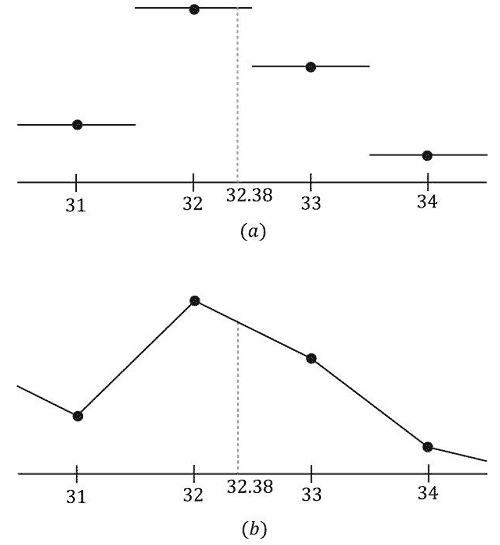
# 8.4 过滤器

## 8.4.1 倍增

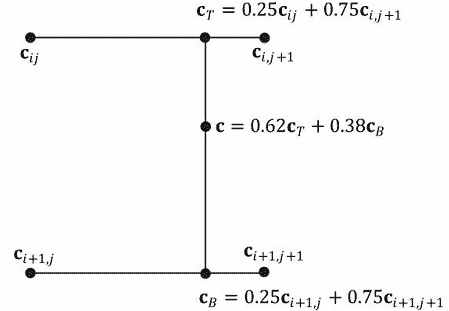
纹理贴图元素应该被视为在一个连续图像上的离散颜色采样；不应该被视为矩形区域。那么问题是：当我们指定的纹理坐标(*u*,*v*)与任何一个纹理元素点都不对应时会产生什么结果？这一问题会发生在如下情景中：当观察点离场景中的一面墙很近时，墙会被放大，以至于会盖住整个屏幕。如果显示器的分辨率为1024×1024，墙体纹理的分辨率为256×256，那么就会出现倍增（magnification）问题——我们将要用很少的纹理元素来覆盖很多的像素。在本例中，每个纹理元素要覆盖4个像素。当顶点纹理坐标在三角形表面上进行插值时，每个像素都会得到一对唯一的纹理坐标。所以，像素的纹理坐标不会与任何一个纹理元素点对应。 我们可以使用插值方法来估算纹理元素之间的颜色。图形硬件提供了两种插值方法：常量插值和线性插值。线性插值是我们最常用的插值方法。

图8.5说明了在1D空间中的插值方法：假设有一个包含256个采样点的1D纹理以及一个插值纹理坐标*u* = 0.126484375。那该纹理坐标对应的纹理元素为0.126484375×256 = 32.38。当然，这个值位于两个纹理采样点之间，我们必须使用插值来估算它。

****

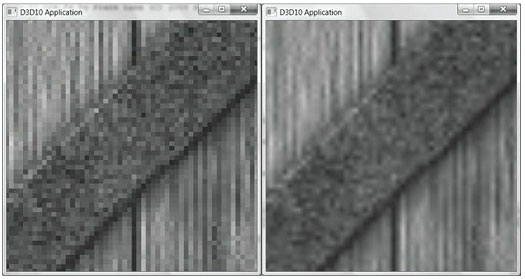
**图8.5 (a)给出纹理元素点，我们构造一个分段常量函数来估算纹理元素点之间的值；这种方法有时也称为最近邻接点采样（nearest neighbor point sampling），因为它总是用最近的纹理元素点的值作为采样结果。(b)给出纹理元素点，我们构造一个分段线性函数来估算纹理元素点之间的值。**

2D线性插值也称为双线性插值（bilinear interpolation），如图8.6所示。给出一对位于4个纹理元素之间的纹理坐标，我们在*u*方向上进行两次1D线性插值，然后在*v*方向上进行一次1D线性插值。

****

**图8.6 这里有4个纹理元素点cij、ci，j+1,ci+1，j、ci+1，j+1。我们想要估算c点的颜色，它位于这4个纹理元素点之间。在本例中，c位于cij右侧0.75单位、cij下方0.38单位处。我们先在上面的两个颜色之间进行1D线性插值得到cT。然后，在下面的两个颜色之间进行1D线性插值得到cB。最后，在cT和cB之间进行线性插值得到c。**

图8.7说明了常量插值和线性插值之间的区别。可以看到，常量插值会使图像出现明显的块状。而线性插值较为平滑，但是与真实数据（例如，一幅高分辨率的纹理）相比，通过插值得到的衍生数据仍然不够理想。

****

**图8.7 我们在一个立方体放大一幅板条箱纹理，使倍增问题出现。左图使用常量插值，可以看到它产生了明显的斑块；这很容易理解，因为插值函数是离散的（图8.5a），它所形成的颜色过渡显得生硬而不平滑。右图使用线性插值，由于插值函数是连续的，所以它产生的图像较为平滑。**

需要强调的是，由于在交互式3D程序中观察点的位置可以自由移动，所以我们对于倍增问题没有一种非常彻底的解决办法。在一定距离内，纹理可以有较好的效果，但是当观察点与物体之间的距离越来越近时，效果会急转直下。使用更高分辨率的纹理可以缓解一问题。

