Dezeming Family

PBRT 系列 20-专业知识理论与代码实战-渲染概率与采样

Art. No 0
ISBN 000–00–0000–00–0
Edition 0.0

Cover design by Dezeming Family

Published by Dezeming
Printed in China

目录

前言及简介

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 *DezemingFamily* 的系列书，可以从我们的网站 *[https://dezeming.top/]* 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

0.1 本书前言

在基于概率的渲染中，还没有一本合适的书能够囊括所有基础知识，各种资料的介绍也都不太一样，因此我想写一本关于概率方面的小书。

本书的目标是系统地讲解基于概率的渲染方法。与前面的系列书不同，本书不再是笔记的整理和完善，而是完全从头到尾书写的内容。本想先写物理材质，但是考虑到物理材质的理解也需要一定的概率知识，因此先介绍概率与采样。

本书不但注重基本原理的讲解，还注重公式的推导，力求全面与详细，会有很多基本原理的分析与讨论。本书在书写中耗费了我们大量的时间和精力，并对内容来回反复进行了多次修改和规划，希望能够更好得帮助到读者。

本书的售价是 12 元（电子版），我们不直接收取任何费用，如果本书对大家学习有帮助，可以往我们的支付宝账户（17853140351，备注：PBRTH）进行支持，您的赞助将是我们 Dezeming Family 继续创作各种图形学、机器学习、以及数学原理小册子的动力！

20211104：完成第一版。

20211108：对 MIS 方法进行了一些内容的补充和修改。

20220810：对体渲染部分进行了大量的修改，增加了不少细节上的描述。

1. 概率与蒙特卡洛采样

本章首先描述基本的概率知识，然后介绍蒙特卡洛求积分、重要性采样技术。然后讲解对光源重要性采样的方法。

1.1 概率与概率密度函数

我们都知道什么是概率，比如我们有一个完美的骰子，投掷之后，正上面为 1 的概率是 1/6。对于概率密度函数，表示如下：

$$p(x) \geq 0, \int_{-\infty}^{\infty} p(x)dx = 1 \tag{1.1.1}$$

我们定义累积概率密度函数 CDF：

$$P(y) = p(x \leq y) = \int_{-\infty}^{y} p(x)dx \tag{1.1.2}$$

$$p(x_1 < x < x_2) = \int_{x_1}^{x_2} p(x)dx = P(x_2) - P(x_1) \tag{1.1.3}$$

假如我们随机在 [0,2) 上取一个点，那么取到任意一个点的概率就为 0，但是我们确实能取到其中的一个点，因此，用概率来表示就不行了，我们用概率密度来表示取到任意一个点的概率密度为 0.5，因此取到的点在 [0.4,0.8] 区间范围的概率为：

$$p(0.4 < x < 0.8) = \int_{0.4}^{0.8} 0.5dx = 0.2 \tag{1.1.4}$$

1.2 求函数积分

我们求一个函数积分：

$$F = \int_{0}^{2} (x^2 + x)dx \tag{1.2.1}$$

其实也可以写为：

$$F = 2 \times average(x^2 + x) \tag{1.2.2}$$

因此我们可以随机产生一堆 [0,2] 之间的随机数，然后求函数值，并取平均，然后最后再乘以 2，得到估计值。

我们假设一个随机变量 x 在区域 [a,b] 的概率密度为 p(x)，则 $f(x)$ 在该区域的平均值就为：

$$E(f(x)) = \int_a^b f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{1.2.3}$$

可以想象，上面的 $p(x)$ 既可以是均匀分布的，也可以是非均匀分布的。均匀分布时，期望就简化为（参考微积分中“函数的平均值”）：

$$E(f(x)) = \int_a^b f(x)\frac{1}{b-a}dx \tag{1.2.4}$$

如果是非均匀分布，得到的期望就不是均匀积分得到的期望，我们得到的 $f(x)$ 值是在一定样本分布下的函数平均值。

很多时候，$f(x)$ 在积分区间的概率是均匀分布的，就可以通过均匀生成的样本 x_i 计算 $f(x_i)$ 然后取平均来估计函数平均值：

$$E(f(x)) \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{1.2.5}$$

但是我们很多时候不希望采样的样本是均匀分布的。我们希望采样的样本在 [0,1) 之间的概率应该要小于 [1,2) 之间的概率，因为比如我们同样采样 1000 次得到的误差在 1% 范围内，积分值大的区间，误差 1% 的值要大于积分值小的区间，所以我们希望在积分值大的区间里多采样一些点。

因此，如果我们希望采样点 x 在积分区间不均匀分布：

$$\int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)} \tag{1.2.6}$$

$$p(x) \geq 0, \int_{-\infty}^{\infty} p(x)dx = 1 \tag{1.2.7}$$

我们可以以此来选择自己需要的 $p(x)$ 进行函数估计。上式的理解我们可以把 $\frac{f(x)}{p(x)}p(x)$ 拆分为两部分，一部分是 $f = \frac{f(x)}{p(x)}$，另一部分是 $p(x)$，表示 f 在积分区间 [a,b] 之间的概率分布。其中，x_i 的分布符合概率密度函数 $p(x)$。

1.3 生成随机变量

那么现在还剩一个问题，怎么产生概率密度为 p(x) 的随机变量呢？

反函数法

由于反函数法最基本也最重要，所以我们当前研究反函数法。

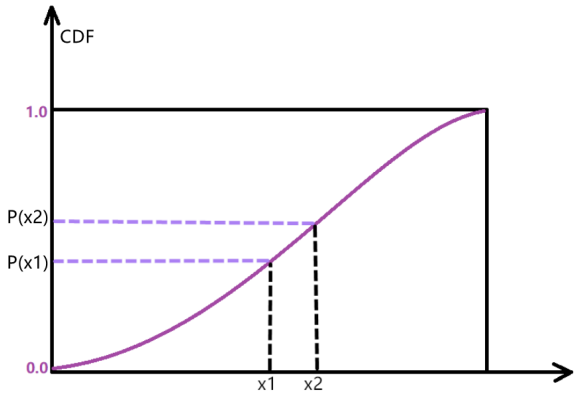
我们先从直觉上进行理解，假如目标概率密度函数是 $p(x)$，累积概率密度函数为 $P(x)$。我们设 $x \in [0,2]$，设 $P(0)=0$，$P(1)=0.25$，$P(2)=1.0$，可以看出，如果我们产生了 N 个数据，按理来说，25% 的数据是在 $[0,1)$ 之间的，75% 的数据是在 $[1,2]$ 之间的。我们设一个函数 $F()$，该函数输入 0.2 以后会得到 1，输入 1.0 以后会得到 2，其他点也是同理，我们设 $p(x)=0.5x$：

$$P(x) = \int p(x)dx = \frac{1}{4}x^2 \tag{1.3.1}$$

$$\begin{cases} P(0)=0 \\ P(0.1)=\frac{1}{400} \\ P(0.2)=0.01 \\ P(0.5)=\frac{1}{16} \\ P(0.8)=0.16 \\ P(1.0)=0.25 \end{cases} \Longrightarrow \begin{cases} F(P(0))=0 \\ F(P(0.1))=0.1 \\ F(P(0.2))=0.2 \\ F(P(0.5))=0.5 \\ F(P(0.8))=0.8 \\ F(P(1.0))=1.0 \end{cases} \Longrightarrow F() = P^{-1}() \tag{1.3.2}$$

则当我们需要一个概率密度为 $p(x)$ 的随机数时，我们可以将 $[0,1]$ 之间的均匀概率密度随机数 x_i 通过函数逆变换来得到：

$$a = P^{-1}(x_i) \tag{1.3.3}$$

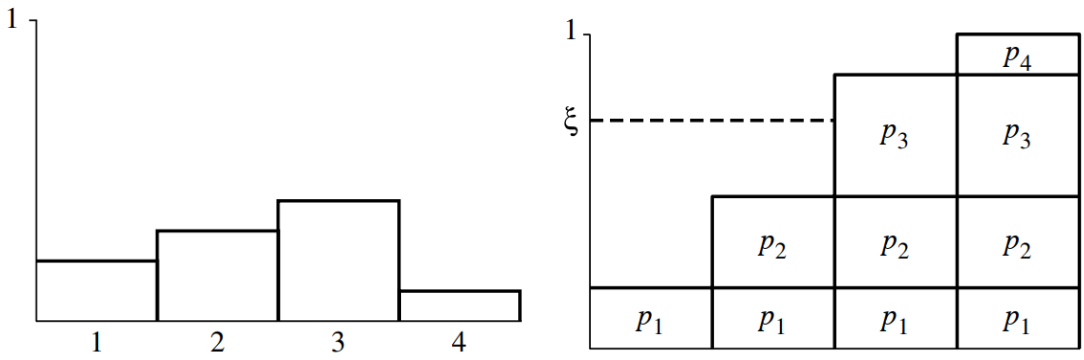


我们可以这么考虑：

$$p(x_1 \le x \le x_2) = P(x_2) - P(x_1) \tag{1.3.4}$$

如果 $p(x_1 \le x \le x_2)$ 的值更大，说明样本更有可能产生在 $[x_1,x_2]$ 之间。我们可以想象纵轴是 $[0,1]$ 之间的均匀随机数，这样通过逆变换得到的值就是符合 $p(x)$ 概率分布的值了。

离散情况更好理解：



左边是概率，右边是累计概率。离散情况下，一共有可能生成 4 个不同随机数，假设为 1,2,3,4，随机数为 i 的概率分别为 p_i。当我们均匀采样中获得了某个 [0,1] 之间的随机数时，如右图，通过反函数法，反变换为 3。

接受拒绝法

假如我们可以采样某种概率密度分布 $g(x)$，其 CDF 为 $G(x)$。我们取某个常数 C，保证任意 x，$C * g(x) \geq f(x)$ 成立（满足该条件的 C 越小采样的效率越高），那么接受-拒绝法表示为：

[1]. 从分布 G 中采样，获得样本 x_i。

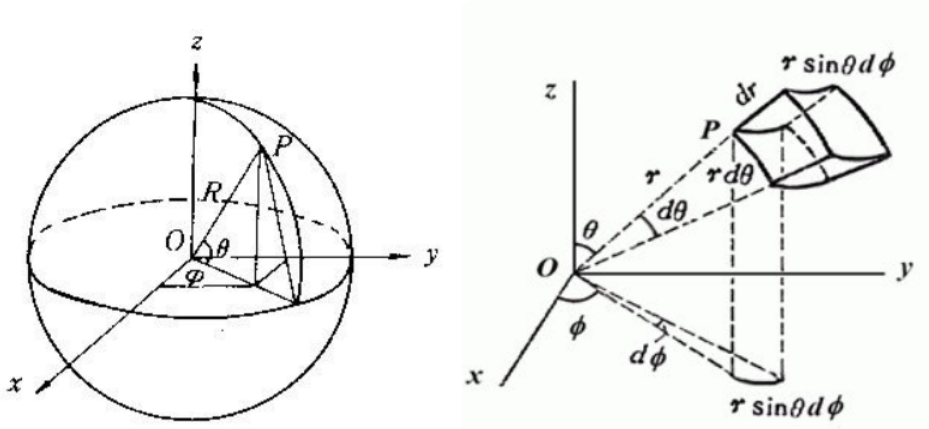
[2]. 从 $[0,1]$ 中均匀采样，获得样本 y_i。

[3]. 如果 $\frac{f(x_i)}{C*g(x_i)} \geq U$，则接受该样本 x_i，否则拒绝该样本值。

该方法可能需要拒绝很多次样本以后才能得到一个样本，相比于反函数法，它的效率并没有那么高。

1.4 生成随机方向

给定一个球面方向的 Pdf，我们假设该 Pdf 只与 θ 方向有关，设为 $p(\theta)$:



