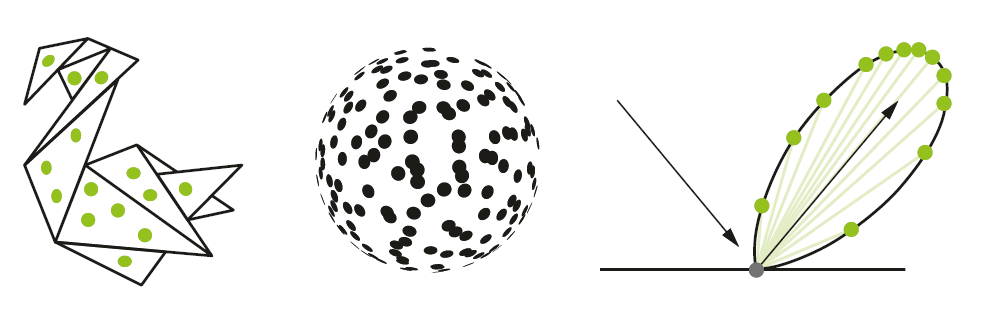
第16章 取样变换集合



概要

我们在这里提供一些公式与方法，用以生成在特定域中在期望的概率密度函数上分布的样本。在渲染中取样是非常重要的一项操作，无论是在预处理时还是在程序运行时。随着光线追踪在标准API中的出现，它变得更加普及，因为许多光线追踪算法本质上都是基于取样的。这一章节简要地列举了一些实用的技巧与方法。

16.1 取样方式

在光线追踪程序中一项通常的任务就是基于某个包含潜在概率密度函数（Probability density function, PDF）的域，选择一组样本：例如，一组在单位半球上，概率密度与极角余弦成正比的点。这通常会通过取一组单位超立方体上的均匀样本，然后将它们变换到目标域的方式来完成。对这类一般的取样步骤不熟悉的读者可以参考Pharr et al. 第13章 [8] 。

这一章节列出了各类作者认为在光线追踪程序中有效的产生特定分布的方法。它们都曾被发表过，或者属于“常识”的一部分。

16.2 分布介绍

在一维中，有一种标准的变换方式来生成特定PDF ρ的样本。这个方法的关键是使用累积分布函数（cumulative distribution function, CDF），通常用大写的P(x)表示：

P(x) = 均匀分布样本 的概率. (1)

为了说明这个函数的作用，首先假设我们想要知道一个特定的均匀分布中的值在代入我们的Warping函数 之后会落在哪里。如果我们假定不递减（即它的导数非负），那么一半样本点会映射到小于 的值，另一半映射到大于 的值。由于CDF本身的特质，当时我们知道PDF下面的一半面积会在的左边，所以我们可以推出

. (2)

这个基础原理同样也适用于之外的任何。因此，我们得到

, (3)

其中P-1是P的反函数。反函数的表示方法可能不够清晰。它实际的数学意义是，对于一个PDF ρ，我们用等式1中的积分公式求它的积分，然后在得出的等式中求解（即求P的反函数）:

对于一系列均匀分布的样本，我们取P的反来得到p分布的样本组x。

二维域需要两个均匀分布的样本，和。它们加起来可以确定在二维单位立方上的一个点：。它们同样可以变换到目标域。

例如，想要在单位圆盘（disk）上选取均匀分布的样本，我们可以写下一个极坐标系中的积分方程，测度为，其中r代表从原点开始以ϕ角度沿x正轴的距离（即半径）。这在二维空间中是可行的，但是两个维度会被分割成两个独立的一维PDF，因此应当小心处理测度中包含的变量，如r。虽然在单位圆盘上方程 的密度是均匀的，但在分割成两个独立的一维密度后，r就会从属于半径的密度。由此得到的两个一维PDF是：

. (4)

常数项 和2使得这两个PDF积分为1（如前文所述，这是PDF所需的特性）。如果我们计算这两个PDF的CDF，我们会得到

. (5)