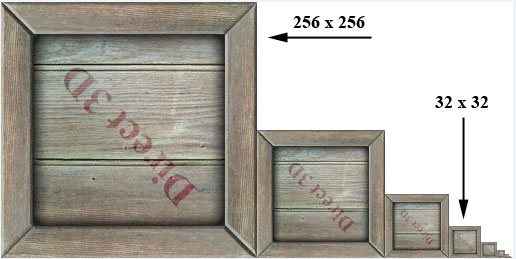
**注意**：在纹理映射的环境下，使用常量插值来求解纹理元素之间的颜色值的过程也称为点过滤（point filtering），使用线性插值来求解纹理元素之间的颜色值的过程也称为线性过滤（linear filtering）。点过滤和线性过滤是Direct3D使用的术语。

## 8.4.2 缩减

缩减（minification）与倍增的情况恰好相反。在缩减中，较多的纹理元素会被映射为较少的像素。例如，考虑下面的情景：我们将一幅256×256的纹理映射到墙体上。然后把观察点对准墙体，并向后移动观察点，使墙体逐渐变小，直到墙体在屏幕上只占64×64的像素区域为止。现在，我们要把256×256个纹理元素映射为64×64个屏幕像素。在这一情景中，像素的纹理坐标不会与纹理贴图中的任何一个纹理元素对应。常量和线性插值过滤器仍然适用于缩减情况。不过，缩减的处理工作稍多一些。简单的讲，我们要把256×256个纹理元素均匀地缩减为64×64个纹理元素。多级渐近贴图映射（mipmapping）为这一工作提供了一种高效的估值手段，只是要额外占用一些内存。在初始化时（或创建资源时），通过对图像进行降阶采样生成纹理的多个缩略版本来创建多级渐近纹理链（mipmap chain，参见图8.8）。求平均值的工作是根据多级渐近贴图的大小提前计算出来的。在运行时，图形硬件会根据程序员指定的多级渐近贴图参数执行两种不同的操作：

1．为纹理映射挑选一个与屏幕几何体分辨率最匹配的多级渐近纹理层，根据需要在多级渐近纹理层上使用常量插值或线性插值。这种用于多级渐近纹理的操作称为点过滤（point filtering），因为它与常量插值很像——它只为纹理映射挑选一个最接近的多级渐近纹理层。

2．为纹理映射挑选两个与屏幕几何体分辨率最匹配的多级渐近纹理层（其中，一个比屏幕几何体分辨率大一些，另一个比屏幕几何体分辨率小一些）。然后，在这两个多级渐近纹理层上使用常量插值或线性插值，分别取出一个纹理颜色。最后，在这两个纹理颜色之间进行插值。这种用于多级渐近纹理的操作称为线性过滤（linear filtering），因为它与线性插值很像——它在两个最匹配的多级渐近纹理层之间进行线性插值。

****

**图8.8 多级渐近纹理链；每个后续多级渐近纹理层的尺寸都是前一层的1/2，纹理层的下限尺寸是1×1。**

通过在多级渐近纹理链中选择最佳纹理层，可以使缩减操作的总开销降至最低。

### 8.4.2.1 创建多级渐近纹理

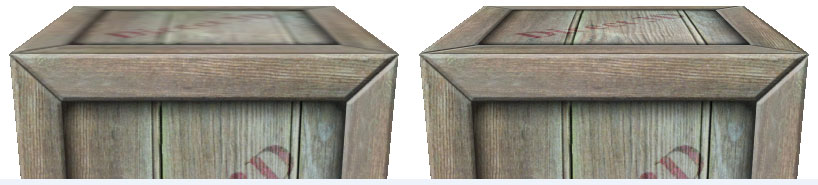
多级渐近纹理层即可以由美术师手工创建，也可以由过滤算法生成。

DDS图像格式（DirectDraw Surface格式）可以直接把多级渐近纹理层存储在文件中；在这种情况下，只需要简单的读入数据——不需要在运行执行任何与多级渐近纹理层相关的算法。DirectX纹理工具可以为纹理生成多级渐近纹理链，并导出DDS文件。当一个图像文件不包含完整的多级渐近纹理链时，函数**D3DX11CreateShaderResourceViewFromFile**或**D3DX11CreateTextureFromFile**会使用指定的过滤算法创建一个多级渐近纹理链（请参阅 SDK文档中**D3DX11\_IMAGE\_LOAD\_INFO**，尤其是要注意**MipFilter**数据成员的描述）。我们可以看到，多级渐近纹理映射基本上是自动的。当资源文件不包含多级渐近纹理链时，**D3DX11**会为我们自动生成一个纹理链。只要启用了多级渐近纹理映射，硬件就会在运行时自动选择正确的多级渐近纹理层。

**注意**：有时通用的过滤算法可能会丢失你想要保留的细节。例如，在图8.7中，木板上的文字“Direct3D”在最低等级的渐进纹理中非常模糊。如果这不可接受，美工就需要手动创建/调整渐进层次以保留重要的细节。

## 8.4.3 各向异性过滤

我们还可以使用各向异性过滤（anisotropic filter）。当多边形的法线向量与摄像机的观察向量夹角过大时（例如，当多边形垂直于观察窗口时），这种过滤可以有效缓解图像的失真问题。只是它的资源占用量较大。不过，当你需要对失真进行校正时，这些消耗是值得的。图8.9是对各向异性过滤和线性过滤的一个比较。

****

**图8.9 板条箱的顶面几乎垂直于观察窗口。（左图）使用线性过滤，板条箱的顶面显得极其模糊。（右图）在同样的角度上，使用各向异性过滤得到的渲染结果要好很多。**