注意球面积分面积 $dA = r^2 sin\theta d\theta d\phi$。在 ϕ 和 θ 方向上的一维概率密度函数分别是：

$$p_\phi(\phi) = \frac{1}{2\pi} \tag{1.4.1}$$

$$p_\theta(\theta) = 2\pi sin\theta p(\theta) \tag{1.4.2}$$

这里的 $2\pi sin\theta$ 其实就是对于某个 θ 上的一个圆周。
因此我们可以设两个均匀随机数:

$$r_1 = \int_0^\phi (\frac{1}{2\pi}) \tag{1.4.3}$$

$$r_2 = \int_0^\theta 2\pi sin(t)p(t) \tag{1.4.4}$$

解出用 r_1 和 r_2 表示的 ϕ 和 θ 即可。然后使用下面的转换公式转换为 xyz 坐标：

$$x = cos(phi)sin(theta) \tag{1.4.5}$$

$$y = sin(phi)sin(theta) \tag{1.4.6}$$

$$z = cos(theta) \tag{1.4.7}$$

上式需要注意的是，$p(t)$ 需要满足的关系是：

$$\int_0^\pi 2\pi sin(t)p(t) = 1 \tag{1.4.8}$$

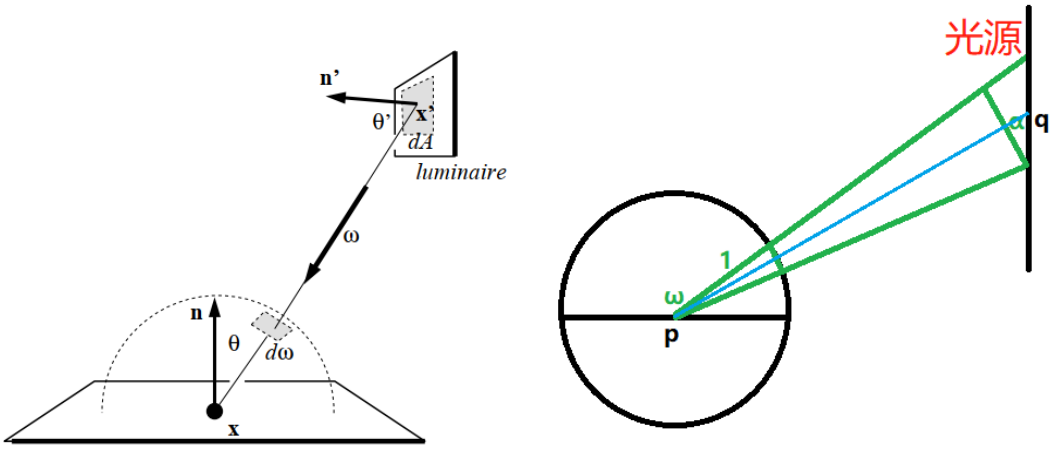
但因为有时候我们只需要采样半球面，因此上式就变成了：

$$\int_0^{\frac{\pi}{2}} 2\pi sin(t)p(t) = 1 \tag{1.4.9}$$

$$p(t) \geq 0,\ t \in [0, \frac{\pi}{2}] \tag{1.4.10}$$

1.5 对光采样

对于直接光照，如果光源某个点与被照射表面没有遮挡，则：



上图球面为单位球面，因此球面上的面积值等于其对应的立体角，我们可以根据三角形相似性得到关系：

$$\frac{d\omega}{dA\cos\alpha} = \frac{1}{\|p-q\|^2} \tag{1.5.1}$$

A 表示立体角投射到光源上时对应的面积。我们根据概率可以知道：

$$\int_S p(direction)d\omega = \int_A p(A)dA = 1 \tag{1.5.2}$$

即在半球面上概率的积分等于在投影方向概率的积分。我们需要注意的是：$p(direction)$ 在能投影到光面上时大于 0，在投影不到面光源上时为 0（因为我们只想对光的方向采样）。

我们先考虑如果微元上 $\cos\alpha = 1$，也就是说面光源的这个微元正好与单位球面的微元平行，此时可以很容易得到 $p(direction)d\omega = p(A)dA$，因为它们对应的概率密度必须是一样的（我们需要产生的光线全部都能与光源相交，因此两个概率密度必须要是一致的）。

当 $\cos\alpha \neq 1$ 时，积分微元之间也是一一对应的关系，$p(direction)d\omega = p(A)dA$，因此在球面微元不与面光源平行时，$p(direction)d\omega = p(A)dA$ 也是成立的。

因此我们考虑下面这个对应关系：

$$p(direction)d\omega = p(A)dA \tag{1.5.3}$$

$$d\omega = \frac{dA\cos\alpha}{\|p-q\|^2} \tag{1.5.4}$$

$$p(A) = \frac{1}{A} \tag{1.5.5}$$

$d\omega$ 来自于上面的三角形相似关系，因为 dA 在光面上是均匀分布的，所以概率密度为 $\frac{1}{A}$。从而推导出：

$$p(direction) = \frac{\|p-q\|^2}{Acos\alpha} \tag{1.5.6}$$

在 PBRT 源码中，DiffuseAreaLight::Sample_Li 中就是这样计算采样面光源的 Pdf 的：

```
Interaction pShape = shape->Sample(ref, u, pdf);
```

调用了 Shape::Sample 函数：

```
// 这里的pdf是在面上均匀采样得到的：1/Area
Interaction intr = Sample(u, pdf);
......
// 得到的结果就是上述p(direction)
*pdf *= DistanceSquared(ref.p, intr.p) / AbsDot(intr.n, -wi);
```

1.6 采样 BSDF

我们知道 BSDF 的内容是关于确定了入射方向 wi 以后，反射到其他方向的辐射分布。当然，对反射的各个方向的分布求积分并不一定等于入射能量，这是因为材质表面会吸收颜色。比如绿叶会吸收红蓝色，反射绿色。

当我们给定出射方向 wo 时，我们需要根据 BSDF 来采样一个入射方向，并为该方向分配一个 BSDF。对于非完全漫反射的材质，不同的 wi 入射方向反射到 wo 的比例是不同的，有的方向反射的比例高一些，有的方向反射的比例低一些。

值得一提的是,BSDF 分为对称和非对称,对称的 BSDF 的一个最重要的特性之一就是 Helmholtz 互异性，也就是说当入射角和出射角互换时，BSDF 函数值不变，当我们令 wo 的反方向表示入射时，就能知道 wi 的反方向作为出射的比例；非对称的 BSDF 则不同，$f(\omega_i, \omega_o) \neq f(\omega_o, \omega_i)$。

只要我们有了 BSDF，我们就可以根据 BSDF 的形式来选择采样的方法，让采样的分布与 BSDF 的分布一致，这就是重要性采样 BSDF 的方法。对于复杂的微表面模型来说，根据微表面 BSDF 分布（包含法线分布项、几何遮蔽项、菲涅尔项的乘积）来采样非常困难，所以 PBRT 只对法线分布进行采样（因为很多微表面材质假设微表面都是镜面，所以采样法线是很有意义的），计算 Pdf 也只是计算法线分布项即可：

```
1  //MicrofacetReflection::Pdf
2  Vector3f wh = Normalize(wo + wi);
3       return distribution->Pdf(wo, wh) / (4 * Dot(wo, wh));
```

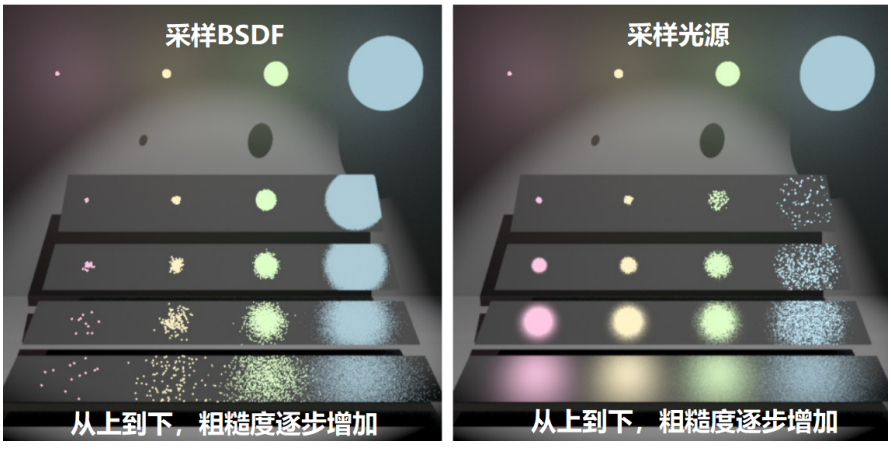
注意，在 PBRT 中，根据 BSDF 来采样光入射方向（也就是采样 Ray 的反弹反向）的函数是 BxDF::Sample_f() 函数。而根据光的入射和出射方向以及材质属性来计算光的反射比率（即计算 BxDF）的函数为 BxDF::f() 函数。

2. 概率方法深入

本章更深入地描述基于概率的渲染方法，阐述方法的原理和应用。重点在于概率算法的提出和作用，并包含有无偏性证明。本章是对上一章的基本概率方法的内容的升级，包括多重重要性采样、俄罗斯轮盘赌等常用的渲染算法。

2.1 MIS 多重重要性采样

蒙特卡洛路径追踪（MCPT）一共有两种方法进行采样，一种是采样 BRDF，根据表面散射分布概率，来发射相应概率的 Ray 进行跟踪。另一种是采样光：即在某个散射点上，我们直接从光源进行采样，然后计算采样到光采样点的概率。



而采样光源时，对于比较大的光源，采样的方向和镜面光方向一旦差别比较大就不算采样到，所以光滑的平板不容易采样到光源：路径追踪到一个散射点然后从光源采样，但是从光源采样到的点不一定能根据 BSDF（因为比较光滑，类似镜面反射）反射到人眼。而采样 BSDF 时，粗糙平面也很难采样到小光源。

MIS 方法的意义

首先，这两种策略都是作用在某个表面交点上的重要性采样技术，只是对应不同的情况，它们的采样方差不一样。这两种采样技术，其实都是运用在该点上的不同的概率分布罢了。对于粗糙面来说，BSDF 采样的方差会大很多，这是因为 BSDF 采样的 Ray 比较发散，很难对准比较小的光源位置。

由于被积函数通常是许多不同因子的和或积，并且过于复杂，无法直接得到函数形式，比如在渲染方程的积分，无法得到确定的被积函数形式。而样本的产生则是确定的：样本是从概率密度函数中选择的，该密度函数与因子的某个子集成比例（如上文所述的 BSDF 采样策略和光源采样策略）。当一个未考虑的因素对被积函数有很大影响时，这可能导致高方差（比如前面我们在积分光源的时候有时候只考虑了对光源重要性采样，有时候只考虑了对 BRDF 重要性采样）。

