尽管像pbrt这样的渲染器的最终输出是彩色像素的2D网格,但是入射辐射实际上是在胶片平面上定义的连续函数.根据此连续函数计算离散像素值的方式会明显影响渲染器生成的最终图像的质量,如果不仔细执行此过程,将出现伪像.相反,如果执行良好,则为此目的进行的相对少量的附加计算可以显着提高渲染图像的质量.

本章首先介绍采样理论,即从连续域中定义的函数中获取离散样本值,然后使用这些样本来重构与原始函数相似的新函数的理论.本章中定义的采样器基于采样理论的原理以及低分布点集的思想(低离散点集是分布均匀的采样点的一种特殊类型),以各种方式生成维采样向量.本章描述了五种采样器实现以及各种解决采样问题的方法.

本章以Filter类和Film类作为结束.Filter用于确定每个像素附近的多个样本如何混合在一起以计算最终像素值,而Film类将图像样本贡献累加到图像的像素中.

7.1 采样理论

数字图像表示为一组像素值,通常在矩形网格上对齐.当数字图像显示在物理设备上时,这些值将用于确定显示器上像素发射的光谱功率.考虑数字图像时,重要的是要区分代表特定样本位置上的函数值的图像像素和显示像素,它们是发射具有一定分布光的物理对象.(例如,在LCD显示器中,当以倾斜角度观看显示器时,颜色和亮度可能会发生很大变化.)显示器使用图像像素值在显示表面上构造新的图像函数.此函数定义在显示器的所有点上,而不仅仅是数字图像像素的无穷小点.收集样本值并将其转换回连续函数的过程称为重构.

为了计算数字图像中的离散像素值,必须对原始连续定义的图像函数进行采样.与大多数其它光线追踪渲染器一样,pbrt获取有关图像功能的信息的唯一方法是通过追踪光线对其进行采样.例如,没有通用的方法可以计算胶片平面上两点之间图像功能变化的界限.虽然仅通过在像素位置精确地采样函数就可以生成图像，但是可以通过在不同位置进行更多采样并将与图像函数有关的此附加信息合并到最终像素值中来获得更好的结果。实际上，为了获得最佳质量结果，应该计算像素值，以使显示设备上的重建图像尽可能接近虚拟相机的胶片平面上的场景原始图像。请注意，这与期望显示器的像素在其位置上采用图像功能的实际值有微妙的不同。处理这一差异是本章实现的算法的主要目标.

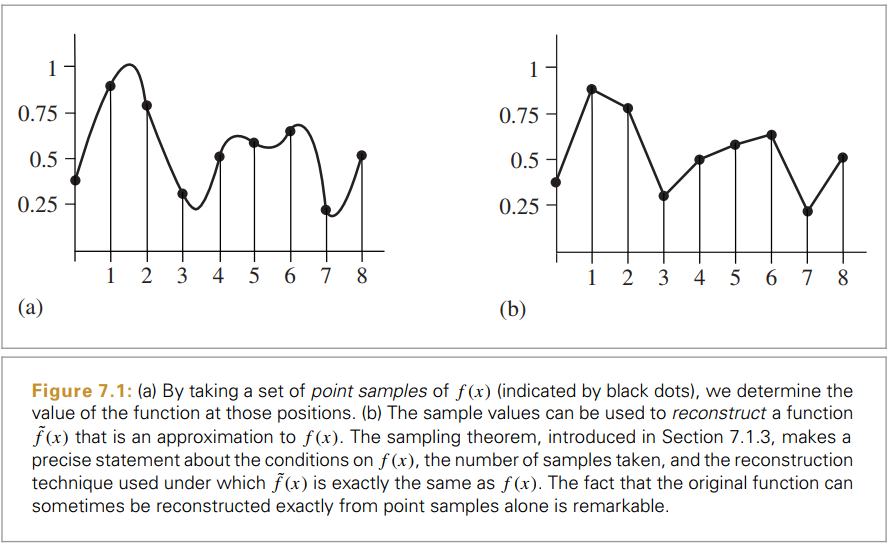
在本书中，我们将忽略与物理显示像素特性有关的问题，并将在显示器执行本节稍后部分介绍的理想重构过程的假设下工作。 这种假设显然与实际显示的工作方式不符，但可以避免此处不必要的分析复杂化。 Glassner（1995）的第3章对非理想的显示设备及其对图像采样和重建过程的影响进行了很好的论述。

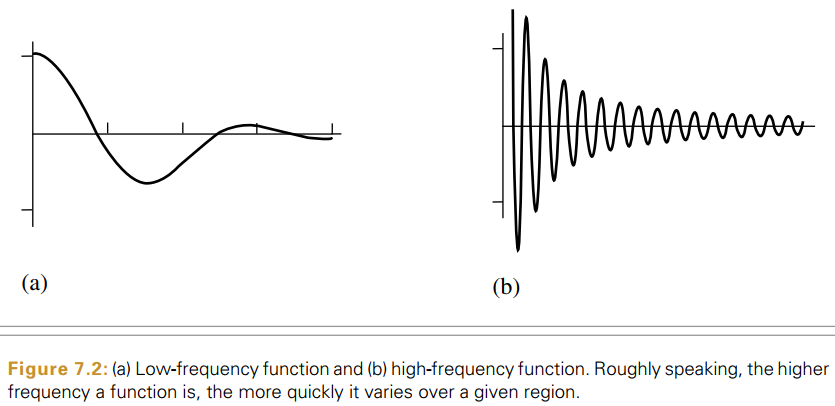
以这些想法为例,考虑一个一维函数(我们将其称为信号),在这里我们可以在函数域中任何所需位置处评估.每个这样的称为**样本位置**,值为**样本值**.图7.1显示了一维函数上一组平滑的样本,以及近似于原始函数的重构信号.在该示例中，是通过线性内插相邻样本值来近似于的分段线性函数(已经熟悉采样理论的读者会将其识别为具有hat函数的重构).因为有关的唯一可用信息来自位置处的样本值,所以不可能完全匹配,因为样本之间没有有关行为的信息.

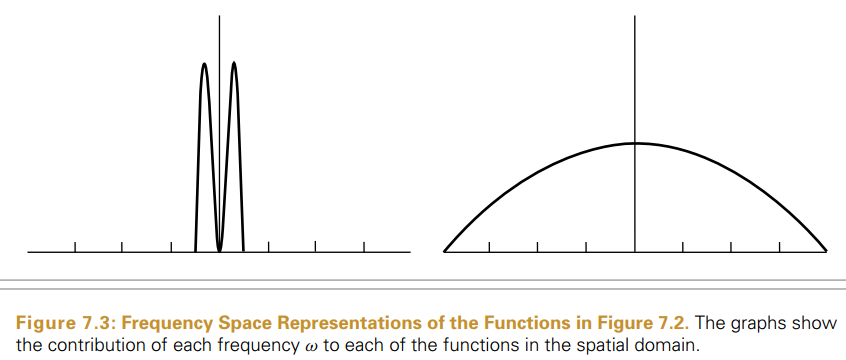
傅立叶分析可用于评估重构函数与原始函数之间的匹配质量.本部分将介绍傅立叶分析的主要思想,并提供足够的细节以完成采样和重构过程的某些部分,但将省略许多属性的证明,并跳过与pbrt中使用的采样算法不直接相关的细节.本章的“更多阅读”部分提供了指向有关这些主题的更详细信息的指针.

7.1.1 频率域和傅立叶变换

傅立叶分析的基础之一是傅立叶变换,它表示频域中的一个函数.(函数通常在时域中表示.)考虑图7.2中绘制的两个函数.图7.2(a)中的函数随的变化相对较慢,而图7.2(b)中的函数变化较快.变化较慢的函数具有较低的频率成分.图7.3显示了这两个函数的频率空间表示.较低频率函数的表示比较高频率函数的表示更快地到达0.



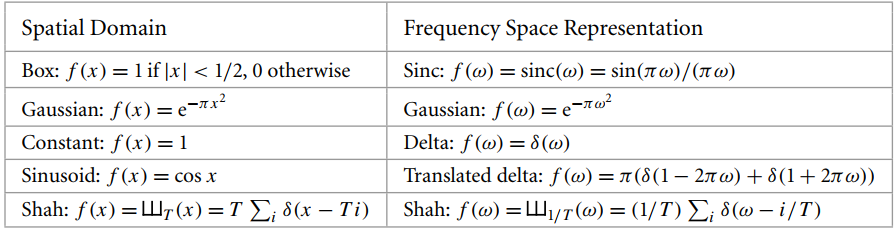




大多数函数可以分解为正弦曲线的加权和.约瑟夫·傅里叶[Joseph Fourier]首先描述了这一非凡的事实,傅里叶变换将函数转换为这种表示形式.函数的这种频率空间表示可以洞察其某些特性-正弦函数中的频率分布对应于原始函数中的频率分布.使用这种形式,可以使用傅立叶分析来深入了解由采样和重建过程引入的误差,以及如何减少该误差的感知影响.

一维函数的傅立叶变换为

(回顾,其中.)为简单起见,这里我们仅考虑的偶函数,在这种情况下的傅立叶变换没有虚数.新函数是频率的函数.我们将用表示傅立叶变换算子,使得.显然是线性算子-也就是说,对于任何标量,且.



**表7.1**:傅里叶函数对.函数在时域和频率空间的表示.由于傅里叶变换的对称性,如果将左列视为频率空间,则右列也将等价于时域空间.

等式被称为傅立叶分析方程,有时也称为傅立叶变换.我们还可以使用傅立叶合成方程或傅立叶逆变换从频域变换回时域:

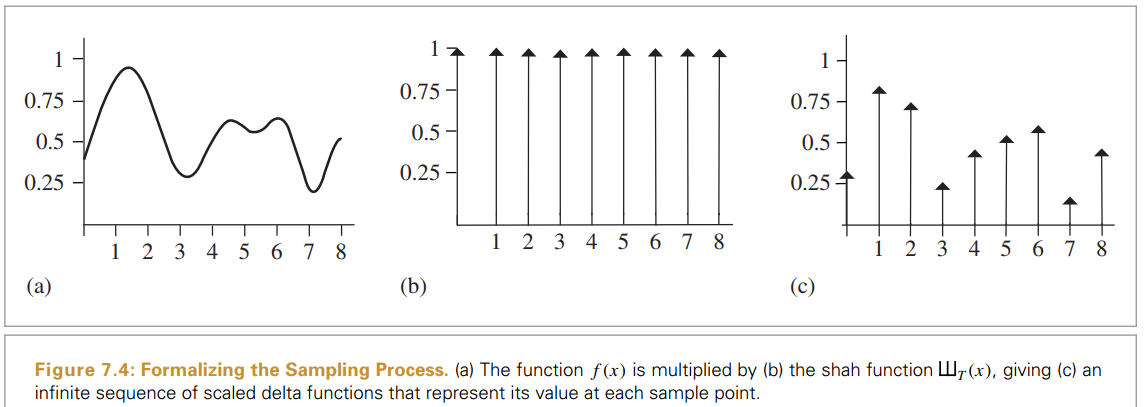
表7.1列出了许多重要函数及其频率空间表示.其中许多函数都基于Dirac delta分布,这是一个特殊函数,其定义为,对于所有.其中一个重要的性质为

delta分布不能表示为标准数学函数,而是通常被认为是以宽度为0的原点为中心的单位面积框函数的极限.