如果我们想要基于这两个CDF来变换均匀样本和，我们可以对每个一维CDF代入等式2：

, (6)

然后解出ϕ和r，得到：

(7)

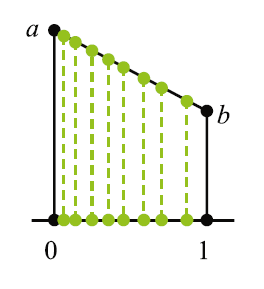
这种基本手法被用于大部分文献中的变换。

重要的一点是，我们的方法假定在单位立方体上是均匀分布的，但是在更高的维度上面这些点是均匀分布在单位超立方体上的。这些样本可以通过（伪）随机或拟随机方法生成[6]。

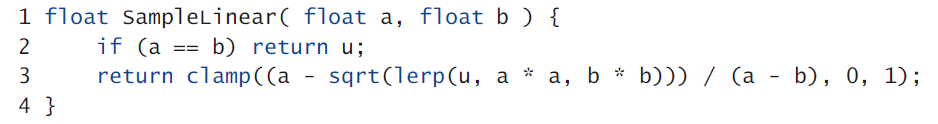
这章的剩余内容介绍了几种我们认为在光线追踪程序中有用的变换方法，它们通常不包含求导，且大多在二维空间内。

16.3 一维分布

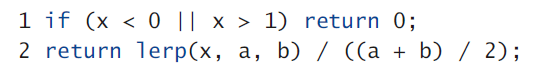
16.3.1 线性



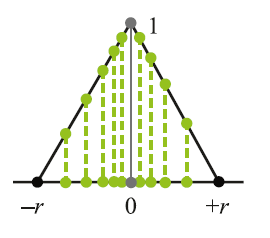
现有在上的线性方程，，，有均匀分布的样本，以下代码可生成基于分布的值：



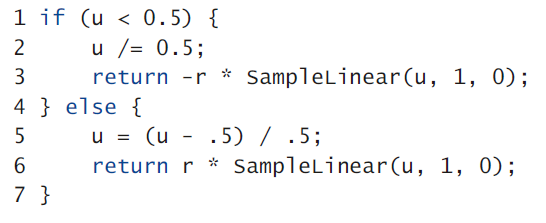
样本的PDF值可用以下代码得到：



16.3.2 帐篷

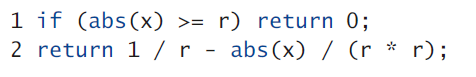


帐篷函数（tent function，又称三角形函数）由宽度与一对线性方程定义：在时它的值为0，在原点时值为1，在又变回0。以下代码中的SampleLinear() 函数实现了16.3.1章节描述的功能：

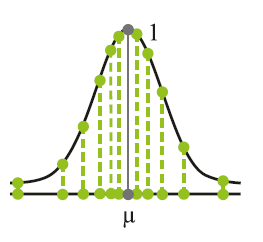


现在我们可以用均匀分布的样本来选择一半帐篷函数，然后将样本映射回以对正确的线性函数进行取样。

一个在点取样的PDF的计算方式如下：



16.3.3 正态分布



正态分布有以下定义：

. (8)

（译者：原文为It has infinite support，不知要表达什么，没有翻译）但一旦的几倍时，它的值下降得就很快。从这个分布中解析生成一个单一的样本是不可能的，因为这需要取误差函数的反，

, (9)