其实我们可以针对某物体的粗糙度来选择使用哪种策略——这肯定是可以的，比较粗糙的，同时光源比较小的时候，就用采样光的技术；比较光滑的，同时光源比较大，就用采样 BSDF 的技术。但是，对于多光源场景，以及光源大小如何判断等问题，以及如何确定光滑与不光滑的阈值等，这种方法是没有解决的办法的。

针对这类集成问题，一种方法称为多重重要性采样 MIS。它基于使用几种不同的技术进行采样的思想，旨在对被积函数的不同特征进行采样。例如，假设被积函数的形式为 $a_1 f_1 + a_2 f_2$，我们就可以采用对应每个 f_i 的重要性采样 p_i 来进行采样了。

MIS 方法关注的不是如何构造一套合适的抽样技术，甚至不是如何确定从每一种技术中应抽取的样本数量。相反，我们考虑的问题是，一旦采集了这些属于不同概率密度的样本，它们应该如何组合，即如何以一种无偏的方式来做到这一点，并且方差可以证明是接近最优的（当然一般也不是最优的，比如对粗糙物体小光源来说，没必要使用对 BSDF 积分）。

其实 MIS 的基本思路也非常简单，我们甚至可以这么操作：先用策略 A 采样 100 次来估计结果 $R1$，再用策略 B 采样 100 次来估计结果 $R2$，然后使用 $\frac{R1+R2}{2}$ 作为最终的估计值。这样做，如果在这种情况下，A 方法很差，B 方法很好，则得到的结果虽然并不如直接使用 B，但至少比直接使用 A 要好。在一个复杂场景中，光源有大有小，材质有光滑有粗糙，这种组合策略可以使我们不至于得到过于差的结果。只是 MIS 更高级的地方在于，它的组合策略会更精妙（需要一定的概率知识来证明），而不仅仅是用不同的采样方法采样然后取平均——但无论如何，我们都应该明白，MIS 方法不会比最好的方法好，也不会比最差的方法差，它是比较“折中”的方法。

路径追踪中，可以同时使用 BSDF 和光源采样策略进行采样。后面会讲解如何自动组合这些样本，以获得整个表面粗糙度和光源参数范围内的低方差结果。

MIS 方法的过程

假如我们要计算积分：

$$\int_a^b f(x)dx \tag{2.1.1}$$

如果我们使用了两种采样技术，最简单的思想就是，既然我用一种技术重要性采样可以收敛到真实值，另一种技术也可以收敛到真实值，那么我将这两种技术得到的结果相加再除以 2，不也是真实值么。当然，MIS 方法更巧妙一些。

我们设一共 n 种采样技术，第 i 个采样技术一共采样 n_i 个样本，来自第 i 种采样技术的第 j 个样本记为 $X_{i,j}$，$p_i(x)$ 表示采用采样技术 i 产生该样本 x 的概率密度。设 $\omega_i(x)$ 为样本 x 的组合权重，用一种采样方法估计积分值的公式是：

$$F = \frac{1}{n_i}\sum_{j=1}^{n_i}\frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{2.1.2}$$

同时使用多个采样方法得到的估计公式是：

$$F = \sum_{i=1}^{n}\frac{1}{n_i}\left(\sum_{j=1}^{n_i}\omega_i(X_{i,j})\frac{f(X_{i,j})}{p_i(X_{i,j})}\right) \tag{2.1.3}$$

需要保证的是：

$$\sum_{i=1}^{n}\omega_i(x) = 1 \ \ whenever \ \ f(x) \neq 0 \tag{2.1.4}$$

$$\omega_i(x) = 0 \ \ whenever \ \ p_i(x) = 0 \tag{2.1.5}$$

我们可以计算得到期望：

$$E[F] = E\left[\sum_{i=1}^{n}\frac{1}{n_i}\left(\sum_{j=1}^{n_i}\omega_i(X_{i,j})\frac{f(X_{i,j})}{p_i(X_{i,j})}\right)\right] \tag{2.1.6}$$

$$= \sum_{i=1}^{n}\frac{1}{n_i}\left(\sum_{j=1}^{n_i}\int_a^b\frac{\omega_i(x)f(x)}{p_i(x)}p_i(x)dx\right) \tag{2.1.7}$$

$$= \int_a^b\sum_{i=1}^{n}\omega_i(x)f(x)dx \tag{2.1.8}$$

$$= \int_a^b f(x)dx \tag{2.1.9}$$

所以只要保证上面的两个条件，就一定能收敛到真实值。

组合权重的计算方法

常用的权重组合方法有平衡启发式和幂启发式两种，经过 [6] 的证明，这两种方法方差都比较小，尽管比最优采样方法方差大，但最起码也不会太大。其中，幂启发式的方差比平衡启发式的还要小一些。

平衡启发式算法如下，其中，n_i 表示使用第 i 种采样技术采样的样本数：

$$\omega_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)} \tag{2.1.10}$$

这相当于当我们使用了某种方式产生了样本 x 的时候，我们还需要计算出其他采样方法产生 x 的概率密度，包括每种采样方法产生的总样本数。

假设我们有两种采样方法，分别采样一个样本，则：

$$F = \sum_{i=1}^{2}\frac{1}{1}\left(\frac{p_i(X_{i,1})}{p_1(X_{i,1}) + p_2(X_{i,1})}\frac{f(X_{i,1})}{p_i(X_{i,1})}\right) \tag{2.1.11}$$

$$= \left(\frac{p_1(X_{1,1})}{p_1(X_{1,1}) + p_2(X_{1,1})}\frac{f(X_{1,1})}{p_1(X_{1,1})} + \frac{p_2(X_{2,1})}{p_1(X_{2,1}) + p_2(X_{2,1})}\frac{f(X_{2,1})}{p_2(X_{2,1})}\right) \tag{2.1.12}$$

当一种采样方法采样了一个样本，另一种采样了两个样本时：

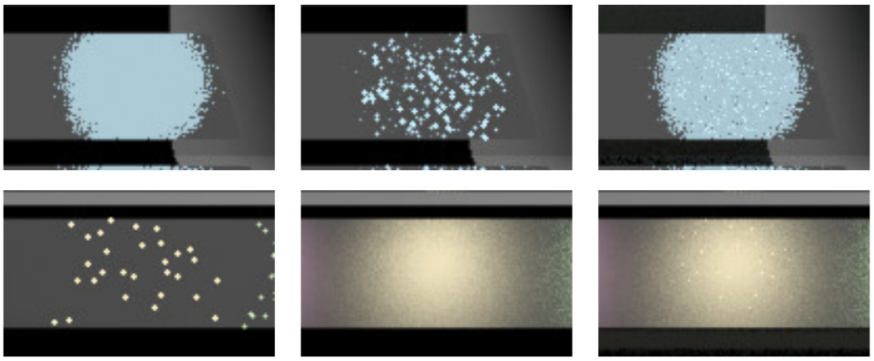
$$F = \frac{1}{1}\left(\frac{p_1(X_{1,1})}{p_1(X_{1,1}) + 2\cdot p_2(X_{1,1})}\frac{f(X_{1,1})}{p_1(X_{1,1})}\right) + \tag{2.1.13}$$

$$\frac{1}{2}\left(\frac{2\cdot p_2(X_{2,1})}{p_1(X_{2,1}) + 2\cdot p_2(X_{2,1})}\frac{f(X_{2,1})}{p_2(X_{2,1})} + \frac{2\cdot p_2(X_{2,2})}{p_1(X_{2,2}) + 2\cdot p_2(X_{2,2})}\frac{f(X_{2,2})}{p_2(X_{2,2})}\right) \tag{2.1.14}$$

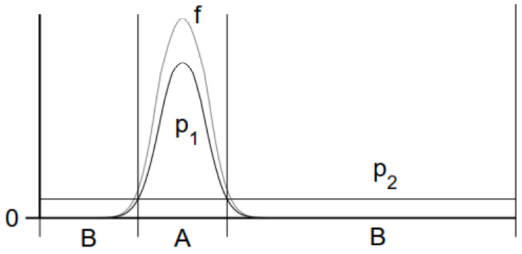
有趣的是，尽管这个权重并不能保证相加一定为 1：

$$\left(\frac{p_1(X_{1,1})}{p_1(X_{1,1}) + 2\cdot p_2(X_{1,1})} + \frac{p_2(X_{2,1})}{p_1(X_{2,1}) + 2\cdot p_2(X_{2,1})} + \frac{p_2(X_{2,2})}{p_1(X_{2,2}) + 2\cdot p_2(X_{2,2})}\right) \ not\ \ equal\ \ to\ \ 1 \tag{2.1.15}$$

但总的期望却是无偏的。



假如我们有两种采样方法 p_1 和 p_2，分别产生了两个样本，来估计函数 f 的积分值：



问题一：当从 p_1 抽样得到的样本值在 AB 边界附近，权重 $\omega_1 = p_1/(p_1 + p_2)$ 就小于 1，因此样本贡献就比最优采样 f/p_1 小很多。在渲染上可以理解为，某个比较光滑的物体，对光源采样之后恰好采样点对积分贡献为 0，对 BSDF 采样后能够采样到光源，但是一平衡，权重就变小了，导致暗了很多。

问题二：当从 p_2 采样到区域 A 时，虽然权重值很小，但是估计的贡献 $f/(p_1 + p_2)$ 的值确接近于 f/p_1。在渲染上可以理解为，对粗糙物体采样时，对光源采样得到了合理理想值，对 BSDF 采样也得到了很大的估计值，两者一平衡，估计值大于实际值，就会出现小亮点（当然这是低采样样本数的情况，毕竟无偏算法一定会收敛到真实值）。对于粗糙物体只对 BSDF 采样时，要么特别亮（碰巧击中光源，光照值除以很小的概率密度）要么特别暗（没击中光源），这种特别亮的情况和平衡启发式中出现的小亮点本质是一样的。

更好的策略其实就是让更大的权重函数更接近 1，让小的权重函数更接近 0。这对改善上面两种问题、减少方差都有很大好处。

幂启发式算法：

$$\omega_i(x) = \frac{(n_i p_i(x))^\beta}{\sum_k (n_k p_k(x))^\beta} \tag{2.1.16}$$

一般 β 取 2。

为什么叫“启发式”算法。是因为有种靠“猜、捏造”的成分。顺便提一句，我在做实际积分分析的时候，发现幂启发式对于平衡启发式在某些类型的积分区域确实有更好的作用，但是在某些区域反而加剧了方差。有想深入了解幂启发式的读者可以去文献 [6] 阅读。

代码实现

我们先看一下总体流程的代码，对于 UniformSampleOneLight 函数中选择的某个光源，使用 EstimateDirect 对其进行采样，我们看一下 EstimateDirect 的总体流程：

```
1  float lightPdf = 0, scatteringPdf = 0;
2  //采样光
3  Spectrum Li = light.Sample_Li(it, uLight, &wi, &lightPdf, &
       visibility);
4  //如果采样光的Pdf>0且光源不是黑的（即光有发出能量），就可以采样光源
5  if (lightPdf > 0 && !Li.IsBlack()){
6    ......
7  }
8  //如果当前光源不是Delta类型的光源（不是点光源、方向光源），就可以采
       样BSDF
9  if (!IsDeltaLight(light.flags)){
10   ......
11 }
12 return Ld;
```