7.1.2 理想的采样和重构

使用频率空间分析,我们现在可以正式研究采样的属性.回想一下,采样过程需要我们选择一组等距的采样位置,并在这些位置计算函数的值.从形式上讲,这相当于将函数乘以“shah”或“脉冲串”函数,即等间隔的delta函数的无限和.shah 定义为(其中等价于表7.1是山字形字母)

其中定义周期或采样率.图7.4说明了抽样的正式定义.乘法在等距的点上产生函数值的无限序列:



这些样本值通过选择重构滤波器函数并计算卷积来重构函数

其中卷积算子定义为

对于重构,卷积给出了以采样点为中心的重构滤波器的缩放实例的加权和:

例如,在图7.1中,使用了三角形重构滤波器.图7.5显示了该示例使用的比例三角形函数.

为了完成一个直观的结果,我们经历了一个看起来似乎非常复杂的过程:重构函数可以通过以某种方式在样本之间进行内插获得.但是,通过仔细设置此背景,傅立叶分析现在可以更轻松地应用于该过程.

通过分析频域中的采样函数,我们可以对采样过程有更深入的了解.特别是,我们将能够确定从样本位置的原始值中准确恢复原始函数的条件,这是非常有力的结果.对于此处的讨论,我们现在假设函数受频带限制——存在某个频率,使得不包含大于的频率.根据定义,带限函数具有紧凑支持的频率空间表示,即对于所有.图7.3中的两个频谱都是频带受限的.

傅立叶分析一个重要思想是,可以证明两个函数的乘积的傅立叶变换是它们各自的傅立叶变换和的卷积:

类似的情况是,时域中的卷积等效于频域中的乘积:

这些特性在傅里叶分析的标准参考文献中得出.利用这些思想,可以在空间域中找到Shah函数和原始函数的乘积的原始采样步骤,可以等效地通过与频率空间中另一个shah函数的卷积来描述.

从表7.1我们也可以知道shah函数的频谱.周期为的shah函数的傅立叶变换是周期为的另一个shah函数.需要牢记的是,周期之间的这种倒数关系很重要:这意味着,如果样本在空间域中相距较远,则它们在频域中相距较近.

因此,采样信号的频域表示由和这个新的shah函数的卷积给出.将一个函数与一个delta函数进行卷积只会生成该函数的拷贝,因此,与一个shah函数进行卷积会生成一个原始函数拷贝的无限序列,其间隔等于shah的周期(图7.6).这是一系列样本的频率空间表示.

现在我们有了函数频谱拷贝的无穷集合,我们如何重建原始函数?从图7.6看,答案是显而易见的:只需丢弃所有频谱拷贝,只保留以原点为中心的频谱,则得到原始.为了舍弃除中心频谱以外的所有频谱,我们将其乘以适当宽度的框函数(图7.7).宽度为的框函数定义为

该乘法步骤对应空间域中与重构滤波器的卷积.这是理想的采样和重建过程.总结一下:

这是一个了不起的结果:仅通过在一组规则间隔的点上对其进行采样,我们就能够确定的确切频率空间表示.除了知道该函数受频带限制之外,没有使用函数组成的其它有关信息.

在时域中应用等效过程将同样准确地恢复.由于盒函数的傅立叶逆变换是sinc函数,因此在空间域中的理想重构为

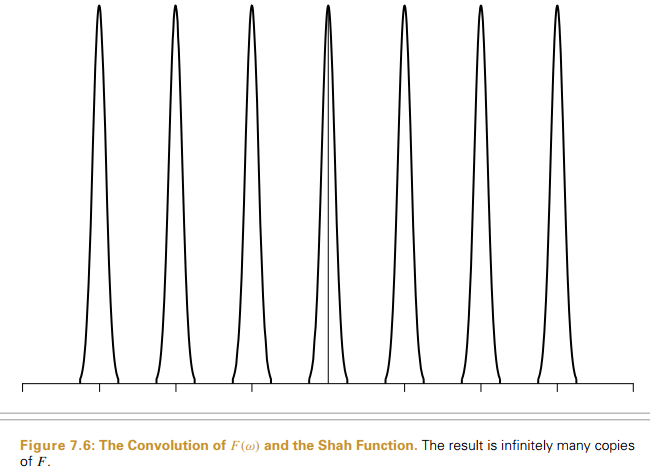
或

不幸的是,由于sinc函数具有无限范围,因此有必要使用所有样本值来计算的任何特定值.具有有限空间范围的过滤器即使无法完美地重构原始功能,也更适合实际应用.

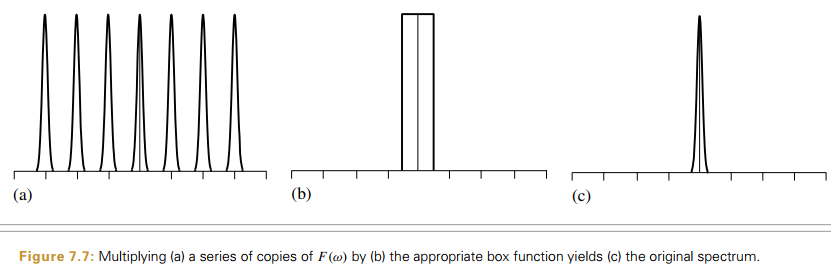
图形中常用的替代方法是使用box函数进行重构,有效地平均周围某个区域内的所有样本值.从频域考虑盒式滤波器的行为可以看出,这是一个非常差的选择:该技术试图通过乘以来隔离函数频谱的中心副本,这不仅在选择频域上做得很不好.该函数频谱的中心副本,但也包括该函数其他副本的无穷系列的高频贡献.

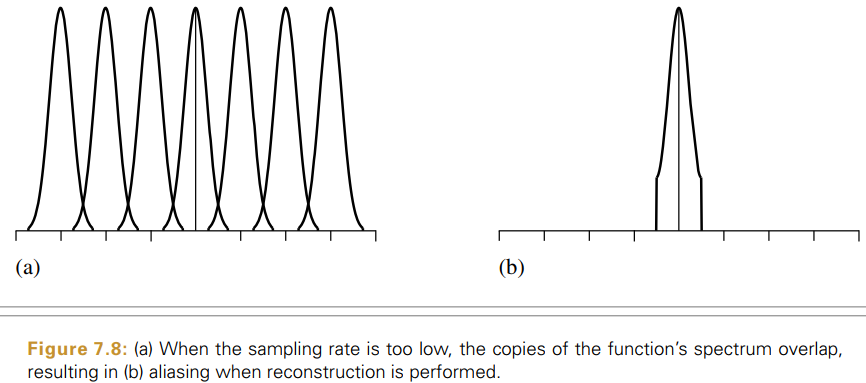
7.1.3 走样 2020年3月4日17点27分

除了sinc函数的无限范围问题之外，理想的采样和重构方法最严重的实际问题之一就是假设信号受频带限制.对于不受频带限制的信号,或者未针对其频率内容以足够高的采样率采样的信号,前面所述的过程将重建与原始信号不同的函数.



成功重建的关键是通过将采样光谱乘以适当宽度的方盒函数来精确恢复原始光谱的能力.请注意,在图7.6中,信号频谱的副本被空白空间隔开,因此可以实现完美的重构.但是,如果以较低的采样率对原始函数进行了采样,请考虑会发生什么.回想一下,周期为的shah函数的傅立叶变换是周期为的新shah函数.这意味着,如果样本之间的间隔在空间域中增加,则样本间隔在频域中会减小,从而将频谱的副本推近.如果副本之间的距离太近,它们将开始重叠.





由于将这些副本加在一起,因此得到的光谱不再像原始副本的许多副本一样(图7.8).当这个新频谱乘以一个框函数时,结果是一个类似于但不等于原始的频谱:原始信号中的高频细节泄漏到了重建信号频谱的低频区域中.这些新的低频伪像称为走样(因为高频“伪装”为低频),并且所产生的信号被称为走样信号.图7.9显示了欠采样后再构造一维函数的效果.

解决频谱重叠问题的可能解决方案是简单地提高采样率,直到频谱副本足够远而不会重叠,从而完全消除混叠.实际上,采样定理告诉我们确切需要多少率.该定理表明,只要统一采样点的频率大于信号中存在的最大频率的两倍,就有可能从采样中完美地重建原始信号.该最小采样频率称为奈奎斯特频率.

对于不受频带限制()的信号,不可能以足够高的采样率执行完美重构.非带限信号具有无限支持的频谱,因此无论其频谱副本有多远(即我们使用的采样率有多高),都将始终有重叠.不幸的是,计算机图形学中很少有函数受频带限制.特别是,任何不连续函数都不会受到频带限制,因此我们无法完美地对其进行采样和重构.这是有道理的,因为函数的不连续性将始终介于两个样本之间,并且这些样本不提供有关不连续性位置的信息,因此,除了增加采样率外,还必须应用其它方法,以抵消混叠会导致渲染器结果产生的误差.

7.1.4 反走样技术

如果不注意采样和重建,则最终的图像中可能会出现大量的伪影.有时,将由于采样和重建造成的伪影区分开是很有用的.当我们希望精确时,我们将采样伪影称为预混叠,将重建伪影称为后混叠.修复这些错误的任何尝试均被广泛归类为抗锯齿.本节将介绍许多抗混叠技术,这些技术不仅可以提高各处的采样率.

非均匀采样

尽管我们将要采样的图像函数具有无限的频率分量,因此无法从点样本中完美重构,但是可以通过非均匀地改变样本之间的间隔来减少混叠的视觉影响.如果表示一个介于0和1之间的随机数,则基于脉冲序列的一组非均匀样本为

如果固定采样率不足以捕获函数,则均匀采样和非均匀采样都会产生错误的重构信号.然而,不均匀的采样趋于将规则的混叠伪像变成噪声,这对人的视觉系统的干扰较小.在频率空间中,采样信号的副本最终也将随机移位,因此在执行重建时,结果是随机误差而不是相干混叠.

自适应采样

已提出的另一种解决混叠的方法是自适应超级采样:如果我们可以识别出频率高于奈奎斯特极限的信号区域,则可以在这些区域中进行其他采样,而无需增加增加采样频率的计算成本.很难使这种方法在实践中很好地工作,因为很难找到所有需要超采样的地方.大多数这样做的技术都是基于检查相邻的样本值并找到两者之间值发生显着变化的位置.假设信号在该区域具有高频.

通常,相邻的样本值不能确定地告诉我们它们之间真正发生的事情:即使值相同,函数之间的差异也可能很大.另一方面,相邻样本可以具有实质上不同的值,而实际上不存在任何混叠.例如,第10章中的纹理过滤算法努力消除由于场景中的图像贴图和过程纹理而造成的混叠.我们不希望自适应采样例程在纹理值快速变化但实际上不存在过高频率的区域中不必要地获取额外的样本.

预过滤

消除混叠的另一种方法是对原始函数进行滤波(即模糊处理),这样就不会保留无法以所使用的采样率准确捕获的高频.在第10章的纹理函数中应用了这种方法.虽然该技术通过从中删除信息来更改被采样函数的特征,但与混叠相比,模糊通常不那么令人讨厌.

回想一下,我们想将原始函数的频谱乘以选择宽度的盒式滤波器,以去除高于奈奎斯特极限的频率.在空间域中,这相当于将原始函数与sinc滤波器进行卷积,

实际上,我们可以使用效果良好的有限范围的过滤器.该滤波器的频率空间表示可以帮助阐明它与理想滤波器的性能近似程度.

图7.10显示了函数与sinc的一个有限范围的卷积进行卷积,这将在7.8节中介绍.请注意,高频细节已被删除;可以以图7.9中使用的采样率对该函数进行采样和重构,而不会出现混叠.

