索一个可用于修改颜色，反射率，透明度或表面法线的值。 Heckbert [12]将纹理过滤定义为将纹理图像重新采样到屏幕网格上的过程。每个屏幕坐标x，y映射到纹理空间坐标s，t。对于每个离散的屏幕元素或像素，滤波操作确定哪些离散的纹理空间像素或纹理像素将起作用。在本文中，我们关注在光栅化硬件环境中作为基于投影屏幕的渲染器一部分的纹理过滤操作。在这种情况下，过滤操作在透视投影的三角形和2D图像图的面之间进行。

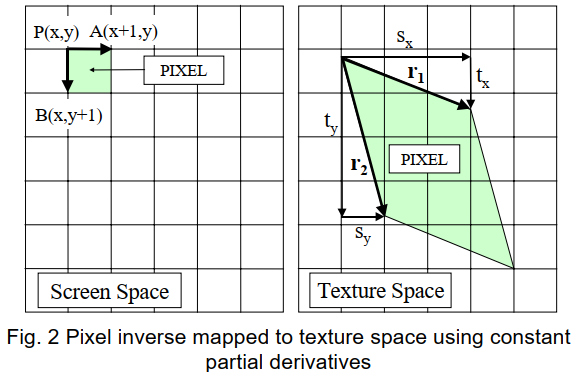


从图1可以看出，像素的形状和比例在映射到纹理空间时都会改变。 纹理映射的整个操作，包括将纹理图像收缩包装[1]到不规则物体上的过程，可能会产生曲线四边形，如红色所示[25]。 但是，出于实用性考虑，它近似于如图所示的平面四边形，其纹理线性映射到形成表面的多边形[20]。基于屏幕像素的正方形表示，Heckbert [13]提出可以假定大多数表面在像素覆盖的区域内大致是平面，并且基于此假设，屏幕空间像素将映射到纹理空间中的四边形。

屏幕空间像素表示为正方形仅是一种解释。像素实际上是一个点样本，在正方形表示的情况下，四个顶点本身只是样本值。对像素使用离散的有限区域并不是显示技术的特别准确的模型[21]，但是这样做是为了方便。虽然纹理过滤算法也已经基于包括圆形在内的其他形状表示[11,12]，但正方形表示很方便，这是本文中使用的表示。

在图1中可以看出，几个纹理像素可以映射到一个屏幕空间像素。着色像素轮廓下方的每个纹理像素的颜色应有助于最终像素的颜色。此纹理空间与屏幕空间缩放的比率称为纹理最小化[minification]。这种纹理的缩小是由于映射到不规则物体和透视投影而产生的[2,13]。纹理缩小率可以在每个像素的整个投影表面上变化。因此，纹理滤镜的任务是确定哪些纹理像素有助于每个像素。这样的滤波器必须是空间可变的，即，滤波器的尺寸和形状应适应于每个像素的映射。第一步是有效地估计像素逆映射。这可以通过三种方式执行：

1. 四个偏移像素角的直接寻址-可以准确印象出所产生的不规则四边形，因此考虑了子像素缩小的变化。 但是，它确实需要四个单独的映射操作。
2. 使用相对于和的和的偏导数—假设一组偏导数在像素区域内是恒定的,并导致平行四边形近似.实际上，这些导数值将在子像素级别连续变化.但是，这种近似简化了计算，并且通常用于滤波算法中。Heckbert [13]提出，除了接近消失点和奇点以外，平行四边形是一个合理的近似值。这种映射技术的结果如图2所示。
3. 像素裁剪—一种替代但相当昂贵的方法涉及将纹理多边形裁剪到屏幕像素边界[24]。然后，计算得到的裁剪区域的每个顶点的纹理坐标，并将其用于确定像素的平均纹理样本地址。剪切区域的面积可以用于确定纹理缩小率。