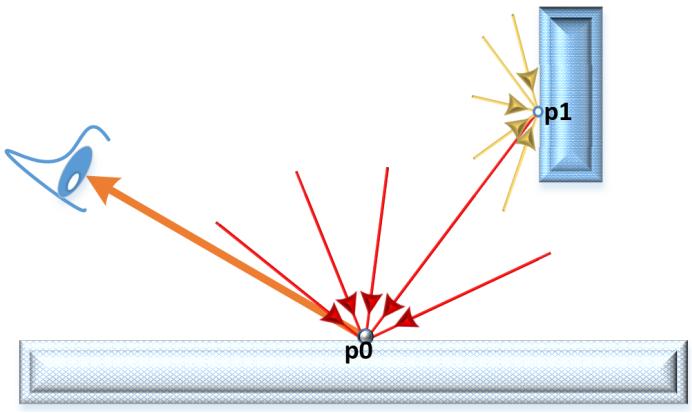
注意对光采样时也需要计算产生的 wi 在对 BSDF 采样时的概率密度，这样才能计算平衡启发式或者幂启发式；对 BSDF 采样时也是同理。

2.2 基于概率的渲染方法——路径追踪

回头重看了一下自己写的 PBRT 基础知识入门中的《路径追踪》一书，感觉内容有些粗糙（由于之前没有太多读者反馈，所以内容几乎没有更新）。本书将详细讲解一下路径追踪技术，关于路径追踪中使用俄罗斯轮盘赌的内容将放在下一节介绍。

路径追踪

在求解渲染方程时，当我们要计算从眼睛直接看到的点的 radiance 时，我们需要计算所有方向照射到该点以后反射出来的 radiance。而每个方向又需要计算其他所有方向反射过来的光：



这个逐步积分的过程就可以使用重要性采样。我们可以把 p0 射到眼睛的光分为两个部分，一部分是 p0 自身发出的光，另一部分是入射光从 p0 反射到眼睛的光。

我们把入射到眼睛的光分为光源从 p0 反射一次到人眼的光 $P(reflect\_1\_times)$、从光源照射到其他点（如 p1）再反射到 p0 再反射到人眼的光 $P(reflect\_2\_times)$、从光源出发总共反射三次到达人眼的光 $P(reflect\_3\_times)$……

反射一次

我们先以 $P(reflect\_1\_times)$ 为例，如果我们不计算 $P(reflect\_n\_times)$, $n>1$，那么我们得到的渲染积分器就是直接光照积分器。计算它很简单，就是只考虑直接光照射到 p0 再射入人眼即可，可以用 MIS 的方法求解。

设直接光照反射 n 次为 D_n，此时，我们计算出的人眼接受到的 radiance 描述为：

$$radiance = D_0 + P(reflect\_1\_times) \tag{2.2.1}$$

p0 本身不发光时，则：

$$radiance = P(reflect\_1\_times) \tag{2.2.2}$$

反射两次

我们思考 $P(reflect\_2\_times)$，我们需要采样一个方向，这个方向表示射入到 p0 的方向，当然我们可以使用表面材质 BSDF 的重要性采样来得到。我们需要注意的是我们采样的这个方向与最近表面的交点处不能在光源上，因为如果在光源上，就相当于重复计算了 $P(reflect\_1\_times)$。设我们采样到的方向是上图中 p1 到 p0 的方向。

从 p1 入射到 p0 方向的光分为两部分，第一部分是直接光照射到 p1 然后再反射的部分（$P(reflect\_2\_times)$),第二部分是光在其他表面上反射到 p1 然后再反射的部分（$P(reflect\_3\_times)$），我们暂时因为只考虑 $P(reflect\_2\_times)$，所以就不需要管（$P(reflect\_3\_times)$）了。

此时，我们计算出的人眼接受到的 radiance 描述为：

$$radiance = P(reflect\_1\_times) + P(reflect\_2\_times) \tag{2.2.3}$$

如何计算多次反射

除了计算直接光照，当我们想要计算 $P(reflect\_2\_times)$ 时，我们就需要采样入射光线 $p1 \longrightarrow p0$ 了，此时我们需要进行重要性采样，对于采样到的方向，我们需要知道其 Pdf，以及从该方向反射到人眼的 BSDF。下面的代码中，一开始时，beta=1，当光要从 p1 入射到 p0 时：

```
Spectrum f = isect.bsdf->Sample_f(wo, &wi, sampler.Get2D(), &pdf,
    BSDF_ALL, &flags);
if (f.IsBlack() || pdf == 0.f) break;
beta *= f * AbsDot(wi, isect.shading.n) / pdf;
```

也就是说，从 p1 反射到 p0 的光要乘以 beta（这里的 p1 不是一个点，而表示的是 $P(reflect\_2\_times)$ 所有可能的非光源点），就得到了 $P(reflect\_2\_times)$。使用重要性采样是为了计算全部方向（除直接光照以外）所有方向的反射两次的光照的积分。

（讲到这里我已经尽力了，我也实在无法把这个过程讲成跟 1+1+1 这种简单的顺序形式，因为这个过程本身其实是一个递归的过程，beta 值的定义可以把这个过程用循环的方法来计算，但是本质上还是一个递归的过程。）

代码实现

首先我们先回顾一下 PBRT 概率采样中遇到的各种函数的意义。

BxDF::Pdf 函数计算给定方向 wo 和 wi 以后，光源从 wi 入射并反射到 wo 的比例。

```
BxDF::Pdf(const Vector3f &wo, const Vector3f &wi) const;
```

BxDF::Sample_f 函数根据给定的出射方向 wo 来采样一个入射方向 wi，并计算生成它的 Pdf（默认调用上面的 BxDF::Pdf 函数），并调用 f 函数返回 f(wo, *wi)。

```
Spectrum BxDF::Sample_f(const Vector3f &wo, Vector3f *wi, const
    Point2f &u,
float *pdf, BxDFType *sampledType) const;
```

BxDF::f 函数在给定 wo 和 wi 的条件下返回 BSDF 值（光从 wi 方向射入，并射出到 wo 方向以后的值，该值与材质有关）。

```
Spectrum BxDF::f(const Vector3f &wo, const Vector3f &wi) const;
```

BSDF 的这三个同名函数其实就是计算当材质包含多个 BxDF 时怎么组合它们。如果 BSDF 里只包含了一种 BxDF，那其实 BSDF 的这三个函数就是直接使用该 BxDF 的三个函数。

用伪代码形式描述一下 PBRT 中的路径追踪，为了更清晰我们不讨论完美镜面反射：

```
Spectrum L(0.f), beta(1.f);
for (bounces = 0;; ++bounces) {
  （1）计算交点
  （2）if(第0次追踪 并且 有交点) L += beta * isect.Le(-ray.d);
  （3）计算生成BSDF
```

```
    （4）L += beta * 使用MIS采样直接光照
    （5）f = BSDF->Sample_f 生成wi和pdf，返回BSDF
    （6）beta *= f * AbsDot(wi, isect.shading.n) / pdf;
    （7）产生新的Ray
}
return L;
```

2.3 俄罗斯轮盘赌

估计器 F 的效率定义为：

$$\epsilon[F] = \frac{1}{V[F]T[F]} \tag{2.3.1}$$

其中 $V[F]$ 表示的是方差，$T[F]$ 表示的是运行时间。

使用俄罗斯轮盘

俄罗斯轮盘赌可以提高渲染的效率。例如对于直接采样光时，光源 radiance 设为 L_d，使用蒙特卡洛估计渲染方程的公式为：

$$\frac{1}{n}\sum_{i=1}^{n}\frac{f_r(p,\omega_o,\omega_i)L_d(p,\omega_i)|cos\theta_i|}{p(\omega_i)} \tag{2.3.2}$$

求和的每一项的大部分计算费用来自于从 p 点追踪阴影光线，以查看从 p 点所看到的光源是否被遮挡。但是对于 BRDF 项非常小的时候，对最终的估计贡献并不会很大。我们可以使用俄罗斯轮盘赌来解决这个问题。

选择终止概率 q，当某次要渲染的时候，当当前随机数属于 q 内时，积分值在某些时候不计算，而是使用某个常数值替代；当当前随机数属于 1-q 内时，积分值开始计算，但最终结果需要乘以权重 $1/(1-q)$，即：

$$F' = \begin{cases} \frac{F-qc}{1-q} & \zeta > q \\ c & otherwise \end{cases} \tag{2.3.3}$$

使用俄罗斯轮盘赌的期望值与正常求积分值是一样的：

$$E[F'] = (1-q)(\frac{E[F]-qc}{1-q}) + qc = E[F] \tag{2.3.4}$$

终止概率 q 可以用多种方式选择，例如它可以基于所选特定样本计算出的被积函数值，随着被积函数值变小而增加。

俄罗斯轮盘赌从不会减少方差。事实上，除非 c=F，否则它总是会增加方差。然而，如果根据概率来选择，从而跳过可能对最终结果作出微小贡献的样本，则确实可以提高效率。

其中一个陷阱是，选择不当的俄罗斯轮盘赌权重可以大大增加方差。想象一下，将俄罗斯轮盘赌应用到所有的相机光线，终止概率为.99：我们只跟踪 1% 的相机光线，每个光线的权重为 1/0.01=100。从严格的数学意义上讲，得到的图像仍然是“正确”的，尽管从视觉上看结果会很糟糕：大部分是黑色像素，还有一些非常明亮的像素。

路径追踪中的无偏性

假如我们研究路径中的积分估计，我们可以按照 PBRT 的思路进行（见《PBRT 系列 11-代码实战-路径追踪》）：我们设每个路径的计算结果相加，得到一次路径追踪的估计值。但是要注意的是光线反弹的权重（PBRT 源码 path.cpp 的 beta 变量）会不断改变，我们只关心每个路径：

$$P(\overline{p_1}) + P(\overline{p_2}) + P(\overline{p_3}) + \frac{1}{1-q}\sum_{i=4}^{\infty} P(\overline{p_i}) \tag{2.3.5}$$

以这种方式使用俄罗斯轮盘赌并不能解决需要计算无穷和的问题，如果我们把这个想法更进一步，设每一项终止概率 q_i：

$$\frac{1}{1-q_1}\left(P(\overline{p_1}) + \frac{1}{1-q_2}\left(P(\overline{p_2}) + \frac{1}{1-q_3}\left(P(\overline{p_3}) + \cdots\right.\right.\right. \tag{2.3.6}$$

可以看到，我们可以在任意路径中单独使用俄罗斯轮盘，只需要使用后权重除以概率值即可：

```
1 //PBRT源码 path.cpp
2 beta /= 1 - q;
```

3. PBRT 概率方法编程实现

本章从 *PBRT* 代码角度来阐述概率方法，包括一维二维某概率密度函数随机数的生成、采样圆盘等方法。由于本章内容相对简单而且零散，故先留下提纲，等日后有需要再进行补充。

3.1 一维离散概率密度函数

本节内容暂时未设计和规划。

3.2 生成随机方向

本节内容暂时未设计和规划。

3.3 一维离散概率密度

本节内容暂时未设计和规划。

3.4 二维离散概率密度

本节内容暂时未设计和规划。

4. 体渲染采样

光（粒子）通过参与介质时，会受到介质中粒子的影响，介质会吸收光来减弱当前光束的能量，会发射光来增强当前光束的能量，当前光束会与介质中粒子碰撞后散射到各个方向，其他方向也会有散射过来的光增强当前光束的光量。本章的内容就是为了构建体渲染中的采样体系。