7.1.5 在图像合成中的应用

这些想法在采样和重建渲染场景图像的2D情况下的应用很简单:我们有一幅图像,我们可以将其视为2D图像位置与辐射值L的函数:

好消息是,使用我们的光线跟踪器,我们可以在我们选择的任何点处评估此函数.坏消息是,通常无法在采样前对进行预过滤器以去除高频.因此,本章中的采样器将使用两种策略来提高采样率,使其超出最终图像中的基本像素间隔,以及不均匀地分布采样以将混叠化为噪声.

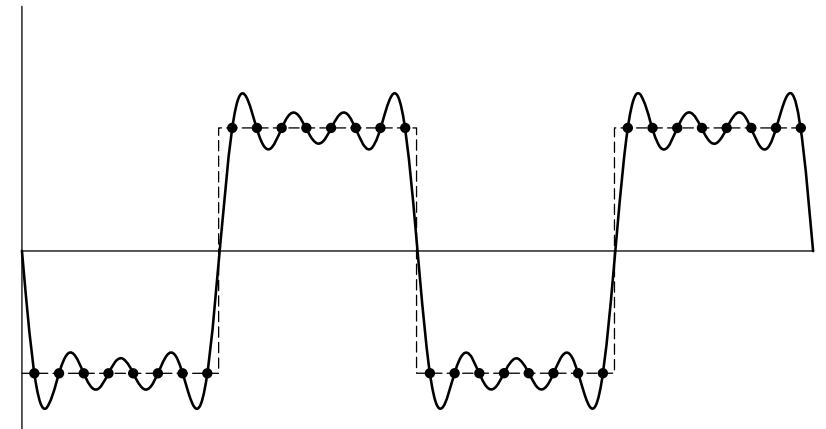
将场景函数的定义概括为高维函数很有用,该函数还取决于时间和对其采样的镜头位置.因为来自相机的光线基于这五个量,所以改变它们中的任何一个都会产生不同的光线,因此的值可能会有所不同.对于特定的图像位置,该点的辐射通常会在整个时间(如果场景中有移动的物体)和镜头上的位置(如果相机具有有限光圈镜头)之间变化.

甚至更一般地说,由于在第14章至第16章中定义的许多积分器都使用统计技术来估计沿给定射线的辐射率,因此当重复给定相同射线时,它们可能返回不同的辐射率值.如果我们进一步扩展场景辐射度函数以包括积分器使用的样本值(例如,用于选择面光源上的点进行照明计算的值),我们将拥有更高维度的图像函数

对所有这些维度进行良好采样是有效生成高质量图像的重要部分.例如,如果我们确保图像附近位置在镜头上倾向于具有不同的位置,则生成的渲染图像将具有较少的误差,因为每个样本都更可能考虑有关 其相邻样本没有的场景.接下来几节中的Sampler类将有效解决对所有这些维度进行采样的问题.

7.1.6 渲染中的混淆源

几何形状是渲染图像中出现锯齿的最常见原因之一.当投影到图像平面上时,对象的边界会引入阶跃函数-图像函数的值会立即从一个值跳到另一个值.如前所述,阶跃函数不仅具有无限的频率含量,且更糟的是,完美的重构滤波器在应用于混叠样本时会导致伪像:振铃伪像出现在重构函数中,这种现象被称为吉布斯现象.图7.11显示了这种效果的一维函数示例.面对混叠,选择有效的重建滤波器需要科学,艺术性和个人品味的结合,这将在本章的后面看到.



**图7.11:吉布斯现象**.如果尚未以奈奎斯特速率对函数进行采样,并且使用sinc滤波器重建了一组别名采样,则重建的函数将具有“振铃”伪影,并在其中围绕真实函数振荡.在这里,一维阶跃函数(虚线)已以0.125的采样间隔进行采样.用sinc重建时,出现振铃(实线).

场景中的小物体也会引起几何锯齿.如果几何体足够小,使其落在图像平面上的样本之间,则它会意外地消失并重新出现在动画的多个帧中.

混叠的另一个来源可能来自物体上的纹理和材质.着色混叠可能是由于纹理贴图未正确过滤(解决此问题是第10章的大部分内容)或光泽表面上的小高光引起的.如果采样率不足以对这些特征进行充分采样,则会导致混叠.此外,物体投射的尖锐阴影会在最终图像中引入另一个阶跃函数.虽然可以从图像平面上的几何边缘识别阶跃函数的位置,但从阴影边界检测阶跃函数更加困难.

关于渲染图像中的锯齿现象的关键见解是,我们永远无法删除其所有来源,因此我们必须开发减轻其对最终图像质量影响的技术.

7.1.7 了解像素

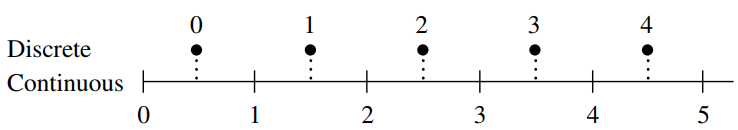
在本章的其余部分中,有两个关于像素的概念需要牢记.首先,至关重要的是要记住,构成图像的像素是图像函数在图像平面上离散点的点样本;没有与像素关联的“面积”.正如Alvy Ray Smith(1995)着重指出的那样,将像素视为具有有限面积的小正方形是一种错误的心理模型,会导致一系列错误.通过使用信号处理方法介绍本章的主题,我们试图为更准确的心理模型奠定基础.

第二个问题是最终图像中的像素自然是在像素网格上的离散整数坐标上定义的,但是本章中的Samplers会在连续的浮点位置生成图像样本.在这两个域之间进行映射的自然方法是将连续坐标四舍五入为最接近的离散坐标.这很吸引人,因为它会将恰好具有与离散坐标相同值的连续坐标映射到该离散坐标.但是,结果是给定一组跨越范围的离散坐标，则覆盖该范围的连续坐标集为.因此,对于给定的离散像素范围,生成连续采样位置的任何代码都会散布1/2个偏移量.容易忘记其中的一些,导致细微的错误.

连续坐标通过下列方式截断为离散坐标

从离散转换为连续

则离散范围的连续坐标范围自然为,并且结果代码要简单得多(Heckbert 1990a).我们将在pbrt中采用的这一约定如图7.12所示.



**图7.12**:图像中的像素可以使用离散或连续坐标进行寻址.五像素宽的离散图像覆盖连续像素范围.特定的离散像素坐标的连续表示为.

7.2 采样接口

正如在第7.1.5节中首次介绍的那样,在pbrt中实现的渲染方法包括选择图像平面上二维点以外的其他维度的采样点.将使用各种算法来生成这些点,但是它们的所有实现都继承自定义接口的抽象Sampler类.核心采样声明和函数位于文件core / sampler.h和core / sampler.cpp中.每个样本生成实现都在samplers /目录中的其自己的源文件中.

采样器的任务是在中生成一系列维样本,为每个图像样本生成一个这样的样本矢量,并且每个样本中维数可能会有所不同,具体取决于光传输算法执行的计算.(见图7.13.)

由于样本值必须严格小于1,因此定义一个常数OneMinusEpsilon非常有用,该常数表示小于1的最大可表示浮点常数.稍后,我们将样本向量值限制为不大于该值.

每当需要采样矢量的其它分量时,采样器的最简单可能的实现方式就是在中返回均匀随机值.这样的采样器将产生正确的图像,但是将需要更多的采样(因此,将跟踪更多的光线并花费更多的时间),以创建更复杂的采样器可以达到的相同质量的图像.使用更好的采样模式的运行时开销与使用均匀随机数等低质量模式的运行时开销大致相同.因为评估每个图像样本的辐射度比计算样本的成分值要昂贵得多,所以这项工作带来了回报(图7.14).

这些样本向量的一些特征假定如下:

1. 采样器生成的前五个参数通常供相机使用.在这种情况下,前两个专门用于在当前像素区域内的图像上选择一个点;第三个用于计算应采样的时间;第四和第五给出景深的镜头位置.
2. 一些采样算法在采样矢量的某些维度上比其他算法生成更好的采样.在系统的其他地方,我们通常假定较早的维度具有放置最充分的样本值.

还应注意,Sampler生成的维样本通常不明确表示或未完整存储，而是通常根据光传输算法的需要以增量方式生成.(但是,存储整个样本矢量并对其组成进行增量更改是16.4.4节中MLTSampler的基础,MLTIntegrator在16.4.5节中使用了它.)

7.2.1 评估样本模式:差异

傅立叶分析为我们提供了一种评估2D采样模式质量的方法,但它仅使我们能够量化能够通过添加可表示的频带受限频率而增加间隔更均匀的采样所带来的改进.考虑到图像边缘中存在无限的频率成分,并且对于蒙特卡洛光传输算法需要()维样本矢量，仅傅里叶分析不足以满足我们的需求.

给定一个渲染器和一个用于放置样本的候选算法,一种评估算法有效性的方法是,与使用大量样本渲染的参考图像相比，使用该采样模式来渲染图像并计算图像中的误差.在本章的后面,我们将使用这种方法来比较采样算法,尽管它只告诉我们该算法在一个特定场景下的性能如何,并且在不经历渲染的情况下也无法使我们了解采样点的质量.

在傅立叶分析之外,数学家提出了一种称为差异的概念,该概念可用于评估n维样本位置模式的质量.分布良好的模式(将在不久后形式化)具有较低的差异值,因此可以将样本模式生成问题视为找到合适的点的低差异模式之一.许多确定性技术具有即使在高维空间中也可以生成低偏差点集.(本章稍后使用的大多数采样算法都使用这些技术.)

差异的基本思想是,可以通过查看域的区域并计算每个内部点的数量来评估n维空间中一组点的质量 区域，并将每个区域的体积与内部的采样点数量进行比较.通常,体积的给定分数应在其中占采样点总数的大致相同分数.尽管不可能总是这样,但我们仍然可以尝试使用一些模式,以最大程度地减少实际体积与按点估算的体积之间的最大差异(差异).图7.15二维显示了该想法的示例.

为了计算一组点的差异,我们首先选择形状族B,它们是的子集.例如,经常使用在原点有一个角的盒子.这对应于

其中.给定一系列采样点,与的差异为(运算符是离散运算符的连续类比.)

其中是中的点数,而是的体积.

公式（7.4）为什么是质量的合理度量的直觉是,值是由特定点组成盒子的体积的近似值.因此,差异最严重通过这种近似体积的方式,所有可能的盒子都出现了误差.当形状B的集合是原点带有角的盒子的集合时,此值称为星号差异.B的另一个流行选项是所有轴对齐的框的集合,其中取消了一个角位于原点的限制.

对于一些特定的点集,可以分析计算出差异.例如,考虑一维中的一组点

我们可以看到的星号差异是

例如,取区间.则,但.此区间(即)是体积与点位于体积内的个数比之间的最大差异.

可以通过稍加修改来改善此序列的星号差异:

则

一维点序列的星号差异的边限已显示为

因此,等式中的较早序列对于一维序列具有最低的可能差异.通常,对于一维序列中的差异而言,分析和计算边界要比在高维中分析和计算边界容易得多.对于不太简单构造的点序列,对于较大尺寸的序列以及比盒子更不规则的形状,通常必须通过构造大量形状,计算其差异并报告找到的最大值来对差异进行数值估算.

精明的读者会注意到,根据低差异度量,这种一维均匀序列是最佳的,但是在本章的前面,我们声称,不规则抖动模式在感知上要优于二维图像采样的均匀模式,因为它们用代替了混叠误差 噪声.在这种框架下,均匀样本显然不是最佳的.幸运的是,高维中的低差异模式比一维中的均匀性要差得多,因此在实践中通常可以很好地用作样本模式.尽管如此,它们的基本一致性意味着低差异模式比具有伪随机变化的模式更容易出现视觉上令人反感的混叠.