这在闭式（closed form）中是不可行的。一种方法是用反函数的近似多项式，我们表示为。由此，可以生成样本



样本的PDF即为



如果需要多于一个样本，可用Box-Muller变换基于均匀分布的样本在正态分布中生成两个样本：



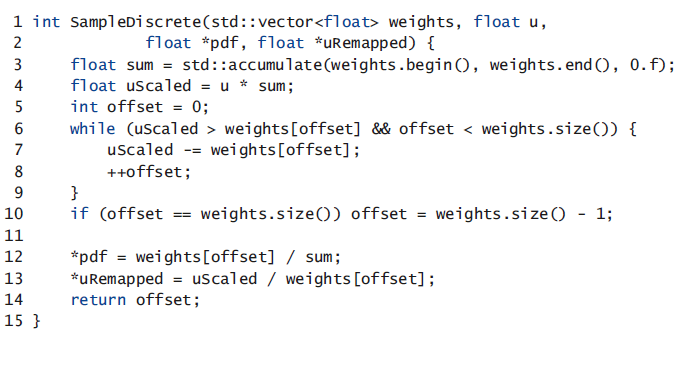
16.3.4 从一维离散分布中取样

对于一个浮点数数组，有几种方法从它们中选出一个，使得这个数被选中的概率与它的相对大小成正比。我们在这里介绍两种方法：一种更适用于只需要一个样本的情况，另一种更适用于需要多个样本的情况。

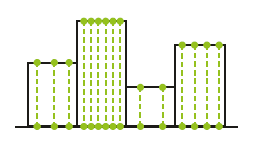
16.3.4.1 取样一次

如果只需要一个样本，那么可以使用以下代码中的函数。它计算了所有值的和（假定它们全部为非负），然后缩放已有的均匀分布的样本，将它从域重新映射到。然后它遍历数组，从映射的样本中减去每个数组元素的值。一旦减去下一个数将会让这个值变负，它就会停下来。

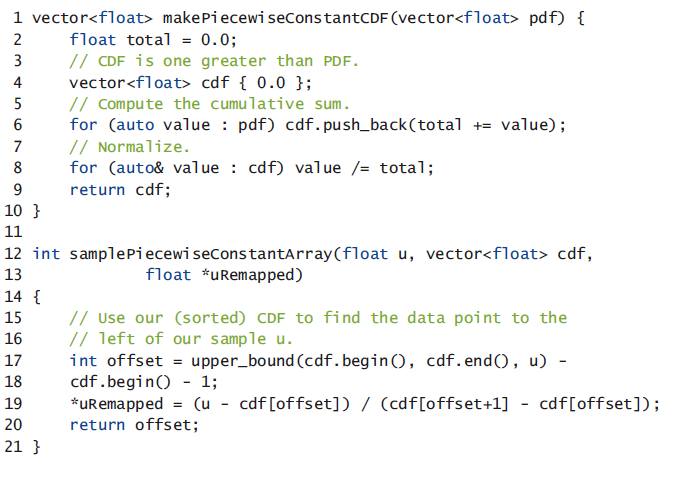
这个函数也返回PDF，用以选择一个数组元素，以及一个基于原样本值映射到的样本值。直觉上说，样本依然会包含均匀分布，因为我们只是用它来做一个离散取样。但是，均匀分布的数值点可能太少，以至于这个样本不能被再次使用，尤其是被选中的事件只有很小的概率会发生时。

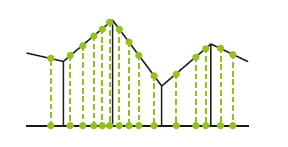


16.3.4.2 取样多次



如果一个数组需要被多次取样，更高效的做法就是提前计算数组的CDF，对每个样本做二分法检索。必须注意区分分段常数和分段线性数据，因为CDF计算与取样对于它们是不同的。例如，我们可以用下列代码从分段常数分布中取样：

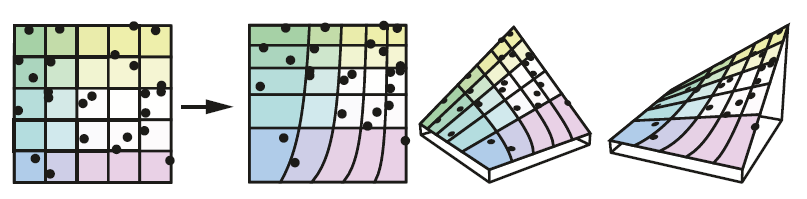




在分段线性分布中取样时，CDF可以通过计算一对样本间的不规则四边形的面积得到。从分布中取样需要在二分法检索后，用16.3.1章节提到的SampleLinear()函数从线性部分中取样。如果使用C++，C++11的标准模板库（Standard Template Library）中的random模块包含了piecewise\_constant\_distribution以及piecewise\_linear\_distribution。