4.1 本章概要

体渲染是一个比较大的话题，有多种方法。对于医学影像等科学数据的渲染和烟雾的等人造数据的渲染手段经常是不同的。

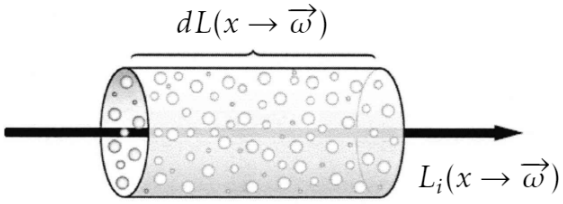
光在沿着直线在体空间前进时，假设体空间充满了微粒，不同位置的微粒密度不同。光在前进时，在每个位置都会有 σ_s 比例的部分被散射，σ_a 比例的部分被吸收，因此沿着这个直线方向，会有 $\sigma_s + \sigma_a = \sigma_t$ 比例的光被衰减，剩下的部分继续沿着当前方向前进。

光在体空间中前进的过程是一个连续的过程，而蒙特卡洛算法就是来估计这个连续的过程的，基于 MC 方法的 Woodcock 追踪（Delta 追踪）技术可以在一条直线上产生合理的样本分布，它可以无偏估计估计光沿着某一段距离的衰减。在体路径追踪过程中，Woodcock 追踪会一次前进一个随机距离，然后估计这段距离对光的衰减值；然后在当前位置用随机方法决定光的散射方向（光从哪个方向反射到当前方向）。

体渲染的概率证明较为复杂，难度也较大，本文不介绍这些内容，但会在文中附加一些论文资料，大家可以去论文中找到相关内容的证明方法。

4.2 介质粒子对穿越光线的作用

我们先介绍一些各种符号的意义。$x \to \vec{\omega}$ 表示射线从 x 发出，方向为 $\vec{\omega}$。$x \leftarrow \vec{\omega}$ 表示射线沿着 $\vec{\omega}$ 方向射向 x。



光线以 $\vec{\omega}$ 方向穿过介质，每 Δt 小步都会与介质交互而被吸收。如果吸收量的系数是 $\sigma_a(x+t\vec{\omega})$，则有 $\sigma_a(x+t\vec{\omega})\Delta t$ 比例的光能会被吸收掉，因此，我们可以推导出微分形式：

$$L((x+\Delta t\cdot\vec{\omega})\to\vec{\omega}) = L(x\to\vec{\omega})(1-\sigma_a(x+\Delta t\cdot\vec{\omega})\Delta t) \tag{4.2.1}$$

$$\frac{L((x+\Delta t\cdot\vec{\omega})\to\vec{\omega}) - L(x\to\vec{\omega})}{\Delta t} = -L(x\to\vec{\omega})\sigma_a(x+\Delta t\cdot\vec{\omega}) \tag{4.2.2}$$

$$\Delta t\to 0 \ : \ dL(x\to\vec{\omega}) = -L(x\to\vec{\omega})\sigma_a(x) \tag{4.2.3}$$

粒子发光就比较简单了，可以直接得到（$L_{ve}(x\to\vec{\omega})$ 表示粒子在当前点发出的朝向 $\vec{\omega}$ 方向的光）：

$$dL(x\to\vec{\omega}) = L_{ve}(x\to\vec{\omega})dt \tag{4.2.4}$$

外散射也相当于衰减辐射度，内散射相当于增强辐射度。外散射的公式可以跟吸收的公式一样推导出来（散射和吸收都会对发射到 $\vec{\omega}$ 方向的光进行衰减）：

$$dL(x\to\vec{\omega}) = -\sigma_s(x)L(x\to\vec{\omega})dt \tag{4.2.5}$$

内散射的公式依赖于入射光 L_i，我们定义相位函数的符号为 p：

$$dL(x\to\vec{\omega}) = \sigma_s(x)L_i(x\to\vec{\omega})dt \tag{4.2.6}$$

$$L_i(x\to\vec{\omega}) = \int_{\Omega_{4\pi}} p(x,\vec{\omega}'\to\vec{\omega})L(x\leftarrow\vec{\omega}')d\vec{\omega}' \tag{4.2.7}$$

$$\int_{\Omega_{4\pi}} p(x,\vec{\omega}'\to\vec{\omega})d\vec{\omega}' = 1 \tag{4.2.8}$$

有人可能会好奇，相位函数已经决定了某个方向的光在当前点散射到另外一个方向的比例，为什么还有一个 σ_s 系数，这是因为散射到 x 点的 $\vec{\omega}$ 方向的光会因为材质本身属性被削弱。相位函数只决定了散射的方向比例，但是在当前点还需要取决于当前点的散射属性（这是我在 [7] 中找到的比较合理的解释，据说在书籍 [8] 中有理论解释，但我找不到这本书的原版。在 1970 年以后的论文中我就没有再找到关于内散射推导的论文了，现有论文都是直接使用上面的结论）。

体渲染方程就是上述几种效应的组合，我们下一节再介绍。

为了方便，人们通常把外散射和吸收系数合并，作为消光系数：

$$\sigma_t = \sigma_a + \sigma_s \tag{4.2.9}$$

我们设经过 x 的直线段 $x+t\vec{\omega}$ 中，$t\in[0,d]$：

$$T_r(x'\leftrightarrow x) = e^{-\int_0^d \sigma_t(x+t\vec{\omega})dt} = e^{-\tau(x'\leftrightarrow x)} \tag{4.2.10}$$

它满足乘法性：

$$T_r(x\leftrightarrow x'') = T_r(x\leftrightarrow x')T_r(x'\leftrightarrow x'') \tag{4.2.11}$$

4.3 体渲染方程

我们把所有的效果合起来，就能得到辐亮度在 x 处沿着方向 $\vec{\omega}$ 总的变化率：

$$dL(x \to \vec{\omega}) = -L(x \to \vec{\omega})\sigma_a(x) - \sigma_s(x)L_i(x \to \vec{\omega}) + L_{ve}(x \to \vec{\omega}) + \sigma_s(x)L_i(x \to \vec{\omega})dt \tag{4.3.1}$$

在两边进行一下积分，来得到纯积分形式。注意这个积分形式是令视线采样的方向为 $\vec{\omega}$，所以体空间内光传输在方向就写为 $-\vec{\omega}$：

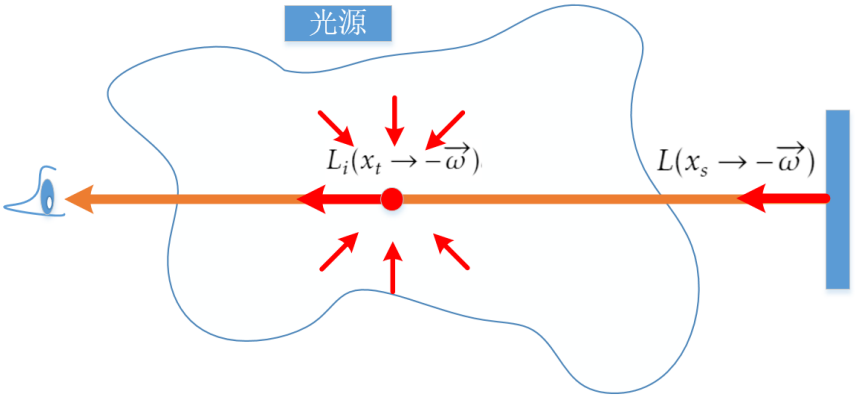
$$x_t = x + t\vec{\omega} \tag{4.3.2}$$

$$\begin{aligned} L(x \leftarrow -\vec{\omega}) =& T_r(x' \leftrightarrow x_s)L(x_s \to -\vec{\omega}) \\ &+ \int_0^s T_r(x \leftrightarrow x_t)L_{ve}(x \to -\vec{\omega})dt \\ &+ \int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t \to -\vec{\omega})dt \end{aligned} \tag{4.3.3}$$

其中：

$$L_i(x_t \to -\vec{\omega}) = \int_{\Omega_{4\pi}} p(x_t, \vec{\omega}' \to -\vec{\omega})L(x_t \leftarrow \vec{\omega}')d\vec{\omega}' \tag{4.3.4}$$

上式中，x_s 表示 $x + s\vec{\omega}$，即光穿过介质以后打在物体表面上时得到的亮度。上式右边的第一项表示采样到体空间外的亮度，乘以积累的光衰减率得到的辐亮度；第二项表示累积的介质自发光亮度；第三项表示累积的内散射辐亮度。



4.4 最简单的采样思想-光线步进

我们忽略发光项，令视线采样的方向为 $\vec{\omega}$，得到体渲染方程为：

$$L(x \leftarrow -\vec{\omega}) = \int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t \to -\vec{\omega})dt + T_r(x \leftrightarrow x_s)L(x_s \to -\vec{\omega}) \tag{4.4.1}$$

该方程是沿着视线直线前进的过程，左边部分是积累的内散射辐射度，右边部分是该直线上从边界射入体空间的光经过衰减以后到达人眼的辐射度。

其实我们也可以理解为体空间中某个点 x 发射到 ω 方向的辐射度，这是很多资料上都如此进行的定义（相当于另一种思考方法，就是表示从该点 x 发射到指定方向 ω 的光等于 x 自身发

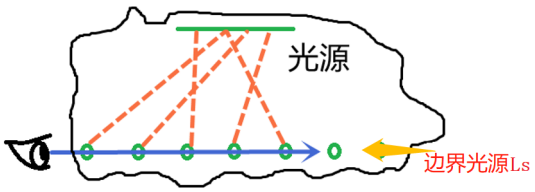
射项加上其他各个方向射到点 x 然后散射到 ω 方向的光之和）：

$$L(\overrightarrow{\omega} \leftarrow x) = L_{ve}(\overrightarrow{\omega} \leftarrow x) + \sigma_s(\overrightarrow{\omega} \leftarrow x) \int_{\Omega_{4\pi}} p(x, \overrightarrow{\omega}' \rightarrow \overrightarrow{\omega}) L(x \leftarrow \overrightarrow{\omega}') d\overrightarrow{\omega}' \tag{4.4.2}$$

该方程可以使用黎曼和的方法求解，在 Ray 采样中，这种离散的方法称为 Ray Marching，即光线步进。为了减少 artifacts，我们一开始步进时选择一个随机的步进长度，之后每次步进在前一个采样点处前进一个固定步长（长度为自定义值）。

4.4.1 单散射

如下图，采样过程中，假设光只会散射一次，因此我们从光源随机采样，然后计算采样点能否照射到当前步进的位置，提供给内散射增强当前光路。



我们给出这个过程的伪代码描述，但是先解释一下符号。tr 表示积累的穿透率（为 0 表示完全不透明），$\mathbf{x}$ 表示起始点，d_{max} 为当前点沿当前方向穿过介质的长度，$\mathbf{d}_{ray}$ 表示步进方向，l 是采样到光源点的光辐射度；$e^{-\tau(p_1,p_2)}$ 函数表示 p_1,p_2 两点之间的衰减量；$\sigma_s(p)$ 函数表示 p 点的散射系数，visible() 函数表示两点间的可见性。Ls 表示采样 Ray 离开体空间后最近相交的平面的发出（或反射）的亮度（在采样光线从体空间的射出点，体空间外部光源沿着 $-\mathbf{d}_{ray}$ 方向射入体空间内的光）。