单独的差异并不一定是一个好的指标:一些低差异点集会显示出一些样本集,其中两个或多个样本可能非常接近.7.7节中的Sobol采样器尤其受此问题困扰-参见图7.36,该图显示了其前两个维的图.直观地讲,过于紧密的样本并不能很好地利用抽样资源:一个样本与另一个样本越接近,就不可能提供有关被抽样函数的有用的新信息.因此,计算一组点中任意两个样本之间的最小距离也已被证明是样本模式质量的有用度量.最小距离越大,越好.

有多种算法可用于生成泊松磁盘采样模式,这些算法在此指标上得分很高.通过构造,泊松圆盘图案中没有两个点比某个距离更近.研究表明,眼睛中的视锥细胞和视锥细胞以类似的方式分布,这进一步证实了这种分布是成像的好方法的想法.在实践中,我们发现泊松圆盘模式对于2D图像采样非常有效,但是比在更复杂的渲染情况下进行高维采样的更好的低差异模式效果要差.有关更多信息,请参见“更多阅读”部分.

7.2.2 Sampler基类接口

Sampler基类不仅定义了采样器的接口,而且还提供了一些供Sampler实现使用的常用函数.

所有的Sampler实现都必须为构造函数提供将为最终图像中的每个像素生成的样本数量.在极少数情况下,系统将胶片建模为覆盖整个观看区域的单个“像素”可能会很有用.(像素定义的这种重载只是一小部分,但我们允许它简化某些实现方面.)由于此“像素”可能具有数十亿个样本,因此我们使用具有64位精度的变量存储样本数.

当渲染算法准备开始在给定像素上工作时,它将通过调用StartPixel()开始,以提供图像中像素的坐标.一些采样器实现使用对哪个像素进行采样的知识来改善它们为该像素生成的采样的总体分布,而其他实现则忽略此信息.

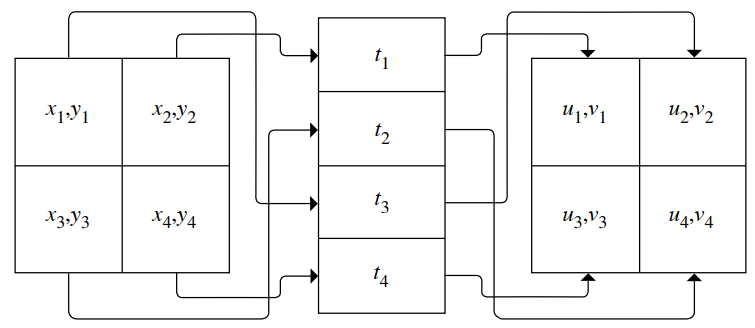
7.3 分层抽样

我们将引入的第一个Sampler实现将像素区域细分为矩形区域,并在每个区域内生成一个样本.这些区域通常称为地层,该采样器称为StratifiedSampler.分层背后的关键思想是,通过将采样域细分为非重叠区域并从每个区域中获取一个样本,我们保证了不会丢失图像的重要特征,因为可以保证样本不会全部靠在一起.从信号处理的角度来看,我们隐式定义了总体采样率,以使层越小,我们拥有的样点就越高,因此采样率就越高.

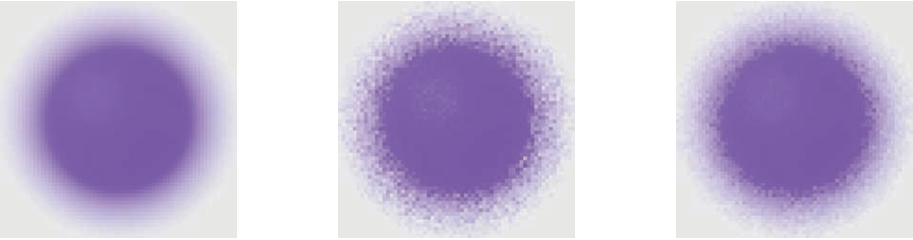
分层采样器通过使层的中心点抖动任意量,直到层的宽度和高度的一半,将每个样本放置在每个层内的随机点上.抖动引起的不均匀性有助于将混叠化为噪声,如7.1节所述.采样器还提供了无抖动模式,可在层中进行均匀采样.此模式对于不同采样技术之间的比较最有用,而不是渲染高质量图像.

将分层直接应用于高维抽样会很快导致大量样本.例如,如果我们将5D图像,镜头和时间样本空间在每个维度上划分为四个层次,则每个像素的样本总数将为45 =1024.我们可以通过在某些维度上减少样本数量来减少这种影响(而不是有效地使用单个层次对某些维度进行分层),但是我们将失去在这些维度中进行分层的样本的好处.分层的问题被称为维数诅咒.

我们可以通过为域维度的子集计算较低维度的分层模式,然后随机将每个维度集合中的样本进行关联,从而获得分层带来的大部分好处,而不必为总采样过多付出代价.(此过程有时称为填充.)图7.16显示了基本思想:我们可能希望每个像素仅获取四个样本,但仍然需要在所有维度上对样本进行分层.我们独立生成四个2D分层图像样本,四个1D分层时间样本和四个2D分层透镜样本.然后,我们将时间和镜头样本值与每个图像样本随机关联.结果是每个像素具有样本,这些样本一起具有对样本空间的良好覆盖.图7.17显示了在渲染景深时,使用分层透镜样本与使用未分层随机样本相比,图像质量得到了改善.



**图7.16**:我们可以生成一个良好的样本模式,从而获得分层的好处,而无需同时对所有样本维度进行分层.在这里,我们将图像位置,时间和镜头位置分成了独立的层次,每个层次有四个区域.分别对每个样本进行采样,然后将时间样本和镜头样本与每个图像样本随机关联.我们在每个维度中都保留了分层的优势,而不必成倍地增加样本总数.



**图7.17**:采样模式在渲染具有景深的紫色球体中的效果.(a)模糊球体的高质量参考图像.(b)通过随机采样在每个像素中生成的图像,没有分层.(c)使用相同数量的样本但使用StratifiedSampler生成的图像,该图像将图像以及镜头图像(更重要的是对图像)进行分层.分层可以大大改善这种情况.

图7.18显示了一些采样模式的比较.第一个是完全随机的模式:我们完全不使用分层就生成了许多样本.结果是可怕的.一些地区的样本很少,而其它地区的样本很多.第二个是均匀的分层图案.最后,均匀的图案被抖动了,每个样本的位置都增加了一个随机偏移,将其保留在其单元格内.与纯随机模式相比,这提供了更好的总体分布,同时保留了分层的好处,尽管仍然有一些样本块和某些区域的样本不足.图7.19显示了使用StratifiedSampler渲染的图像,并显示了抖动的采样位置如何将混叠伪像转变为不那么令人讨厌的噪声.

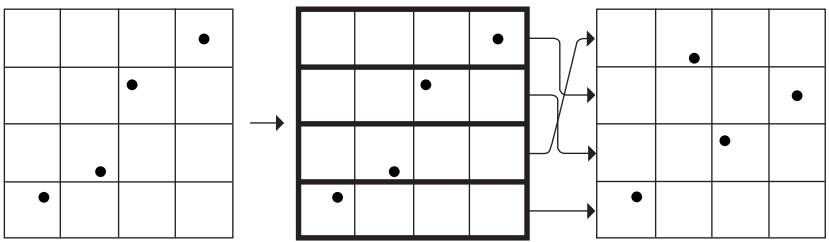
样本数组给我们带来了难题:例如,如果积分器要求像素中每个样本的样本矢量中包含64个2D样本值的数组,则采样器有两个不同的目标可以尝试实现:

1. 阵列中的样本本身最好以2D形式分布(例如,使用8×8分层网格).此处的分层将提高每个样本矢量的计算结果的质量.
2. 最好确保阵列中每个图像样本与图像附近样本的样本值不太相似.相反,我们希望这些点相对于它们的相邻点分布均匀,以便在单个像素周围的区域内,可以很好地覆盖整个样本空间.

分层采样器不是在这里尝试同时解决这两个问题,而是仅解决第一个问题.本章后面的其他采样器将使用更复杂的技术重新审视此问题,并在不同程度上同时解决这两个问题.

第二个复杂性来自以下事实:调用者可能已要求每个图像样本提供任意数量的样本,因此分层可能不容易应用.(例如,如何生成由七个样本组成的2D分层模式?)我们只能生成n×1或1×n分层模式,但这仅给我们带来了一维分层的好处,而不能保证良好的质量.我们将使用一种称为拉丁超立方体采样（LHS）的方法,该方法可以生成任意数量的任意维度的样本的合理分配.

LHS将每个维度的轴均匀地划分为n个区域，并沿对角线在个区域的每个区域中生成一个抖动样本，如图7.20左侧所示.然后，将这些样本在各个维度上随机打乱,以创建具有良好分布的图案.LHS的一个优点是,当将样本投影到采样尺寸的任何一个轴上时，它可以将样本的聚集降到最低.此属性与分层采样相反,在分层采样中，二维图形中的个采样中有个可以投影到每个轴上的基本相同点.图7.21显示了分层采样模式的最坏情况.



**图7.20**:Latin hypercube采样(有时称为n-rooks采样)选择的样本应使网格的每一行和每一列中仅存在一个样本.这可以通过在沿对角线的单元格中生成随机样本,然后随机排列其坐标来完成.LHS的优点之一是它可以生成任何数量的具有良好分布的样本,而不仅仅是分层样本的个样本.

尽管解决了总体问题,但LHS不一定能改善分层抽样;很容易构造出这样的情况,即样本位置基本上是共线的,并且采样域的大面积区域附近没有样本(例如,当原始样本的排列是同一性时,将它们全部留在它们的开始位置).特别是,随着的增加,与分层模式相比,Latin hypercube模式的有效性越来越低.

7.4 Halton Sampler

StratifiedSampler的基本目标是生成分布良好但不均匀的采样点集,没有两个采样点靠得太近,并且所有区域都有样本点.如图7.18所示,尽管相邻层中的样本恰好接近两个层的共享边界时,抖动模式的性能会比随机模式好得多,但其质量会受到影响.

本节介绍HaltonSampler,它基于直接生成低差异点集的算法.与由分层采样器生成的点不同,HaltonSampler不仅生成保证不会太紧密聚集的点,而且还生成在采样矢量的所有维度上同时分布良好的点,而不仅仅是一两个 像StratifiedSampler一样一次标注尺寸.

7.4.1 Hammersley和Halton序列

Halton和Hammersley序列是两个密切相关的低差异点集.两者均基于称为根逆的构造,该构造基于以下事实:正整数可以作为基数以及唯一序列确定

其中所有数字都在0和之间.

基数中的根逆函数通过反映这些关于小数点的数字,将非负整数转换为中的分数值:

因此,数字对跟逆的贡献为.

范德科普特序列序列是最简单的低差异序列之一,它是1D序列,由基数为2的跟逆函数给出:

表7.3列出了范德科普特序列的前几个值.请注意,它是如何将一维线的间隔递归地分成两半,从而在每个间隔的中心生成一个采样点.该序列的差异是

与个维度中无限序列获得的最佳差异相匹配,

为了生成维Halton序列中的点,我们使用根逆基,对于该图案的每个维都有不同的基.所使用的基数必须彼此互为素数,因此自然的选择是使用前个素数:

Halton序列最有用的特征之一是,即使事先不知道所需的样本总数,也可以使用它;序列的所有初始均分布良好,因此将其他样本添加到序列后,将保持较低的差异.(但是,当样本总数是指数的基数的幂的乘积时,其分布是最佳的.)

维Halton序列的差异为

这是渐近最优的.