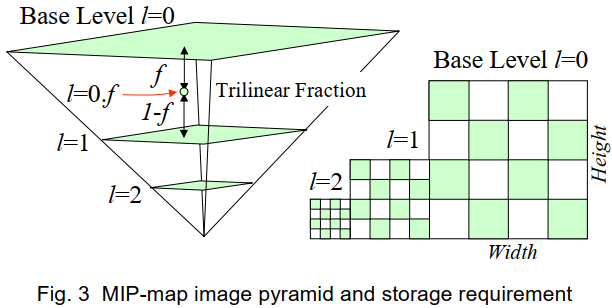
* 1. 纹理过滤

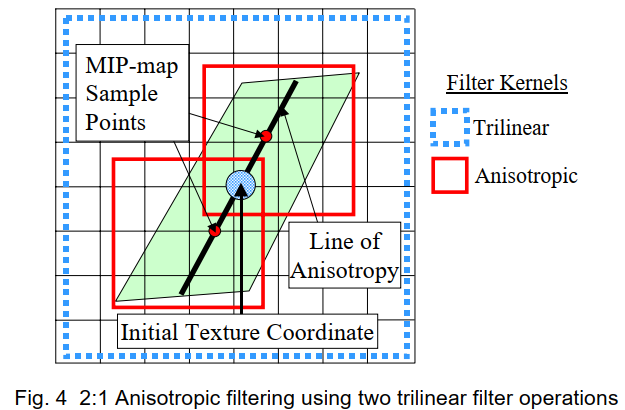
已经开发并实现了几种高质量的空间变量滤波算法[3,8,9,10,11]，但是这些算法在计算上往往非常昂贵，根据每个像素的缩放比例会产生任意映射。纹理过滤需要访问纹理内存，这是一项耗时的操作，应将其保持在最低限度，并且在可能的情况下保持不变的成本。为此，开发了利用预过滤纹理存储的过滤算法。图像金字塔或MIP贴图是减少滤波器构造期间降低每像素纹理访问成本的常用方法。 MIP-map一词由Williams提出[26]。 MIP一词代表细小变化，或“小地方很多”，是指对分辨率递减的多个纹理贴图的预过滤和存储，它们试图在越来越小的空间中包含相同的信息。这是通过将四个相邻纹理像素的组逐渐平均以形成图像金字塔的每个新层来完成的。从初始的完整细节或基础纹理级别（称为级别0）继续进行此过程，直到达到最终的单个纹理像素级别。每层代表的纹理最小化是基于侧面尺寸，而不是面积。初始纹理尺寸必须是2的幂。图3说明了图像金字塔和纹理存储要求。

金字塔中的每个MIP映射级别都是基本级别的两个缩放版本的幂。就MIP映射生成和级别计算而言，这对于硬件实现都是非常方便的。尽管如图3所示，但是基本纹理的宽度和高度不必相等。

对于纹理过滤，将MIP贴图与双线性和三线性插值结合使用[26]。对于双线性滤波，计算纹理空间到屏幕空间的最小化，并用于选择MIP贴图级别，该级别包含预滤波的纹理像素数据，且缩小比最接近计算得出的缩小率。此过程也称为详细程度计算。通常，采用此技术的系统将设置最大MIP映射级别。几乎没有必要使用所有可能的级别，直到1x1纹素图像。这种水平钳位的使用避免了在消失点和奇点处可能出现的缩小比率大的问题。在级别选择之后，将纹理坐标s，t缩放到所选级别的尺寸，并适当偏移存储地址。然后使用最接近采样点的四个纹理像素执行双线性插值。如图3所示，计算得出的缩小值通常会落在两个预先过滤的MIP映射级别之间。为了改善结果，使用了三线性滤波。在此，在计算出的最小值的两侧的两个级别上都执行双线性插值。然后，通过在这两个双线性插值产生的颜色值之间线性插值到中间插值分数f,可以实现三线性插值。三线性过滤是当今实时硬件中最常用的过滤方法之一。过滤操作可以总结如下:

1. 当前屏幕像素x，y接收纹理地址s，t.
2. 计算纹理最小化j
3. 提取要使用的细节级别或MIP映射级别,例如,对于6到1的缩小比例，将使用参考缩小比例4和8的级别,即MIP映射级别2和3（l = 2）
4. 计算三线性插值分数f
5. 缩放纹理地址s，t为所选级别
6. 执行三线性插值





MIP映射预滤波和三线性插值基于各向同性正方形滤波器形状。过滤器的尺寸是空间变化的，但形状不是。在图1和图2中可以看出，像素的映射实际上是各向异性的。也就是说，屏幕到纹理空间在s和t轴上的缩放比例不相等。因此，这样的各向同性滤波器将减少在一个轴上引起的混叠，但以另一轴上的一些模糊为代价。 Crow [5]建议使用求和面积表，以允许映射到任意矩形。虽然返回了改进的结果，但存储要求高于Williams的MIP地图金字塔，并且对于倾斜到45o的纹理，滤镜形状仍是各向同性的。除了基于3-D正方形的MIP-map图像金字塔外，还使用了由轴向对齐的矩形形成的4-D图像金字塔[14]。对于椭圆滤波器，也有人建议使用包含旋转信息以支持任意对齐的5-D金字塔[14]，但同样，存储要求很高。 Schilling [18]提出的一种有效且实用的方法是，通过沿着各向异性方向执行多个三线性滤波操作，将空间方差与MIP映射结合起来。该滤波器的操作如图4所示。