注意，采样光源其实是需要用蒙特卡洛采样光来估计的，为了避免伪代码过于复杂，所以下面并没有写进去。

Algorithm 1: Ray March

$march = stepSize \cdot random(0,1)$;
$Li = (0.0, 0.0, 0.0)$;
$tr = (1.f, 1.f, 1.f)$;
Ls = illumination at $\mathbf{x} + d_{max} \cdot \mathbf{d}_{ray}$;
$\mathbf{p}_{old} = \mathbf{x} + \mathbf{d}_{ray} \cdot march$;
while $march < d_{max}$ **do**
    $\mathbf{p}_{cur} = \mathbf{p}_{old} + \mathbf{d}_{ray} \cdot march$;
    $tr = tr \cdot e^{-\tau(\mathbf{p}_{cur}, \mathbf{p}_{old})}$;
    Sample $LightPos$;
    if *visible(lightPos,* $\mathbf{p}_{cur}$*)* **then**
        l = light illumination at $lightPos$;
        根据光线步进计算 l 到达 $\mathbf{p}_{cur}$ 后的衰减 ;
        $Li = Li + tr \cdot l \cdot \sigma_s(\mathbf{p}_{cur})$;
    end
    $march = march + stepSize$;
end
$Li = Li \cdot stepSize$;
return $Li + tr \cdot Ls$;

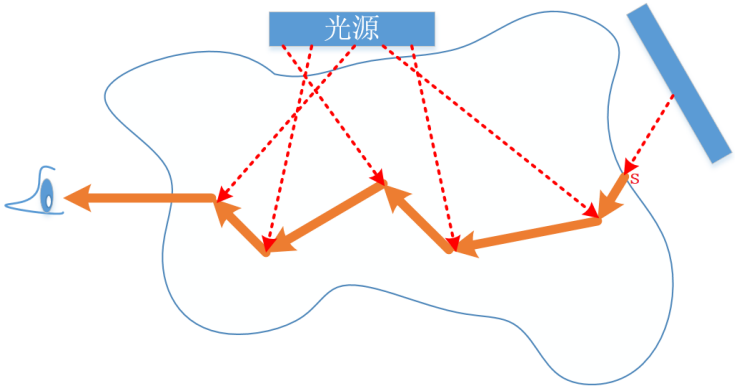
根据光线步进计算 l 到达 $\mathbf{p}_{cur}$ 后的衰减，如果是均匀介质，那么相当于直接计算：

$$l \Longleftarrow l \cdot e^{-\tau(lightPos, \mathbf{p}_{cur})} \tag{4.4.3}$$

如果是非均匀介质，那么就通过 RayMarching 方法求光衰减。

4.4.2 多重散射

基于光线步进的多散射过程描述如下，其中，s 表示采样光线不断前进和散射后最先离开体空间的位置。



注意对于多散射事件中，我们的估计方法如果发生了吸收事件，意味着采样光线的终止。这个方法的理念就是通过 $0-1$ 分布来得到真实值（见 [4] 中对 Delta 追踪的解释。虽然光线步进并不是 Delta 追踪，但计算多散射时，采样光线前进的思路也是一样的）。

在下面的伪代码中，$Tr()$ 函数就是求光到达当前散射位置时衰减后的值。hasEvent 表示是否发生了散射事件。由于散射方向会经常改变，因此 d_{max} 的值也会不断调整，表示当前点沿着当前采样方向射出体空间外的最短距离（在体空间内前进的最大距离）。$\mathbf{s} = traceRay()$ 函数用来计算当前点沿着当前方向前进后，射出体空间的位置 $\mathbf{s}$。

同单散射一样，为了避免伪代码过于复杂，用蒙特卡洛采样光源的计算也没有写进去。

在每轮计算中，p_{pass} 概率为从 $\mathbf{p}_{old}$ 传输 $stepSize$ 距离能够通过的概率，这个概率要根据 tr 来进行计算，但这种计算并不是特别简单——因为 tr 对于彩色光来说，是 RGB 三通道的，因此我们需要单独对任意通道来采样，我们放在后面的蒙特卡洛方法来讲解。

伪代码总结自 [3]，这里的代码中，有一点需要注意，就是当没有发生吸收，但是发生了散射事件时，需要计算 $tr = tr \cdot e^{-\tau(\mathbf{p}_{cur}, \mathbf{p}_{old})}$，而直接穿透则不用（吸收事件直接暂停退出循环）。我认为这里不需要更新 tr，原因解释如下：当采样到某位置时，有三种可能发生的事件：散射、吸收、穿透。如果比例分别是 0.2,0.3,0.5，由于我们每次只跟踪一条光线，根据概率，跟踪穿透的几率是 0.5，而剩下 0.5 则是无法直接穿过，受到了粒子的影响，0.3 比例的光被吸收后终止，0.2 比例的光被衰减以后继续前进。所以根据概率跟踪，这 0.2 比例的光没有被吸收衰减，而是无损地进行了反射。出于谨慎，我也没有在伪代码中修改过来，因为 woodcock 追踪也是十几年前就被用到图形学中的产物了，Ray Marching 的很多资料也都没有了，读者如果感兴趣可以自己再斟酌斟酌。

Algorithm 2: Ray Marching

$Li = (0.0, 0.0, 0.0)$;
$tr = (1.f, 1.f, 1.f)$;
$\mathbf{p}_{old} = \mathbf{x} + random(0,1) \cdot stepSize$;
while *True* **do**
    hasEvent = False;
    p_{pass} = 从 $\mathbf{p}_{old}$ 传输 $stepSize$ 距离能够通过的概率;
    if $random(0,1) > p_{pass}$ **then**
        # 没有直接穿过当前点，要么被散射，要么被吸收 ;
        if $random(0,1) < \frac{\sigma_s(\mathbf{p}_{old})}{\sigma_t(\mathbf{p}_{old})}$ **then**
            hasEvent = true;
            $\overrightarrow{\omega} = samplePhaseFunction(\mathbf{p}_{old}, \overrightarrow{\omega})$;
        end
        else
            光被吸收，退出循环;
        end
    end
    # 没有吸收事件发生（可能有散射事件），则计算最大前进距离 ;
    $\mathbf{s} = traceRay(\mathbf{p}_{old}, \overrightarrow{\omega})$;
    $d_{max} = length(\mathbf{p}_{old}, \mathbf{s})$;
    if $stepSize > d_{max}$ **then**
        Ls = 射出介质点 $\mathbf{s}$ 到当前点 $\mathbf{p}_{old}$ 衰减以后的辐射度;
        break;
    end
    else
        $\mathbf{p}_{cur} = \mathbf{p}_{old} + \mathbf{d}_{ray} \cdot stepSize$;
        Sample $LightPos$;
        if $visible(lightPos,\ p_{cur})$ **then**
            l = light illumination at $lightPos$;
            $l = l \cdot Tr(lightPos, \mathbf{p}_{cur})$;
        end
        if *hasEvent* **then**
            $tr = tr \cdot e^{-\tau(\mathbf{p}_{cur}, \mathbf{p}_{old})}$;
        end
        $Li = Li + tr \cdot l \cdot \sigma_s(\mathbf{p}_{cur})$;
    end
    $\mathbf{p}_{old} = \mathbf{p}_{cur}$;
end
return $Li + tr \cdot Ls$;

4.5 均匀介质无偏方法——基于蒙特卡洛方法

现在我们开始介绍无偏的体空间采样算法。在这之前我们需要特别声明一下，对于成像表面来说，每个像素是一个格子，我们其实是需要积分整个格子的响应，但我们经常会简化这一点——我们把沿着光线追踪初始光线方向的反方向射入的光取平均值，来作为当前一个像素的 radiance 值。

光沿着直线在体空间中照射到某个散射点以后的剩余量，就是光初始能量乘以这段距离的光穿透率。在均匀介质中，穿透率的计算很简单，见下面 PBRT 的代码：

```
1 Spectrum HomogeneousMedium::Tr(const Ray &ray, Sampler &sampler)const{
2   return Exp(-sigma_t * std::min(ray.tMax*ray.d.Length(), MaxFloat));
3 }
```

而在体渲染积分中，最难计算的过程是内散射项的积分。虽然光可能会散射向各个方向，但我们最终需要的只是沿着 $-\omega$ 方向射入到 x 点处的光 $L(x \leftarrow -\vec{\omega})$。沿着 $-\omega$ 方向积累的内散射辐射度为：

$$\int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t \rightarrow -\vec{\omega})dt = \int_0^s e^{-t\sigma_t}\sigma_s(x_t)L_i(x_t \rightarrow -\vec{\omega})dt \tag{4.5.1}$$

$$L_i(x_t \rightarrow -\vec{\omega}) = \int_{\Omega_{4\pi}} p(x_t, \vec{\omega}' \rightarrow -\vec{\omega})L(x_t \leftarrow \vec{\omega}')d\vec{\omega}' \tag{4.5.2}$$

均匀介质中，$T_r(x \leftrightarrow x_t)$ 可以表示为 $e^{-t\sigma_t}$。上面的的计算可以采用蒙特卡洛方法：

$$\frac{1}{N}\sum_{j=1}^{N}\frac{e^{-t\sigma_t}\sigma_s(x_t)L_i(x_t \rightarrow -\vec{\omega})}{pdf(t)} \tag{4.5.3}$$

该方法其实主要还是考虑的是单散射，也就是光源发出的光散射一次以后，映射到人眼里（因为没有对 $\vec{\omega}$ 进行采样）。不过我们可以扩展为多散射：每次采样一个长度后，如果没有出介质，就根据相位函数计算散射方向（在后面 HomogeneousMedium::Sample 函数中有介绍）。

为了采样更符合积分项，我们让分布符合指数分布：

$$pdf(t) = ce^{-t\sigma_t} \tag{4.5.4}$$

$$\int_0^\infty ce^{-t\sigma_t} = 1 \tag{4.5.5}$$

$$c = \sigma_t \tag{4.5.6}$$

$$P(t) = 1 - e^{-t\sigma_t} \tag{4.5.7}$$

$$P^{-1}(t) = -\frac{ln(1-t)}{\sigma_t} \tag{4.5.8}$$

对于黑白介质（即 σ_t 和 σ_s 的三通道都是相同值，后面还会再进行描述）可以得到：

$$\frac{T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t \rightarrow -\vec{\omega})}{pdf(t)} = \frac{\sigma_s}{\sigma_t}L_i(x_t \rightarrow -\vec{\omega}) \tag{4.5.9}$$

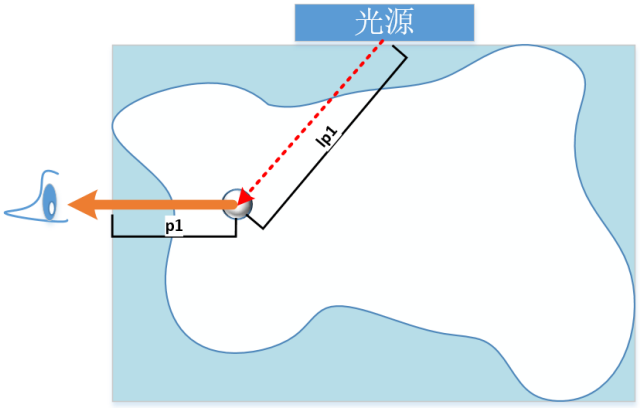
也就是说，对于黑白介质，我们可以用 $\frac{\sigma_s}{\sigma_t}$ 来估计两点之间的穿透率，用 $P^{-1}(random(0,1))$ 来随机产生前进距离。当我们在计算多重散射时，得到的多个散射点的 $\frac{\sigma_s}{\sigma_t}$ 相乘，就是多次散射以后得到的穿透率。