如果样本数是固定的,则可以使用Hammersley点集,从而使差异略低.Hammersley点集定义为

其中是要采样的总数,并且像以前一样,所有基都是相对质数的.图7.25(a)显示了二维Halton序列的前216个点的曲线图.图7.25(b)显示了Hammersley序列的前256个点.

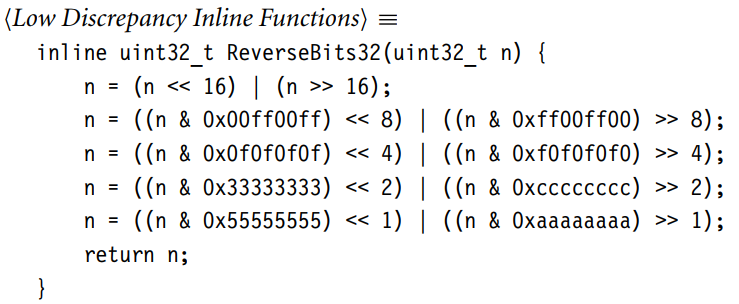
对于以2为底的根逆,我们可以利用以下事实:数字计算机中的数字已经以二进制表示,可以更有效地计算根逆.对于64位值,我们可以从公式(7.6)中获得

首先考虑反转的位的结果,仍然为整数值,这将得出

如果我们将该值除以,

这就是.因此,可以用位反转和2的幂除法等效地计算以2为基的根逆.

可以使用一系列逻辑位运算来有效地反转整数数量的位.ReverseBits32（）函数的第一行将32位整数的位取反,将值的低16位与高16位交换.下一行同时将结果的前8位与第二8位和第三8位与第四位交换.此过程一直持续到最后一行,该行交换相邻的位.要理解此代码,写出各种十六进制常量的二进制值会很有帮助.例如,0xff00ff00是二进制的11111111000000001111111100000000;可以很容易地看到,使用此值的按位或运算会掩盖第一个和第三个8位数量.



然后,可以通过分别反转两个32位组件然后互换来反转64位值的位.

为了计算以2为基的根逆,然后,我们将这些位反转并乘以,其中十六进制浮点常数0x1p-64用于值.如第3.9.1节所述,通过相应的2的幂乘以实现2的幂除法会得到与IEEE浮点数相同的结果.(浮点乘法通常比浮点除法更高效.)

对于其他基数,RadialInverseSpecialized()模板函数通过计算以开头的数字并计算序列来计算根逆,其中且

（例如,以10为基数,它将值1234转换为4321.）可以完全使用整数算术找到该值,而不会累积任何舍入误差.

然后,通过转换为浮点并乘以(其中是该值中的位数)来求出根逆的最终值,以得到等式的值.在处理数字时,invBaseN中会建立此乘法项.

要问的自然问题是，为什么在此处使用在基础上参数化的模板函数(而不是像常规函数调用那样将基础作为参数，这样可以避免为每个基础生成单独的代码路径).这样做的动机是在现代CPU上整数除法的速度惊人地慢,而且可以用编译时常数进行除法的效率更高.

例如,将32位值乘以2863311531以获得64位中间值,然后将结果右移33位,即可精确计算出32位值除以3的整数.这些都是相当有效的操作.(可以使用类似的方法将64位值除以3,但是魔术常数要大得多;有关这些技术的更多信息,请参阅Warren(2006).)因此,在此处使用模板函数可以使编译器看到 计算while循环中next值的除法实际上是一个常数的除法,这使它有机会应用此优化.经过优化的代码在2015年时代的笔记本电脑上的运行速度是基于整数除法指令的实现的5.9倍.

Hammersley和Halton序列的缺点是,随着基数的增加,样本值可能会显示出令人惊讶的规则图案.这个问题可以通过加扰的Halton和Hammersley序列来解决,在计算根逆时,其中对数字应用了置换:

其中是数字的排列.注意,相同的置换用于每个数字，并且相同的置换用于生成给定基数中的所有采样点.图7.26显示了用哈尔顿序列加扰的效果.在下文中,我们将使用随机置换,尽管置换的特定构造可能会产生更好的结果;有关更多详细信息,请参见“进一步阅读”部分.

ComputeRadicalInversePermutations()函数计算这些随机排列表.它为所有排列初始化单个连续数组,其中前两个值是整数0的排列,并且b = 2时是一个，接下来的三个值是0、1、2的排列（对于b = 3以及 依次为主要基础.在下面的for循环的入口处，p指向要为当前素数基数初始化的置换数组的开始.

7.4.2 Halton采样器实现

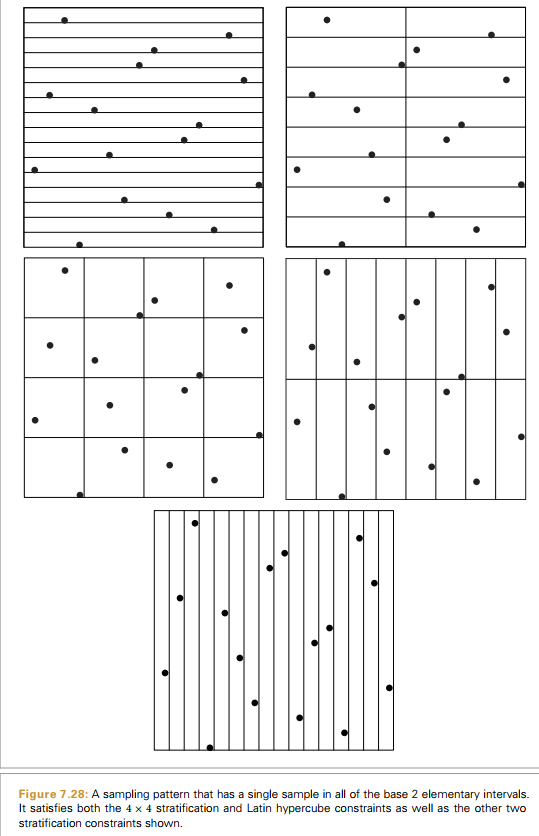
HaltonSampler使用Halton序列生成样本矢量.与StratifiedSampler不同,它是完全确定性的.它在操作中不使用伪随机数.但是,如果对图像的采样不够充分,则可能会导致Halton采样失真.图7.27比较了使用基于Halton的采样器与使用上一节中的分层采样器对棋盘纹理进行采样的结果.请注意沿前景边缘和水平方向的不愉快图案.

为了将样本的前两个维度从映射到像素坐标，HaltonSampler找到最小的比例因子，该因子大于每个维度中图像分辨率或kMaxResolution中较小者.(我们很快就会看到这种特定的比例选择如何使您轻松查看样本落在哪个像素上.)缩放后,图像范围之外的任何样本都将被忽略.

7.5 (0,2)—序列采样器 2020年3月27日09点31分

生成高质量样本的另一种方法是利用某些低差异序列的显着特性,该特性使我们能够满足样本的两个理想特性(StratifiedSampler只满足其中一个要求);它们生成像素值的样本矢量 使得每个像素样本的样本值就可以很好地相对于彼此分布,并且同时使得像素中所有像素样本的样本值的集合可以集中地很好地分布.

该序列使用由Sobol导出的低差异序列的前两个维度.该序列是一种特殊类型的低差异序列,称为(0,2)—序列,(0,2)—序列以非常通用的方式分层.例如,(0,2)序列中的前16个样本满足第7.3节中分层抽样的分层约束,这意味着每个扩展区中只有一个样本.但是,它们也满足拉丁超立方体约束,因为在范围和的每个框中只有一个.此外,在范围和）的每个框中只有一个样本.图7.28显示了将域划分为(0,2)—序列的前16个样本满足分层特性的区域的所有可能性.来自此模式的16个样本的每个后续序列也满足这些分布特性.



通常,来自(0,2)—序列的任何长度为的序列(其中是一个非负整数)都满足此一般分层约束.二维(基数2)基本区间集定义为

其中整数.序列中前个值中的每一个样本将位于每个基本间隔中.此外,对于每个后续的值集合,该属性均成立.

现在要了解如何将(0,2)—序列应用于生成2D样本,考虑一个像素,该像素具有2×2个图像样本,每个像素具有4×4个2D样本阵列.根据相应的基本间隔集,(0,2)—序列的前(2×2)×(4×4)= 26个值相对于彼此分布良好.此外,前4个4×4 = 24个样本本身按照其对应的基本间隔分布良好,接下来的24个样本以及随后的样本依此类推.因此,对于一个像素的第一个图像样本,我们可以使用前16个(0,2)—序列样本作为4×4阵列的样本,然后对下一个图像样本使用接下来的16个序列,依此类推.结果是一组非常均匀分布的采样点.

7.5.1使用生成器矩阵进行采样

Sobol序列基于与HaltonSampler不同的机制来生成采样点,而HaltonSampler使用了各个维度的基本逆.即使将基本逆函数中的整数除法转换为乘和移位,高质量,高分辨率渲染所需的数十亿个样本的计算量仍然可能很大.大多数计算费用来自在本机以2为基运行的计算机上执行非base-2计算的开销.(考虑Compute base-2部首逆片段与RadicalInverseSpecialized（）模板函数之间的对比）.

鉴于非base-2运算的高昂成本,自然而然地尝试开发完全在base 2上运行的样本生成算法.一种有效的方法是使用生成器矩阵，该矩阵允许所有计算都在base 2中进行.相同的基,与其像霍尔顿采样器那样在每个维度中使用不同的基数，在每个维度中使用不同的生成器矩阵。 通过为每个采样维选择正确的矩阵，可以生成非常好的点的低差异分布。 例如，可以使用以2为底的两个特定生成器矩阵定义(0,2)—序列.

要查看生成器矩阵的使用方式,请考虑以为底的位数字,其中的第位数字为,我们有个生成器矩阵.然后,相应的采样点为

其中所有算术都在环中执行(换句话说,当所有运算都以为模执行时).这种构造给出的点总数在0到的范围内.如果生成器矩阵是单位矩阵,则此定义对应于规则根基逆矩阵b.(停顿一下,确保在继续操作之前已看到公式(7.7)和(7.9)之间的这种联系.)

在本节中,我们将仅使用和.虽然在样本生成算法中引入32×32矩阵似乎并不意味着朝着更好的性能迈出了一步,但我们将最终看到样本代码可以映射到使用少量位操作以极其有效的方式执行此计算的实现.

迈向高性能的第一步来自我们以2为基进行工作的事实;这样,C的所有条目都是0或1,因此我们可以用一个无符号的32位整数表示矩阵的每一行或每一列.我们将选择将矩阵的列表示为uint32t\_ts;这种选择导致一个非常有效的算法,用于将列向量乘以.

现在考虑计算矩阵矢量积的任务;使用矩阵向量乘法的定义,我们有:

换句话说,对于的每个数字,其值为1,应将C的对应列相加.依次可以在中非常有效地执行此加法:在该设置中,加法对应于异或运算.(考虑两个可能的操作数值(0和1)的组合,以及将它们加到mod 2的结果,并与具有相同操作数值的异或运算所得的值进行比较.)因此,乘法仅是对位为1的C列进行异或运算的问题.此计算是在MultiplyGenerator（）函数中实现的.(**原书458页有源码**)

现在回到方程(7.9),如果我们表示乘积的列向量,则现在考虑向量乘积

由于的元素存储在单个uint32\_t中,因此将它们的值解释为uint32\_t是

如果我们要反转uint32\_t中的位顺序,那么我们将得到

这是一个更有用的值:如果将该值除以,我们将得到公式（7.11）,即——我们要计算的值.

**因此,如果我们采用MultiplyGenerator（）函数的结果,则反转返回值位的顺序(例如,通过使用ReverseBits32（））,然后将该整数值除以以计算出其中的浮点值[0，1）,我们就得到样本值**.

为了节省反转位的小开销,我们可以等效地反转生成器矩阵C的所有列的位,然后再将其传递给MultiplyGenerator（）.我们将在下面使用该约定.