16.4 二维分布

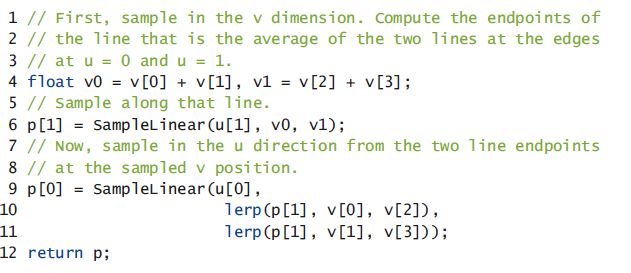
16.4.1 双线性



比较有效的方法是从双线性插值函数中取样。我们将之定义为用四个值定义一个在区间的函数

. (10)

然后，根据两个均匀分布的样本和，可以从中先后取两个维度上的样本。这里，我们使用在16.3.1章节中定义过的一维线性取样函数，SampleLinear()：

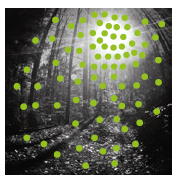


样本值的PDF如下：

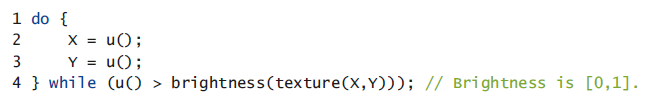


16.4.2. 二维纹理分布

16.4.2.1 拒绝采样



想要在纹理图中以与纹理像素亮度成比例的概率选择一个像素点，一种简单的方法是使用拒绝采样，这样可以均匀地选择像素点，并且一个样本只有在像素的亮度大于另一个均匀分布的值时才被认为是有效的：



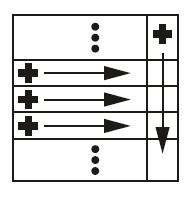
注意纹理图使用拒绝采样的效率与该纹理图的平均亮度成正比，因此，如果担心性能方面的问题，应当避免对稀疏的（大多为暗色的）纹理图使用这个方法。

16.4.2.2 多维反演法

为了在二维中取样一种纹理，我们可以在16.3.4.2章节（从一维数组中取样）采用方法的基础上，从两个分布，垂直分布与水平分布中取样：

>对每一行像素点建立亮度的CDF表格（累积分布），然后归一化（normalize）。

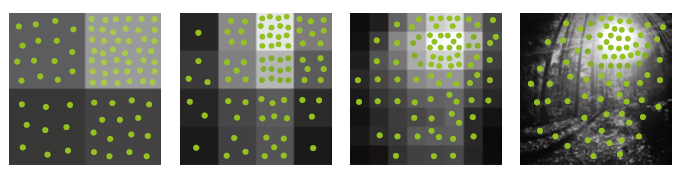
>对最后一列（每一行亮度的总和）建立CDF，然后归一化。



>为了从纹理分布中取样，首先需要取得一个均匀的二维样本。用二分式搜索所有列的CDF，找出哪一行需要被用到。然后，用来二分式搜索所有行，找到样本所在的列。得到的坐标（column, row）就是按照纹理分布的。