这是怎么扩展到多散射的呢？其实我们可以这么来思考。我们相求下面这段积分，但我们并不知道 s 多长，我们可以用采样的方法采样一个距离，然后用蒙特卡洛方法求解积分。

$$\int_0^s e^{-t\sigma_t}\sigma_s(x_t)L_i(x_t \to -\vec{\omega})dt \tag{4.5.10}$$

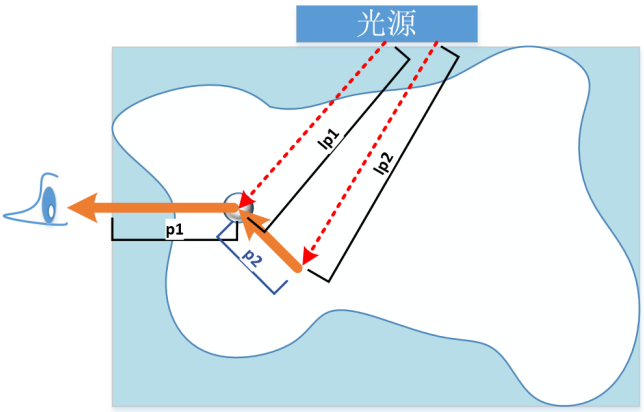
从当前点 $\mathbf{x}$ 沿着某个方向 $\vec{\omega}$ 来说，可能有的光是从 $\mathbf{x}+0.02\vec{\omega}$ 处散射到 $-\vec{\omega}$ 方向的；有的光是从 $\mathbf{x}+0.04\vec{\omega}$ 处散射到 $-\vec{\omega}$ 方向的；我们只能跟踪一个距离，因此就会根据概率来采样这个跟踪的距离 a，然后再计算照射到 $\mathbf{x}+a\vec{\omega}$ 处的光散射到 $-\vec{\omega}$ 方向的部分，然后除以采样距离 a 的概率密度。

假如第一次采样以后，得到了下面的采样点：



这时，相当于估计出的 p1 穿透率为 $\frac{\sigma_s}{\sigma_t}$，光源沿着 lp1 穿透到散射点以后再散射到 p1 方向的值为 $L_i(x_t \to -\vec{\omega})$。这其实相当于对一次散射进行估计，然后我们需要估计二次散射。

在估计二次散射时，通过采样相位函数采样到一个入射方向，再采样距离得到下一个散射点，此时从第二个散射点散射到第一个散射点再散射出体空间的总穿透率其实就是 $\frac{\sigma_s}{\sigma_t} \times \frac{\sigma_s}{\sigma_t}$。



换句话说，粒子密度 Density 决定了光与粒子发生作用的概率，$\frac{\sigma_s}{\sigma_t}$ 决定了发生作用以后被散射的概率，即穿透率；$\frac{\sigma_a}{\sigma_t}$ 则表示光子被吸收的概率。

我们现在再回到采样距离的部分，给定 $[0,1]$ 之间的随机数，需要使用上式反函数法得到前进步长 t 值：

$$r = random(0,1) \tag{4.5.11}$$

$$t = -\frac{ln(1-r)}{\sigma_t} \tag{4.5.12}$$

我们已经知道，当采样到某个 t 时，且沿着该方向前进 t 后仍在参与介质内，则不需要像以前那样根据 Tr 的公式计算衰减，而是将 Tr 乘以 $\frac{\sigma_s}{\sigma_t}$。

在下面代码中，当采样光线已经射出体空间后，即 $t \geq ray.tMax$，则在我们的估计方法中，相当于没有被阻挡，也就是不需要再乘以 *sigma_s* 了。

PBRT[1] 的 HomogeneousMedium::Sample 函数就是该方法。

4.5.1 黑白介质

一开始为了简单，我们假设参与介质是黑白的（非彩色介质）：

```
1  //非彩色介质，sigma_t[0]=sigma_t[1]=sigma_t[2]
2  Float dist = -std::log(1 - sampler.Get1D()) / sigma_t[0];
3  //计算t并计算衰减Tr
4  Float t = std::min(dist / ray.d.Length(), ray.tMax);
5  Spectrum Tr = Exp(-sigma_t*std::min(t, MaxFloat)*ray.d.Length());
6  //我们把PBRT中下面的过程进行修改，变得更通俗一些：
7  if(t < ray.tMax){
8    //没有散射出介质外
9    density = sigma_t * Tr
10   // 对于黑白介质，直接用density[0]表示密度
11   pdf = density[0]
12   // 黑白介质返回值其实就是 sigma_s/sigma_t
13   return Tr * sigma_s[0] / pdf
14 }else{
15   //散射出介质外
16   density = Tr
17   pdf = density[0]
18   // 对于黑白介质，返回值就是 1
19   return Tr / pdf
20 }
```

以及 volPath 积分器的相关逻辑：

```
1  Spectrum L(0.f), beta(1.f);
2  for (bounces = 0;; ++bounces){
3    MediumInteraction mi;
4    if (ray.medium) beta *= ray.medium->Sample(ray, sampler, &mi);
5    if (beta.IsBlack()) break;
6    if (mi.IsValid()) {
7      if (bounces >= maxDepth) break;
8      const Distribution1D *lightDistrib = lightDistribution->Lookup(mi.
         p);
```

```
9       L += beta * UniformSampleOneLight(mi, scene, sampler, true,
            lightDistrib);
10      Vector3f wo = -ray.d, wi;
11      mi.phase->Sample_p(wo, &wi, sampler.Get2D());
12      ray = mi.SpawnRay(wi);
13      specularBounce = false;
14    }else{
15      //不在参与介质内，计算表面模型Path Tracing的光照
16      ......
17    }
18    //使用俄罗斯轮盘赌来终止光线
19    ......
20  }
```

4.5.2 彩色介质

彩色介质要随机选择一个 channel 来采样。为了简单，我们设 Spectrum 是 RGBSpectrum。采样 t 与相应的概率密度：

$$\begin{aligned} r &= random(0,1) \\ t &= -\frac{ln(1-r)}{\sigma_t} \\ p_t(t) &= \sigma_t e^{-\sigma_t t} \end{aligned} \tag{4.5.13}$$

考虑 σ_t 有三通道，分别为 $\sigma_t^{(1)}$、$\sigma_t^{(2)}$ 和 $\sigma_t^{(3)}$，则采样 $p_t(t)$ 应该实际写为：

$$p_t(t) = \frac{1}{3}\sum_{i=1}^{3}\sigma_t^{(i)}e^{-\sigma_t^{(i)}t} = \frac{1}{3}\sum_{i=1}^{3}p_t^{(i)}(t) \tag{4.5.14}$$

关于上面的概率密度，我做一点解释。首先，我们如何采样到 t 是个问题，采样到 t 以后如何计算概率密度又是个问题。如果是黑白介质，那就简单了，直接根据 σ_t 的第一个通道来采样即可；如果是彩色介质，假如我们根据 σ_t 的通道 1 采样到了距离 t，这个 t 对应的概率密度应该是 $p_t^{(1)}(t)$ 吗？不应该是，因为我们不得不假设这个 t 值是三个通道一起采样得到的，而且恰好这三个通道值采样得到的结果都是一样的（都是 t）。读者可能会问，我明明是根据 $p_t^{(1)}(t)$ 分布采样到的 t，概率密度却计算为三种分布的平均值，这样会不会不对？并不会，因为我们是随机选取三通道中的一个，而不是每次都按照通道 1 来采样，记住这一点是理解这个概率密度计算的关键。

在 $t = tMax$ 处采样到表面（体空间内的某物体的表面或者是穿出体空间的表面），与在 $t = 0$ 到 $t = tMax$ 之间有散射点是互补事件，因此：

$$p_{surf} = 1 - \int_0^{tMax} p_t(t)dt = \frac{1}{3}\sum_{i=1}^{3}e^{-\sigma_t^{(i)}tMax} \tag{4.5.15}$$

采样中我们根据下面的公式，把 $pdf(t)$ 带进去来计算即可：

$$\frac{T_r(x \leftrightarrow x_t)\sigma_s(x_t)}{pdf(t)} L_i(x_t \rightarrow -\vec{\omega}) \tag{4.5.16}$$

PBRT 的代码如下：

```
float dist = -std::log(1 - sampler.Get1D()) / sigma_t[channel];
float t = std::min(dist / ray.d.Length(), ray.tMax);
```

如果当前还在介质内部（t<tMax）：

```
Spectrum density = sigma_t * Tr;
float pdf = (density[0] + density[1] + density[2])/3.0;
return Tr * sigma_s / pdf;
```

如果已经离开了介质（t>=tMax）：

```
Spectrum density = Tr;
float pdf = (density[0] + density[1] + density[2])/3.0;
return Tr / pdf;
```

4.6 非均匀介质的多重散射

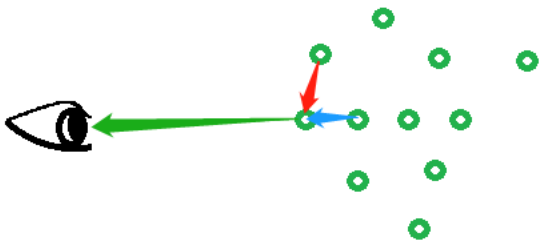
注意 PBRT-V3 只支持黑白的非均匀介质，而比较新的论文（我记得是 TOG2018 年以后）有关于“null-scattering”技术的彩色非均匀介质的计算。

4.6.1 采样散射点

光有可能会在介质中多次散射，才能达到当前光路中。这个过程很像光子映射，我们追踪从光源发出的光子，然后跟踪它在场景中的多次反射。但我们目前还是考虑从相机处开始进行采样。

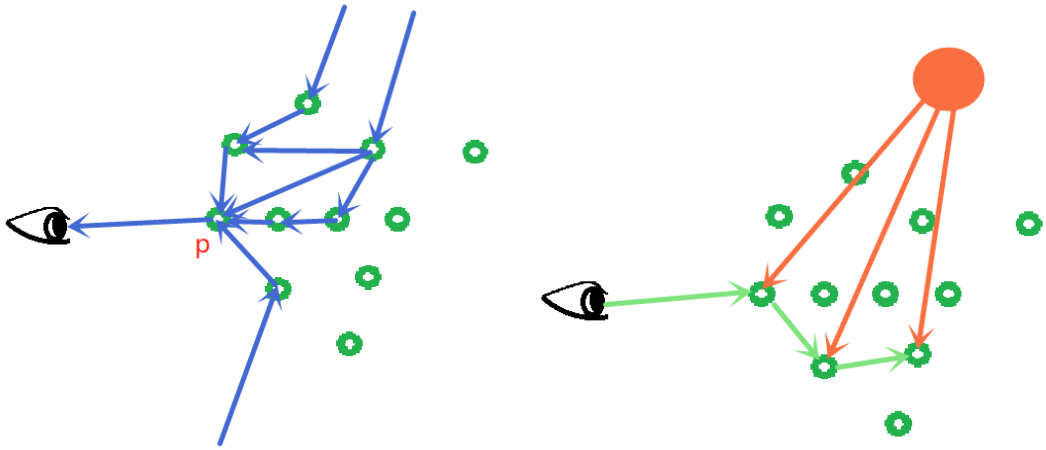
一次追踪所有的散射情况和吸收情况肯定是不可能的，因此我们使用下面的算法进行简化。我们依旧是只跟踪一条采样光线（不过对于单散射模型而言我们只需要跟踪一条光线），我们在每个步进点根据概率来决定它是被吸收还是散射。我们有吸收系数 σ_a 以及散射系数 σ_s，得到总衰减系数为 $\sigma_t = \sigma_a + \sigma_s$，散射概率为 $\frac{\sigma_s}{\sigma_t}$。