为了使(0,2)—序列在实践中有用,我们还需要能够为每个图像样本生成多个不同的2D样本值集,并且我们希望为每个像素生成不同的样本值.解决此问题的一种方法是为每个像素使用精心选择的(0,2)—序列的非重叠子序列.另一种方法是随机扰乱(0,2)—序列,得到一个新的(0,2)—序列——通过对原始序列中的值的基数位应用随机置换而建立.

我们将使用的加扰方法归功于Kollig和Keller(2002).它反复分割并重新排列单位正方形.在这两个维度的每一个中,它首先将正方形分成两半,然后以50％的概率交换两半.然后，它将间隔和中的每一个均分为两半,并随机交换这两个半部中的每一个.此过程以递归方式继续进行,所有base-2表示形式的位均已处理.这个过程经过精心设计,以保留点集的低偏差特性.否则,加扰将失去(0,2)—序列的优势.图7.29显示了一个未加扰的(0,2)—序列及其两个随机加扰的变体.

有两件事使加扰过程有效:首先,因为我们要对两个以2为底的序列进行加扰,所以序列的数字都为0或1,加扰一个特定的数字等效于将其与0或 1.第二,进行简化,使得在递归加扰的每个级别处,将做出关于是否交换对子间隔中的每对子间隔的相同决定.这两种设计选择的结果是,可以将加扰编码为存储在uint32\_t中的一组位,并可以通过异或运算将其应用于原始数字.

SampleGeneratorMatrix（）函数将这些片段组合在一起以生成样本值.

使用格雷码时,连续的二进制值仅在单个位中有所不同.表7.4的第三列按格雷码顺序显示了前八个整数.请注意,不仅单个位在任何一对值之间都发生了变化,而且在从0开始的任何2的幂的n个数的值中,格雷码都列举了从0到的所有值.只是顺序不同.

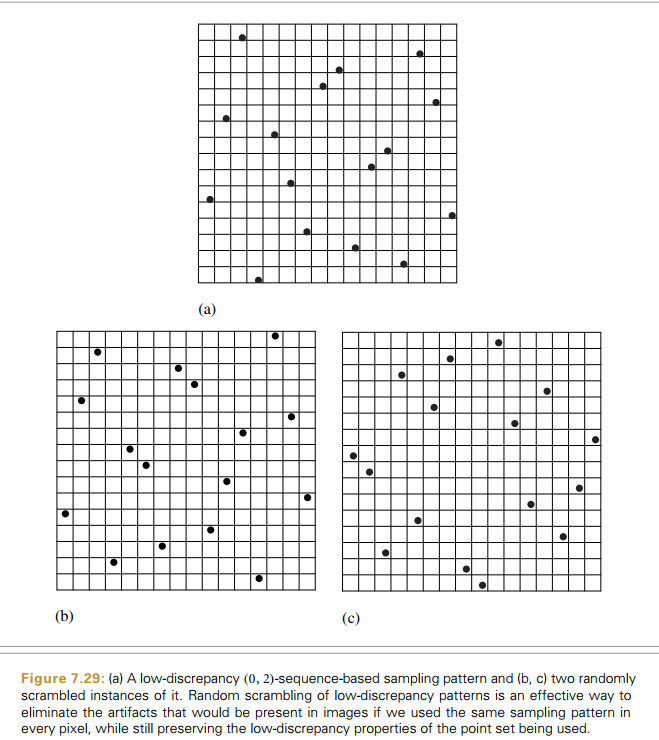
7.5.2 Sampler实现

ZeroTwoSequenceSampler使用加扰的(0,2)—序列生成胶片平面,镜头和其它2D样本上的位置的样本，并生成带有加扰的van der Corput序列的1D样本.图7.30比较了使用（0，2）-序列对景深进行镜头采样与使用分层图案的结果.

在使用加扰的(0,2)—序列时,必须考虑到一个细微的实现细节。通常，积分器会使用采样器在计算特定积分的值的过程中创建的多个采样模式中的一种以上的采样 。 例如，他们可能使用来自1D模式的样本来选择场景中的N个光源之一来从中采样照明，然后可能使用来自2D模式的样本来选择该光源上的采样点（如果是） 区域照明。

即使使用针对每个随机扰码具有不同随机扰码值的随机扰码来计算这两种模式，这些模式的元素之间仍可以保留一些相关性，从而使1D模式的第i个元素和2D模式的第i个元素相关。 这样，在较早的区域照明示例中，由于这种相关性，每个光源上的采样点分布通常不会覆盖整个光线，从而导致异常的渲染错误.

通过在生成各种尺寸后分别随机改组各个尺寸,可以轻松解决此问题.生成加扰的1D低差异采样模式后，在此像素的所有图像样本中提供了分布均匀的样本集，此功能以两种方式对这些样本进行混洗.例如，考虑具有8个图像样本的像素，每个像素都有用于积分器的4个1D样本（总共有32个积分器样本）.首先,它对4个样本的8组中的每个样本进行混洗,将每4个样本的集合随机排列.接下来,相对于其他4个样本块,将4个样本的8个组中的每一个随机地作为块.



Sobol2D（）函数采用与VanDerCorput（）类似的结构,但使用两个生成器矩阵生成Sobol点的前两个维度.

7.6 最大最小距离采样器

**(0,2)—序列采样器比分层采样器更有效，这是由于在所有基本间隔上都已分层.但是,有时仍会生成彼此靠近的采样点**.一种替代方法是使用另一对生成器矩阵,它们不仅生成(0,2)—序列,而且还经过专门设计以最大程度地增加样本之间的距离.该方法由MaxMinDistSampler实现. （有关这些生成器矩阵的起源的更多详细信息，请参见“进一步阅读”部分。）

7.7 SOBOL采样器

本章的最后一个采样器基于Sobol生成的一系列生成器矩阵.这些矩阵生成的序列中的样本的区别在于,它们非常高效地实现(由于完全基于base-2计算),而且在样本向量的所有n个维度上分布都非常好.图7.34显示了前几个Sobol发生器矩阵,图7.35将其与具有景深场景的分层点和Halton点进行了比较.

Sobol点的弱点在于它们在收敛之前很容易出现结构化网格伪像.在图7.36所示的图像采样点和图7.37的图像中可以看到这种问题.作为这种弱点的交换,Sobol序列在样本向量的所有个维度上分布得非常好.

7.8 图像重构 2020年4月1日11点20分

给定精心选择的图像样本,我们需要将样本及其计算的辐射值转换为像素值以进行显示或存储.根据信号处理理论,我们需要做三件事来为输出图像中的每个像素计算最终值:

1. 从图像样本集重建连续图像函数.
2. 对函数进行预滤波以去除像素间隔超过奈奎斯特极限的任何频率.
3. 在像素位置处采样以计算最终像素值.

因为我们知道我们将仅在像素位置对函数进行重采样，所以不必构造函数的显式表示形式.相反,我们可以使用单个过滤器函数来组合前两个步骤.

回想一下,如果原始函数已经对大于奈奎斯特频率的频率进行了均匀采样,并使用sinc滤波器进行了重构,那么第一步中的重构函数将与原始图像函数完全匹配,这是一项壮举,因为我们只有点样本.但是,由于图像功能几乎总是具有比采样率所能说明的更高的频率(由于边缘等),因此我们选择对图像进行非均匀采样,从而权衡噪声和混叠.

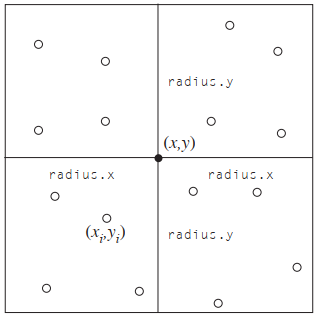
理想重建背后的理论取决于样本的均匀间隔.尽管已经进行了许多尝试以将理论扩展到非均匀采样,但是对于该问题还没有公认的方法.此外,由于已知采样率不足捕获函数,因此无法进行完美的重构.采样理论领域的最新研究重新认识到重构问题,明确承认在实践中通常无法获得完美的重构.视角的这种微小变化导致了强大的新重建技术.有关这些进展的调查,请参见例如Unser（2000）.特别是,重建理论的研究目标已从完美重建转向发展重建技术,这种技术可证明能够最大程度地减少重建函数与原始函数之间的误差,而无论原始函数是否受频带限制.

虽然pbrt中使用的重建技术并非直接建立在这些新方法上,但它们可以用来说明从业人员的经验,即将完美的重建技术应用于为图像合成而采集的样本通常不会产生最高质量的图像.

为了重建像素值,我们将考虑对特定像素附近的样本进行插值的问题.为了计算像素的最终值,内插会导致计算加权平均值

其中

1. 是位于的第个样本的辐射值
2. 是摄像机返回的样本贡献权重.如第6.4.7和13.6.6节所述,这些权重的计算方式决定了胶片测量的辐射量.
3. 是过滤函数.

 图7.38显示了在位置处的像素,该像素具有像素滤镜,其滤镜在方向的范围为radius.x,在方向的radius.y.滤镜范围给出的方框内的所有样本都可能对像素值有所贡献,具体取决于滤镜函数的值.

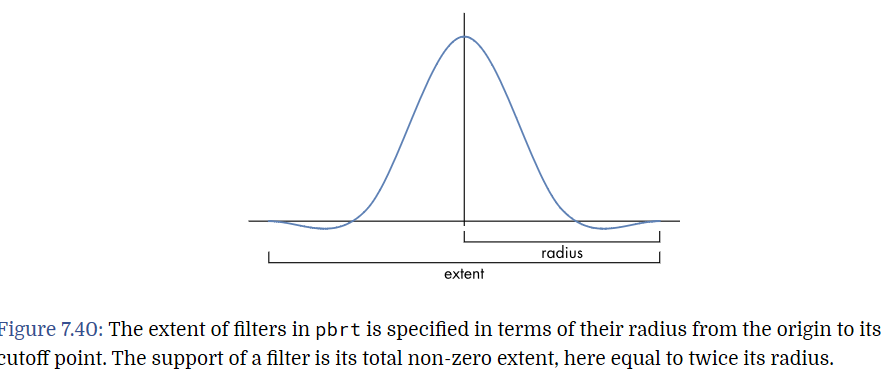
**图7.38**:为计算位于处带有实心圆的像素的滤波后的像素值,需要考虑矩形范围[radius.x,radius.y]内的所有图像样本.用空心圆表示的每个图像样本由2D滤波函数加权.所有样本的加权平均值是最终像素值.

sinc滤波器在这里不是合适的选择:回想一下,当基础函数的频率超过奈奎斯特极限(吉布斯现象)时,理想的sinc滤波器很容易发生振铃,这意味着图像中的边缘在附近像素中具有模糊的边缘复制副本.此外,sinc滤镜具有无限支持:在距其中心一定距离处不会下降到零,因此需要为每个输出像素过滤所有图像样本.实际上,没有单一的最佳过滤器函数.为特定的场景选择最佳的场景需要结合定量评估和定性判断.

7.8.1 滤波函数

pbrt中的所有过滤器实现都派生自抽象的Filter类,该类为过滤中使用的函数提供接口;参见公式(7.12).Film类（在第7.9节中描述）存储一个指向Filter的指针,并在将其累加到最终图像中时使用它来过滤图像样本贡献.(图7.39显示了使用本节中的各种过滤器以重建像素值来渲染图像的放大区域的比较.）Filter基类在文件core/filter.h和core/filter.cpp中定义.