这种反演法的缺点是它不能很好地保留样本点分层的特性（例如，蓝噪音或低差异的点）。如果这是一个问题，那么更好的方法是在二维中分层取样，如下文讨论的那样。

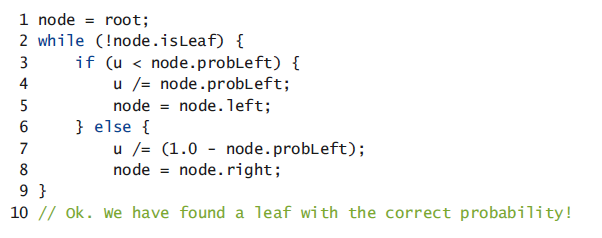
16.4.2.3 分层变换



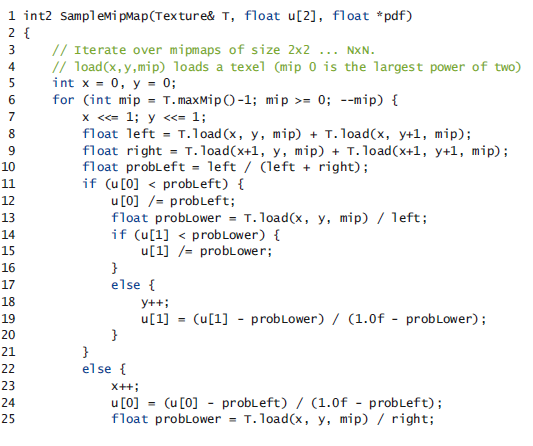
Hierarchical warping（译者：不知此处该如何翻译）用于优化上一章节描述的逆变换采样的不足之处，即基于行和列的逆变换映射可能导致样本聚集成群。我们首先要提醒大家，hierarchical warping不能完全解决连续性和分层的问题，尤其是当使用相关联的样本，如蓝噪音或低差异的点时，但这是一种从纹理取样时保留一些空间相关性的实用方式。Hierarchical warping的应用实例包括对于复杂光源的重点取样法[3,7]。

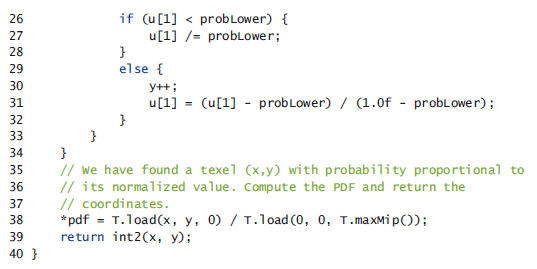
这个方法的核心是建一棵条件概率树，在每一个节点上存储它的子节点的相对重要程度。取样从根部出发，基于均匀分布的样本，在每个节点用概率决定选择哪个子节点。在每一步重映射均匀样本，而不是在每一层创建新的均匀样本，会使得这种算法更高效，生成更好的分布。生成的取样概率的说明可以参考Clarberg et al.的文章[2]。

这个方法不仅仅局限于二维中的离散分布取样。概率树可以是二分树，四分树，或者八分树，取决于所在的域。以下的伪代码展示了这个方法在二分树上的应用：



对于二维纹理图，它的实现尤其简单，因为我们可以基于纹理的mipmap层级直接取样。从纹理像素mipmap开始，我们可以基于纹理像素值计算出条件概率，先是水平方向，再是垂直方向。最优的最终分布由二维均匀分布的样本得到：





需要注意的是，对于所有这些涉及到重新映射一个或多个均匀分布样本的方法，一些数字精度可能会损失。输入值大多格式为32位浮点数，这就意味着当我们取样叶子的时候，可能只剩下几位的精度了。实际情况下这对于通常大小的纹理图一般不是什么问题，但我们依然需要了解。对于更高的精度，我们可以选择在每一步产生新的均匀分布的样本，但是那样分层的特性可能就会丢失。

另一个有用的小技巧是，树中每一层的概率没有必要都等于它之下所有节点的总和。如果不等于的话，我们可以简单地在运算过程中，通过将每一步所选中的概率相乘来计算样本的概率密度方程。这导致了允许全局概率密度函数在最开始未知，而在计算中产生的函数取样算法的诞生。