我们回想一下射到人眼的光分为两部分：



红色表示从别的方向散射到当前方向然后映射到人眼的部分，蓝色表示本身就在该方向，经过衰减以后映射到人眼的部分。当然，也可以理解为蓝色也是散射以后映射到人眼方向的，只不过散射并没有改变其初始方向。注意我们采样的方向是逆着光的方向的。

如下图左，假如某像素采样 Ray 与体空间第一个交点为 p，那么射入该像素的光都是从 p 射到人眼的。这些光，有的是经过一次散射就到人眼，但也有经过多次散射才散射到人眼中。为了保证合理的速度，我们不能让 Ray 一直在体空间中进行散射，因此 Ray 会以一定的概率终止（被吸收）。



我们只追踪一条光线，因此需要保证的是能量守恒以及无偏的渲染结果。渲染的过程非常类似于路径追踪，每段路径都采样一次光，然后再采样一次散射，如上图右，这就表示，有的光（橙色）是经过一次散射以后到人眼的，有的光散射了两次。散射的方向是根据相位函数的可逆性（散射到某方向的概率仅仅与两个方向之间的夹角有关）来进行采样的。

常用的无偏算法（woodcock 追踪）首先求出非均匀介质中的最大消光系数 σ_{max}，然后像采样均匀介质那样采样“假的”散射点，这些“假的”散射点会为了保证无偏，以 $\frac{\sigma_t(x_t \to -\vec{\omega})}{\sigma_{max}}$ 的概率接受为真散射点，若为真散射点就采样相位函数，否则就沿直线前进。

4.6.2 非均匀介质的 WoodCock 跟踪

Woodcock 追踪（伪散射、空穴跟踪或零碰撞算法）基于接受-拒绝技术，用于生成具有任意分布的数据。这项技术是在中子输运（Woodcock）和等离子体物理学（Skullerud）中独立开发的，分别用于在具有许多材料的环境中对中子和离子自由路径进行无偏采样。后来 Coleman 将其正式进行了定义。2013 年 Galtier 将其以整体形式提出。Raab 等人 [10] 将 Delta 追踪引入到图形学中，用于渲染参与介质。

woodcock 追踪其实本质上可以分成两个作用，第一个是采样散射点，第二个是估计两点间穿透率。

采样散射点

WoodCock 跟踪（Delta 跟踪）的思想是通过添加虚拟粒子使非均匀介质“均匀化”。设定虚拟粒子的局部浓度，使组合成消光系数 $\overline{\sigma}$，我们做一下符号定义（为了方便起见就使用 PBRT[1]

书中的符号 σ，在论文 [4] 中使用的符号是 μ）：σ_a 是吸收系数，σ_s 是发射系数，$\sigma = \sigma_a + \sigma_s$ 是衰减系数，$\overline{\sigma}$ 是 majorant 衰减系数（构成均匀介质）。

虚拟粒子的相位函数和 albedo 分别设置为 $\delta(\omega)$ 和 1，因此，光子与假想粒子的相互作用沿着原始方向继续不变。这些相互作用在一些物理文献中被称为“零碰撞”。

WoodCock 跟踪（Delta 跟踪）模型与真实和虚拟粒子的相互作用是表现为概率关系的，它会采样一个随机步长（使用上一章的方法），这里的 $\overline{\sigma}$ 一般都会选择介质中最大的密度值。

使用 woodcock 追踪采样下一个散射点距离的函数表示如下：

```
t = 0;
do
    η = random(0,1);
    t = t - log(1−η)/σ̄;
    if (t ≥ d) then
        break;
    end
    ξ = random(0,1);
while ξ > σ(o+t*ω)/σ̄;
return t;
```

因此，当 $\xi > \frac{\sigma(o+t*\omega)}{\overline{\sigma}}$，表示光线没有与虚拟粒子碰撞，因此不改变方向，继续前进；否则就说明与真实粒子发生了碰撞。如果是估计穿透率，则表示此次采样没有穿透介质，返回 0；如果是采样散射距离，就让采样 Ray 前进 t 个单位，然后采样相位函数。

它的无偏性证明来自于 [5] 这篇比较老的论文，大家有兴趣可以看看，不过其实从直觉上来看，它也应该是无偏的。

PBRT[1] 的 Sample 函数首先将光线转换到介质坐标系并单位化光线方向，然后计算 Ray 与介质包围盒的交点，用来为后面判断是否出介质边界。之后设最大衰减值为 $\sigma_{t,max}$，该值是由介质的最大密度和 σ_t 相乘得到的。注意比较是否击中虚拟粒子时的概率为：

$$\frac{Density(p) * \sigma_t}{MaxDensity * \sigma_t} = \frac{Density(p)}{MaxDensity} \tag{4.6.1}$$

GridDensityMedium 的方法将当前点的体素值 (Density) 用来计算散射事件，估计采用的提前定义好的 σ_t 和 σ_s 常量。当前介质的 Density 与最大密度 MaxDensity 用来计算散射的概率，如果我们定义的 σ_s 值比较大，则表明它是散射型介质，否则表示它是吸收型介质。

同样为了方便,我们假设参与介质是黑白的（非彩色介质）,给出 GridDensityMedium::Sample() 通俗的代码描述：

```
float tMin, tMax;
//计算规范化坐标下的射入射出点
......
float t = tMin;
while (true) {
  t -= std::log(1 - sampler.Get1D()) * invMaxDensity / sigma_t;
```

```
7    if (t >= tMax) break; //射出体空间
8    if (Density(ray(t)) * invMaxDensity > sampler.Get1D()) {
9      // 表示与粒子发生了相互作用，计算生成相位函数。
10     ......
11     return sigma_s / sigma_t;
12   }
13 }
```

估计穿透率

在参与介质中，尤其是非均匀参与介质，计算两点之间的透射率（可见性分数）是一个挑战。这个透射率可以计算照明穿过介质被衰减得到的阴影以及介质另一边表面的可见性。当光学厚度 T_r 具有已知的闭式反导数时（比如均匀材质），可以解析地计算透射率。

当材质体素值不均匀时，可以用分段黎曼和的方法求积分值，这就是传统的 Ray March 技术。但是在估计两点之间的穿透率时会导致不可预测的系统偏差，并且需要在高分辨率数据中执行许多精细步骤。

在估计穿透率时，Woodcock 追踪本质上是一个二值函数，用来估计非均匀介质之间的穿透率。我们可以计算粒子在穿过介质时的穿透长度：

```
1  //采样距离的算法
2  d=0;
3  while true{
4      d += -ln(1-random(0,1))/sigma_max
5      if(d>d_max || random(0,1)<sigma_t(p_cur)/sigma_max)
6          break; //射出体空间（停止）或者发生散射（采样相位函数）
7  }
8  return d;
```

如果 $d > d\_max$，则说明穿过了介质，记为 1；否则就没有穿过介质，记为 0。通过大量的穿透估计，得到一堆 1 和一堆 0，然后求平均，就是估计出的穿透率（比如如果求平均以后是 0.3，表示光通过这段介质，沿着直线前进时会有百分之 70 的光被吸收或者散射到其他方向，只有百分之 30 的光会穿过这段介质），穿透率也可以简单认为是透明度，只不过介质中还包括散射，而不单纯只有衰减。

4.6.3 更有效的穿透率计算方法

在高度复杂的非均匀介质中高效的、无偏的透射率估计很重要，但 woodcock 追踪方法在其他条件相同的场景中，渲染不均匀介质通常比渲染均匀介质的时间高几个数量级。它们会产生具有高方差的粗二元估计量，会导致渲染时间大幅增加。

事实上，仅使用可用于非均匀介质的技术（例如 woodcock 追踪跟踪技术，即不允许解析解形式的透射率计算）简单地渲染均匀介质会立即导致显著更高的渲染时间、更高的偏差或更高的

方差—即使场景是 homogeneous。虽然这很不友好，但也表明问题并非介质本身的异质性所固有的，而是源于缺乏在这种更普遍的情况下准确有效地计算透射率适当的工具。在论文 [4] 中提高了效率（使用比率残差追踪法），大家有兴趣可以学习一下。

在论文 [4] 中，发现可以使用俄罗斯轮盘赌来解释上面的二值函数，然而也可以去掉俄罗斯轮盘赌逻辑，简单地用连续概率乘以透射率，而不是在每次迭代中随机地以零的值终止算法。所得的估计量具有相同的平均值和相当低的方差，PBRT3 中，追踪非均匀介质的 Tr() 的实现中就是使用这种方法，在论文 [4] 中该方法被称为比率跟踪（Ratio tracking）：

```
Float Tr = 1, t = tMin;
while (true) {
    ++nTrSteps;
    t -= std::log(1 - sampler.Get1D()) * invMaxDensity / sigma_t;
    if (t >= tMax) break;
    Float density = Density(ray(t));
    Tr *= 1 - std::max((Float)0, density * invMaxDensity);
    //当Tr()很小的时候就使用俄罗斯轮盘赌（下面的代码是PBRT-V3的书发布
       之后才加上的）
    const Float rrThreshold = .1;
    if (Tr < rrThreshold) {
        Float q = std::max((Float).05, 1 - Tr);
        if (sampler.Get1D() < q) return 0;
        Tr /= 1 - q;
    }
}
return Spectrum(Tr);
```

只要能理解 woodcock 追踪，ratio 追踪以及 ratio-residual 追踪理解起来也会非常简单。

参考文献

[1] Pharr M, Jakob W, Humphreys G. Physically based rendering: From theory to implementation[M]. Morgan Kaufmann, 2016.

[2] Jensen H W . Monte Carlo Ray Tracing. 2003.

[3] 基于 Woodcock 跟踪的高效散射介质绘制算法的研究与实现. 浙江大学.CAD 实验室

[4] Novak, J., A. Selle, and W. Jarosz. 2014. Residual ratio tracking for estimating attenuation in participating media. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2014) 33(6), 179:1–179:11.

[5] Coleman W A . Mathematical Verification of a Certain Monte Carlo Sampling Technique and Applications of the Technique to Radiation Transport Problems[J]. Journal of Clinical Neurophysiology Official Publication of the American Electroencephalographic Society, 1968, 9(8):Q08011.

[6] Veach, E.: Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University (1997)

[7] Preim B, Botha C P. Visual computing for medicine: theory, algorithms, and applications[M]. Newnes, 2013.

[8] CHANDRASEKHAR S.: Radiative transfer. Dover Publications, 1960. 3

[9] https://blog.csdn.net/pizi0475/article/details/ [给了一个关于 BRDF 很重要的提示]

[10] Raab, Matthias et al. “Unbiased Global Illumination with Participating Media.” (2008).