图13a和图13b分别说明了单个各向同性三线性滤波器的结果以及后一种多向同性各向异性纹理滤波的实现[7]。对此类各向异性滤波器的详细研究以及对它们可能的实现的讨论超出了本文的范围，本文主要涉及对单个三线性滤波器的实际实现的研究。但是，应该注意的是，用于这种各向异性滤波器的MIP映射级别选择算法在操作和成本上与此处讨论的相似。在本文的以下各节中，第2节介绍了用于计算纹理最小化的各种算法。包括的所有方法均在C ++仿真环境中实现[6]，以比较结果。这样就可以在整个图形流水线的上下文中对算法进行测试。提供了用于实现这些算法的相对计算成本的表。然后，第3节讨论了从计算的纹理最小化值中提取MIP贴图级别和三线性插值分数的硬件方法，最后，第4节总结了从研究得出的结论。

2 计算纹理最小化比率

本节详细讨论确定纹理最小化比率的方法，包括每个像素和每个基本算法。这项研究首先着眼于由Heckbert推荐的众所周知的斜边比较算法[13]。然后进行一个简单得多的比较测试，比较结果和成本。还研究了几种计算偏导数和的不同方法。首先，考虑代数解，并讨论直接计算和增量插值。然后再看一下利用像素相干性的差异计算。重要的是要记住，在考虑这些不同的解决方案时，对于各向同性滤光片来说，它们都是近似值，基于像素表示本身就是一个近似值，这都是近似值。没有正确或错误的选择。OpenGL规范[19]认识到了这一点，该规范提出了一系列可能的方法，而不是推荐一种特定的方法。归根结底，与实现和性能成本相关的是，尤其是在动画中产生的图像的感知质量。图15比较了将以下各节中讨论的每种主要算法应用于带有病理棋盘图案的透视投影平面的纹理化的结果。

2.1 斜边比较

如前所述，实际上，纹理最小化不会同时在和方向上均等发生.因此,对于各向同性的MIP映射滤波,必须确定一个最小化数字以提供最佳近似.最小化值太大,图像将变得过分模糊,太小并且无法消除锯齿.赫克伯特（Heckbert）的前提是，过渡过滤比欠过滤更为可取。尽管在他的论文中提出了几种不同的方法[13]，但他确定选择投影平行四边形的两个边缘中较大的一个作为最佳解决方案。这可以从图2所示的斜边方向矢量中找到:

其中且.因此

表示最小化比率.

因此,该表达式的计算成本为4乘法，2加法，1平方根和单个比较.此外,还需要计算偏导数和的成本.这些将在以下小节中讨论.

2.1.1 偏导数代数解

由于透视正确的纹理映射的非线性性质，相对于x和y的s和t的变化会在每个像素的基础上发生变化.为了进行透视校正,通过双曲线插值法生成纹理坐标s和t [2,13].非线性纹理映射坐标是两个线性运算的函数，如公式2所示：

其中.

项A到I是S，T和Q三个独立线性表达式的系数。这是从描述该平面的三个三角形顶点中的每个顶点的值计算出的平面方程。S和T是均化到视图平面的顶点纹理坐标，Q是深度Z的倒数的函数。

在对等式2进行偏微分之后,偏导数为:

项对于这些表达式中的每一个都是通用的,可以从比较测试中排除.因此,最终的每像素斜边比较计算可写为:

如等式2所示,对于纹理坐标的透视校正,需要Q的倒数.因此,高于点采样纹理映射所需的每像素计算成本为6加/减,10乘法，1平方根和1最大比较.

可以看出,由于可以分解出Q,所以方向矢量和相关分量偏导数和仅取决于,方向矢量和偏导数和仅取决于.这个事实可以用来简化实现.假设扫描转换实现的原始遍历机制在任何一次仅沿方向(跨度)或方向(扫描线遍历)移动,则和计算以及和计算是独立的.因此,在硬件中,假设存储不变值需要一些额外的逻辑,则相同的算术单元可以使用两次.就所需功能单元而言,这将每像素成本降低了4倍,3次加法和1平方根.

Heckbert[13]提出了另一种节省方法,他建议针对基元的x范围对r2进行每个基元预计算和查找表（LUT）存储，然后对每条扫描线进行r1计算，以最小化每个像素成本。 这种方法的有效性取决于所用三角形的大小.