所有过滤器均以原点为中心,并定义一个半径,超出该半径时其值为0.该宽度在x和y方向上可能不同.构造函数获取半径值并将其与倒数一起存储,以供过滤器实现使用.滤镜在每个方向(其支撑)的总体范围是其相应半径值的两倍(图7.40).



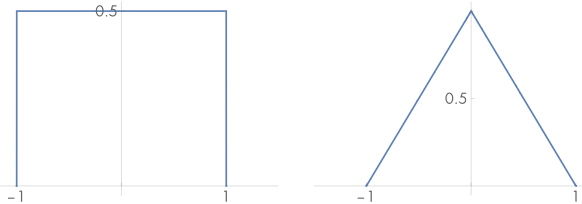
Filter实现需要提供的唯一方法是Evaluate().它以2D点作为参数,该点给出了采样点相对于滤镜中心的位置.此时将返回过滤器的值.系统其他地方的代码永远不会调用过滤器范围之外的点的过滤器函数,因此过滤器实现无需检查这种情况.

Box Filter

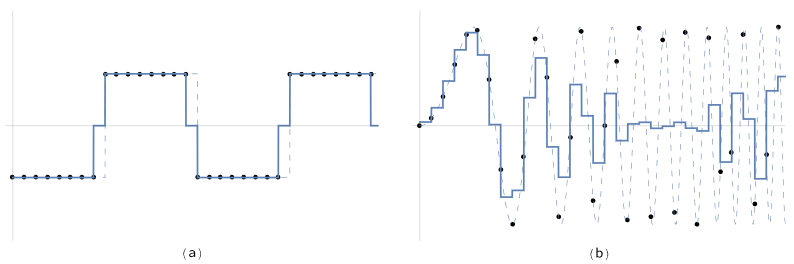
图形中最常用的过滤器之一是Box过滤器（实际上,当未明确解决过滤和重构问题时,Box过滤器实际上就是结果）.Box过滤器对图像正方形区域内的所有样本平均加权.尽管计算效率很高,但它可能是最差的过滤器.回想第7.1节的讨论，Box过滤器允许高频样本数据泄漏到重构值中.这将导致后混叠-即使原始样本值的频率足够高以避免混叠,滤波不良也会引入误差.

图7.41(a)展示了Box滤波器的图像,图7.42显示了使用Box滤波器重构两个一维函数的结果.对于我们先前用来说明Gibbs现象的阶跃函数,Box做得很好.但是,对于沿轴频率增加的正弦函数,结果要差得多.Box滤波器不仅在频率较低时重建函数的效果不佳,即使原始函数很平滑也无法获得连续的结果,而且在函数频率接近并通过奈奎斯特极限时,重建效果也很差.

由于不会使用超出过滤器范围的值调用评估函数,因此它始终可以为过滤器函数的值返回1.



**图7.41**:(a)框过滤器和(a)三角形过滤器的图像.尽管这些都不是特别好的过滤器,但它们在计算效率,易于实现以及评估其他过滤器的良好基准方面都非常出色.



**图7.42**:盒形滤波器重建(a)一个阶跃函数和(b)一个正弦函数,其频率随着的增加而增加.正如预期的那样,该滤波器在阶跃函数中表现良好,但在正弦函数中却表现极差.

三角滤波器

三角形过滤器的结果比方框略好:在过滤器中心的样本的权重为1,并且权重线性下降到过滤器的平方范围.三角形滤波器的图表见图7.41(b).

评估三角形滤波器很简单:该实现仅基于滤波器在和方向上的宽度来计算线性函数.

高斯滤波器

与Box和三角形滤镜不同,高斯滤镜在实践中给出了相当不错的结果.该滤镜应用以像素为中心并围绕其径向对称的高斯凸点.从过滤器值中减去其范围末尾的高斯值,以使过滤器在其极限处变为0（图7.43）.与其他一些滤镜相比,高斯确实会导致最终图像的轻微模糊,但是这种模糊实际上可以帮助掩盖图像中任何剩余的混叠.

半径r的一维高斯滤波函数为

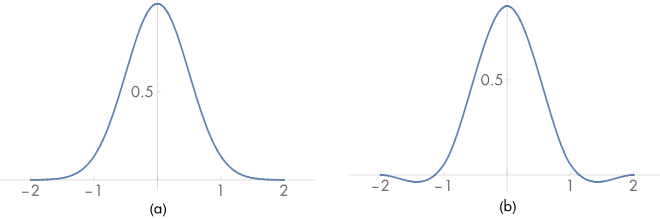
其中,控制滤波器的衰减率.较小的值将导致较慢的衰减,从而产生模糊的图像.这里的第二项确保高斯在其范围结束时变为0,而不是突然出现悬崖.为了提高效率,构造函数在每个方向上预先计算的常数项.

由于2D高斯函数可分为两个1D高斯函数的乘积,因此该实现两次调用Gaussian()函数并将结果相乘.

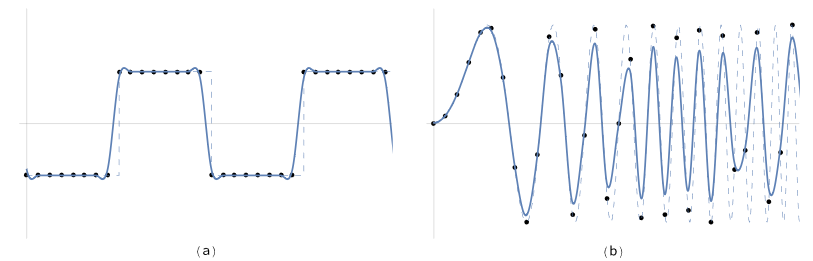
Mitchell Filter

过滤器的设计非常困难,要混合数学分析和感知实验.Mitchell和Netravali(1988)开发了一系列参数化滤波器函数,以便能够系统地探索该空间.在分析测试对象对使用各种参数值过滤后的图像的主观反应后,他们开发了一种过滤器,该过滤器往往在振铃(图像中的虚拟边缘与实际边缘相邻)与模糊(结果过于模糊)之间做出权衡——来自劣质重建滤波器的两个常见伪像.

请注意,从图7.43(b)的图表中可以看出,此过滤器函数的边沿为负值.它具有负裂片.实际上,这些负区域可改善边缘的清晰度,从而使图像更清晰(减少模糊).但是,如果它们变得太大,则铃声会开始进入图像.同样,由于最终像素值可能因此变为负值,因此最终将需要将它们限制在合法的输出范围内.



**图7.43**:(a)高斯滤波器和（b）Mitchell滤波器的曲线图,其中和,每个宽度为2.高斯给出的图像趋于有点模糊,而负瓣 Mitchell滤镜的效果有助于突出和锐化最终图像中的边缘.



**图7.44**:用于重建示例函数的Mitchell-Netravali滤波器.它在这两个函数上都做得很好,(a)通过阶跃函数引入了最小的振铃,并且(b)精确地表示正弦曲线,直到欠采样的混叠开始占主导地位为止.

图7.44显示了该滤波器重构的两个测试函数.两者的结合都非常出色:越阶函数的振铃最少,正弦函数的效果很好,直到采样率不足以捕获功能细节为止.

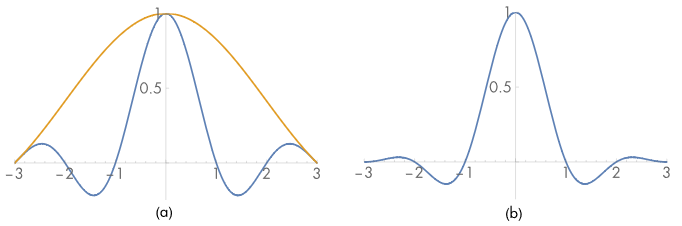
Mitchell滤波器具有两个参数:和.尽管这些参数可以使用任何值,但Mitchell和Netravali建议它们沿着线.

Mitchell-Netravali滤波器是和方向上一维滤波器函数的乘积,因此像高斯滤波器一样是可分离的.(实际上,pbrt中提供的所有过滤器都是可分离的.)但是,Filter::Evaluate()接口并不强制执行此要求,从而为将来实现新过滤器提供了更大的灵活性.

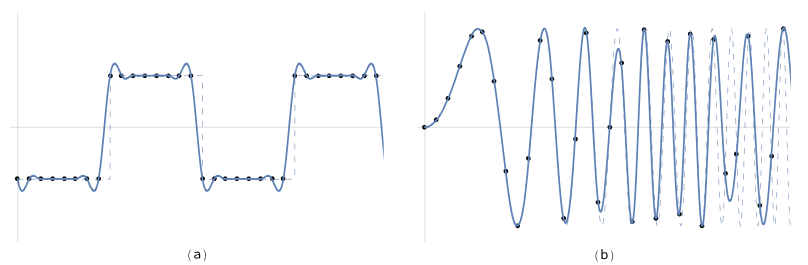
Mitchell滤波器中使用的一维函数是定义在范围内的偶函数.通过将在上定义的三次多项式与在上定义的另一个三次多项式连接在一起来实现此函数.该组合多项式也反映在平面周围,以给出完整的函数.这些多项式由和参数控制,并经过精心选择,以确保和时C0和C1的连续性.

Windowed Sinc Filter

最后,LanczosSincFilter类基于sinc函数实现了一个过滤器.实际上,sinc滤波器通常乘以另一个函数,该函数在一定距离后变为0.这提供了有限范围的过滤器功能,这对于具有合理性能的实现是必不可少的.另外一个参数控制sinc函数在被固定为0之前要经过多少个周期.图7.45显示了一个sinc函数的三个周期的图形,以及我们使用的Windowed函数的图形.由Lanczos Lanczos窗口只是sinc函数的中心瓣,按比例缩放以覆盖周期:



**图7.45**:sinc滤波器的图形.(a)sinc函数,在三个周期(蓝线)和Lanczos窗口函数(橙色线)后被截断.(b)这两个函数的乘积,在LanczosSincFilter中实现.



**图7.46**:使用窗口sinc滤波器重构示例函数的结果.在这里.(a)像无穷大的sinc一样,它的阶跃函数会产生振铃,尽管窗口版本的振铃要少得多.(b)然而过滤器对正弦波的效果很好.

图7.45还显示了我们将在此处实现的过滤器,它是sinc函数和windowing函数的乘积.

图7.46显示了均匀一维样本的窗口sinc重建结果.多亏了windowed,与使用无限范围sinc函数进行的重建相比,重建的阶跃函数呈现出的振铃少得多(请参见图7.11).Windowed sinc滤波器在重构正弦函数之前也非常出色,直到进行预混叠.

7.9 胶片和图像管线 2020年4月8日09点36分

照相机中的胶卷或传感器的类型对入射光转换为图像中的颜色的方式具有戏剧性的影响.在pbrt中,Film类在模拟相机中对传感设备进行建模.找到每条相机射线的辐射后,“胶片”实现将确定样品对胶片平面上相机射线开始的点周围像素的贡献,并更新其图像表示.当主渲染循环退出时,胶片将最终图像写入文件.

对于真实的相机模型,第6.4.7节介绍了测量方程,该方程描述了相机中的传感器如何测量在一段时间内到达传感器区域的能量.对于更简单的相机型号,我们可以考虑传感器在一段时间内测量小区域的平均辐射.选择进行哪种测量的效果封装在Camera :: GenerateRayDifferential()返回的光线的权重中.因此,只要通过这些权重缩放提供的辐射值,就可以执行Film实现而不必考虑这些变化.

本节介绍单个Film实现,该实现应用像素重建方程式来计算最终像素值.对于基于物理的渲染器,通常最好将生成的图像以浮点图像格式存储.与使用具有8位无符号整数值的传统图像格式相比,这样做在输出的使用方式上提供了更大的灵活性.浮点格式可避免由于将图像量化为8位而造成的大量信息丢失.

为了在现代显示设备上显示此类图像,必须将这些浮点像素值映射为离散值以进行显示.例如,计算机监视器通常期望每个像素的颜色由RGB颜色三元组描述,而不是任意光谱功率分布.因此,由通用基函数系数描述的光谱必须先转换为RGB表示形式,然后才能显示.一个相关的问题是,显示器的可显示辐射值范围远小于许多现实场景中存在的范围.因此,像素值必须以一种方式映射到可显示范围,以使最终显示的图像看起来尽可能地接近理想显示设备上显示的方式,而不受此限制.这些问题通过对色调映射的研究得以解决.“更多阅读”部分提供了有关此主题的更多信息.

7.9.1 Film类

Film在文件core / film.h和core / film.cpp中定义.

许多值传递给构造函数:图像的整体分辨率(resolution,以像素为单位);裁剪窗口(crop window),可以指定要渲染的图像子集;胶片物理区域的对角线长度,以毫米为单位指定给构造函数,但在此处转换为米;过滤函数(Fiulter);输出图像的文件名和控制如何在文件中存储图像像素值的参数.

结合整体图像分辨率,裁剪窗口给出了需要实际存储和写出的像素范围.裁剪窗口对于调试或将大图像分解成小块非常有用,这些小块可以在不同的计算机上呈现并在以后重新组合.裁剪窗口在NDC空间中指定,每个坐标的范围为0到1（图7.47）.Film :: croppedPixelBounds存储从裁剪窗口的左上角到右下角的像素范围.分数像素坐标被四舍五入:这样可以确保如果将图像与相邻的裁剪窗口一起渲染,则每个最终像素将仅出现在一个子图像中.

给定(可能已裁剪)图像像素分辨率,构造函数将为每一个像素分配一个Pixel结构数组.光谱像素贡献的运行加权总和用XYZ颜色表示(第5.2.1节),并存储在xyz成员变量中.filterWeightSum保留针对像素的样本贡献的过滤器权重值的总和.splatXYZ持有样本splat的(未加权)总和.字段pad未使用;其唯一目的是确保Pixel结构的大小为32个字节,而不是28字节(假定为4字节浮点数;否则,它确保64字节结构).这种填充确保了Pixel不会跨越缓存行,从而在访问Pixel时不会发生一次以上的缓存未命中(只要在缓存行的开头分配了数组中的第一个Pixel).

使用XYZ颜色存储像素值的两种自然选择是使用“光谱”值或存储RGB颜色.在这里,即使进行完整的光谱渲染,也不值得存储完整的光谱值.由于写入输出文件的最终颜色不包括光谱样本的完整集合,因此与存储光谱并转换为图像输出的三刺激值相比,此处转换为三刺激值并不表示信息丢失.如果Spectrum有大量样本,在这种情况下不存储完整的Spectrum值可以节省大量内存.(如果pbrt支持将SampledSpectrum值保存到文件中,则需要重新考虑此设计选择.)

我们选择使用XYZ颜色而不是RGB来强调XYZ是与显示无关的颜色表示,而RGB需要假设一组特定的显示响应曲线(见5.2.2节).(但是最后,由于几乎没有图像文件格式存储XYZ颜色,因此我们将不得不转换为RGB.)

使用典型的过滤器设置,每个图像样本可能在最终图像中占16个或更多像素.特别是对于简单的场景,在这些场景上射线相交测试和着色计算花费的时间相对较少,因此为每个样本更新图像所花费的时间可能很长.因此,Film预先计算了一个过滤器值表,这样我们就可以避免对Filter::Evaluate()方法调用虚拟函数的开销以及对过滤器进行求值的开销,而可以使用表中的值进行过滤.实际上,在每个样本的精确位置不评估过滤器而导致的错误并不明显.

此处的实现做出合理的假设,即滤波器定义为,因此该表仅需要保存滤波器偏移量的正象限值.这个假设对pbrt中当前可用的所有过滤器都是正确的,并且对于实践中使用的大多数过滤器都是正确的.这使表的大小增加了四分之一,并提高了内存访问的一致性,从而提高了缓存性能.

Film实现负责确定采样器为其生成样本的整数像素值的范围.GetSampleBounds()方法返回要采样的区域.由于像素重建滤波器通常跨越多个像素,因此采样器必须生成实际将要输出的像素范围之外的图像采样.这样一来,即使图像边界处的像素在各个方向上的像素密度也相同,并且不会仅受到来自图像内部的值的影响.当使用裁剪窗口渲染图像时,此细节也很重要,因为它消除了子图像边缘的伪影.

计算样本边界时,需要考虑从离散像素坐标转换为连续像素坐标时的半像素偏移,将其扩展为过滤器半径,然后向外舍入.

GetPhysicalExtent()返回场景中影片的实际范围.现实相机特别需要此信息.给定胶片对角线的长度和图像的长宽比,我们可以计算和方向上传感器的大小.如果用表示对角线长度,用和表示胶片传感器的宽度和高度,那么我们知道.我们可以将定义为图像的长宽比,因此,得出.求解给出

直接执行GetPhysicalExtent()的实现.返回的范围以(0,0)为中心.

7.9.2 为胶片提供像素值

将样本贡献提供给Film有三种方式.第一种由采样器在图像图块上生成的采样驱动.虽然最直接的接口是允许渲染器提供胶片像素位置和“光谱”,并将相应的光线直接贡献给胶片,但在存在多线程的情况下提供此类方法的高性能实现并非易事,其中多个线程可能最终试图同时更新映像的相同部分.

因此，Film定义了一个接口,线程可以在相对于整个图像的子区域指定,它们在一定程度上以像素为单位生成样本.给定样本边界,GetFilmTile()依次返回一个指向FilmTile对象的指针,该对象存储图像相应区域中像素的贡献.FilmTile及其存储的数据的所有权是调用方专有的,因此线程可以向FilmTile提供样本值,而不必担心与其他线程的争用.完成在图块上的工作后,线程会将完成的图块传递回胶片,胶片将其安全地合并到最终图像中.

给定将在其中生成样本的像素区域的边界框,有两个步骤来计算样本值将贡献的图像像素的边界框.首先,必须考虑离散到连续像素坐标转换的影响以及滤波器的半径.第二,结果边界必须裁剪到整个图像像素边界;根据定义,图像外的像素无需考虑.

FilmTile构造函数使用一个2D边界框,该边界框提供了最终图像中必须提供存储空间的像素边界,以及有关所使用的重构滤镜的其他信息,包括指向在<Precompute filter weight table>.

对于每个像素,像素采样的加权贡献之和(根据重建滤波器权重)和滤波器权重之和均被保持.

一旦计算出射线所承载的辐射强度,积分器就会调用FilmTile::AddSample().它会获取一个样本及其对应的辐射值,以及最初由Camera::GenerateRayDifferentia()返回的样本贡献的权重.它使用带有像素滤波方程的重建滤波器更新存储的图像.

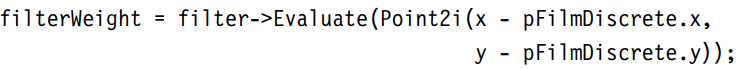
要了解FilmTile ::AddSample()的操作,请首先调用像素过滤方程式:

它使用滤波函数和相机返回的样本权重来计算每个像素的值作为附近样本辐射值的加权和,以计算辐射值到最终像素值的贡献.由于pbrt中的所有“滤波器”都具有有限的范围,因此该方法首先计算当前样本将影响哪些像素.然后,将像素滤波方程式反过来,它为受样本影响的每个像素更新两个运行总和.一个求和累积像素滤波方程的分子,而另一个求和累积分母.在渲染结束时,通过执行除法来计算最终像素值.

为了找到样本可能对哪些像素起作用,FilmTile::AddSample()通过从x和y减去0.5将连续样本坐标转换为离散坐标.然后,它在每个方向上通过滤镜半径偏移此值(图7.48),将其转换为图块坐标空间,并采用最小坐标的上限和最大值的底限,因为范围范围外的像素不受影响.最后,像素边界被裁剪到图块中像素的边界.虽然样本理论上可能对图块外部的像素有所贡献,但任何此类像素都必须在图像范围之外.

给定受此样本影响的像素范围,现在可以遍历所有这些像素,并在每个像素处累积过滤后的样本权重.

每个离散的整数像素均以其为中心具有滤波器函数的一个实例.要计算特定样本的过滤器权重,有必要在离散坐标中找到从像素到样本位置的偏移量,并评估过滤器函数.如果我们明确评估过滤器,则适当的计算将是



但是,实现从表中检索适当的过滤器权重.

为了在给定采样位置的情况下找到像素的滤波器权重,此例程计算偏移量并将其转换为滤波器权重的坐标查找表.这可以通过以下方式直接完成:将采样偏移量的每个分量除以该方向上的滤波器半径,得到0到1之间的值,然后乘以表大小.通过注意到沿着方向上的每一行像素,的差以及因此进入过滤器表的y偏移是恒定的，可以进一步优化此过程.类似地,对于像素的每一列,偏移是恒定的.因此,在此处循环像素之前,可以预先计算这些索引并将其存储在两个1D数组中,从而节省了循环中的重复工作.

现在,在每个像素处,可以找到该像素在滤波器表中的x和y偏移量,从而导致该数组中的偏移量以及滤波器值.

现在，对于每个受影响的像素,我们可以将其加权频谱贡献和滤镜权重添加到像素阵列中的适当值.

GetPixel()方法使用相对于整个图像的像素坐标,并将它们转换为胶片图块中的坐标,然后再索引到像素数组中.除了此处的版本外,该方法还有一个const变体,可返回const FilmTilePixel＆.

渲染线程呈现FilmTiles,以使用MergeFilmTile()方法将其合并到Film存储的图像中.它的实现始于获取互斥锁,以确保多个线程不会同时修改图像像素值.请注意,由于MergeFilmTile()将std :: unique\_ptr带到图块,因此调用此方法时,将转移图块内存的所有权.因此,在调用此方法后,调用代码不应再尝试向图块添加内容.当tile参数超出范围时,在执行MergeFilmTile()时会自动释放FilmTile的存储.

当tile参数超出范围时,在执行MergeFilmTile（）时会自动释放FilmTile.

将图块的贡献合并到最终图像中时,调用代码必须能够找到图块具有贡献的像素范围.

对于某些Integrator实施而言,仅一次为整个图像中的所有像素提供值也很有用.SetImage()方法允许这种操作模式.请注意,image参数指向的数组中的元素数应等于cropedPixelBounds.Area().SetImage()的实现是将给定值转换为XYZ颜色后直接复制的简单问题.

一些光传输算法(尤其是在16.3节中引入的双向路径跟踪)要求能够“散布”对任意像素的贡献.而不是将最终像素值计算为贡献散点图的加权平均值,而是简单地将散点图相加.通常,给定像素周围的碎片越多,像素将越亮.Pixel::splatXYZ成员变量声明为AtomicFloat类型,该变量允许多个线程通过AddSplat()方法同时更新像素值,而无需额外的同步.

7.9.3 图像输出

在主渲染循环退出后,积分器(Integrator)的Render()方法通常调用Film::WriteImage()方法,该方法指示Film进行必要的处理以生成最终图像并将其存储在文件中.此方法采用比例因子,该比例因子应用于AddSplat()方法提供的样本.(有关在MLTIntegrator中使用此比例因子的进一步讨论,请参见第16.4.5节的结尾.)

此方法首先分配一个数组来存储最终的RGB像素值.然后,它循环遍历图像中的所有像素以填充此数组.