Metropolis 光传输

Dezeming Family

2022 年 12 月 8 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

本文是对 [1] 论文的详细解读，该论文可以说是渲染必读论文，但有些符号表示和描述可能对初学者并不友好，由于里面介绍的几种重要技术，例如双向路径追踪、多重重要性采样以及 Metropolis 方法都是非常重要的，因此我打算写一下本论文的解读，作为构建“高端图形渲染学习体系”的一个重要组成部分。

由于论文 [1] 篇幅过长，为了减少 Latex 编译的时间以及更好把控不同部分的内容，我将整个论文划分为了多本小册子来进行讲解。

本文的预备知识：蒙特卡洛方法、蒙特卡洛光线追踪（可以看 Peter Shirley 的光线追踪三本小书）、BSDF 模型、路径追踪、向量空间。

目录

一　光传输建模

要想在光传输中应用 Metropolis 算法，需要对光传输进行建模，同时还需要避免启动偏差 (start-up bias)。我们还需要设计一系列的变异策略，使得 Metropolis 算法更有效。本节我们对光传输问题建模，展示应用 Metropolis 方法来估计图像渲染结果。

1 1　光传输问题的建模

我们先不考虑图像色彩的问题，而是假设图像像素只是有亮暗之分的灰度图。

像素值 I_j 可以表示为：

$$I_j = \int_\Omega f_j(\overline{x}) d\mu(\overline{x}) \tag{一.1}$$

f_j 是测量贡献函数，$f_j(\overline{x}) = h_j(\overline{x}) f(\overline{x})$，即滤波器权重与路径接收到的辐射度的乘积（如果路径射入相机时不在某像素为中心的滤波器值大于 0 范围内，则相当于这条路径对该像素没有任何贡献）。对于一条路径 $\overline{x} = x_1, ..., x_k$，$h_j$ 仅仅取决于路径最后一条边 $x_{k-1}x_k$（射入镜头的边）。

我们再严格做一些定义，设 $\int_D f(\overline{x}) d\mu(\overline{x})$ 表示图像平面能够接受到的全部亮度（D 表示对像素值有贡献的路径集，是全部路径的子集）。后面我们就以 Ω 来作为对图像有贡献的路径集（[1] 就是用的 Ω，我也不清楚为什么不用 D，可能是作者写着写着就写忘了）。

使用蒙特卡洛估计得到的结果是（$\overline{X}_i$ 是根据概率密度 p 抽取的）：

$$I_j = E\Big[\frac{1}{N}\sum_{i=1}^{N} \frac{h_j(\overline{X}_i) f(\overline{X}_i)}{p(\overline{X}_i)}\Big] \tag{一.2}$$

当我们知道辐射度函数积分值 $b = \int_\Omega f(\overline{x}) d\mu(\overline{x})$（所有路径的平均亮度），且能够产生符合概率密度 $p = (1/b)f$ 的样本 $\overline{X}_i$，则对每个像素的估计就是：

$$I_j = E\Big[\frac{1}{N}\sum_{i=1}^{N} b h_j(\overline{X}_i)\Big] \tag{一.3}$$

这是因为 $b \approx \frac{f(\overline{X}_i)}{p(\overline{X}_i)}$，当样本 $\overline{X}_i$ 的分布与 f 相同时，就可以用 b 表示对全部的样本的估计 $b = \frac{f(\overline{X}_i)}{(1/b)f(\overline{X}_i)}$，进而用该式替换掉直接用蒙特卡洛估计的结果。尽管我们并不知道 b，不过 b 可以用蒙特卡洛方法来估计得到（先用上一大堆样本去估计 b）。所以可以看出，当 b 估计有一定偏差时，最终结果只会在亮暗上有点区别，整体效果不会有太大变化（这是最简单的使用方法，但是会有启动偏差，我们后面会介绍启动偏差，要想消除启动偏差还得需要从初始路径开始的整条马尔科夫链，而不能将估计 b 和后面的生成符合 f 的样本这两个过程分离开）。

综上，当我们估计得到一系列路径时，在图像上只需要计算它对图像的滤波器响应，就能估计最终结果。当然在生成 $\overline{X}_i$ 时还需要在路径空间中计算 f 值以进行路径变异，只是计算的 f 值不再用于最终的图像。

但问题是只有在趋于均衡时，产生的样本 $\overline{X}_i$ 才会符合我们可以用 Metropolis 算法生成符合概率密度 $p = (1/b)f$ 的样本集，一般是先选择一个初始状态 $\overline{X}_0$，然后开始启动 Metropolis 迭代，并丢弃最前面的 k 个随机游走的状态。但我们通常不清楚在当前的问题下 k 应该取多少更合适，如果太小，就会对结果有很大的影响（称为启动偏差）。

二　消除启动偏差

2 1　如何消除启动偏差

我们让初始样本 $\overline{X}_0$ 从一个更方便的概率密度 p_0 中产生（使用双向路径追踪方法得到）。为了补偿 p_0 不是理想的均衡分布 $p^* = (1/b)f$，样本 $\overline{X}_0$ 分配一个权重：$W_0 = f(\overline{X}_0)/p_0(\overline{X}_0)$。因此使用这一个样本对于像素 j 的估计是 $W_0 h_j(\overline{X}_0)$（其实就是 $\frac{f(\overline{X}_0)h_j(\overline{X}_0)}{p_0(\overline{X}_0)}$）。

之后，其他样本都根据 $\overline{X}_0$ 变异得到（使用 f 作为目标密度）。每个 $\overline{X}_i$ 都有不同的概率密度 p_i，这些概率密度当 $i \to \infty$ 时才趋向于平稳分布 $p^* = (1/b)f$。为了避免偏差，为这些样本分配与原始样本相同的权重 $W_i = W_0$ 就可以了，并且对像素 j 使用以下估计：

$$I_j = E\Big[\frac{1}{N}\sum_{i=1}^{N} W_i h_j(\overline{X}_i)\Big] \tag{二.1}$$

该方法是对 I_j 的无偏估计。[1] 的附录中有证明；正文也给出了直觉上的解释，我们放在下一小节介绍。

2 2　直觉上理解消除启动偏差的原理

初始的 $\overline{X}_0$ 是一个随机变量，它的期望值就是所有可能的 $\overline{X}_0$ 的平均值。

因此，考虑一大组由多次采样 p_0 得到的初始路径 $\overline{X}_{0,j}$（都属于同一个像素），如果 $p_0 = (1/b)f$（p_0 就是平稳分布），那么这些所有的路径的权重都是相同的，这组路径就是均衡的 (equilibrium)：当应用变异策略为每个 $\overline{X}_{0,j}$ 生成路径序列时，这些序列的分布也都是相同的。

假设我们再采样一大组初始路径 $\overline{X}_0$,这次概率密度 p_0 是任意的,我们对每条路径分配权重 $f(\overline{X}_{0,j})/p_0(\overline{X}_{0,j})$。即使这次 $\overline{X}_{0,j}$ 的分布并不是理想均衡分布，但权重的分布却是正比于理想的均衡 f 的（这点我有些疑惑，按理来说权重 $f(\overline{X}_{0,j})/p_0(\overline{X}_{0,j})$ 的分布跟 f 应该没什么关系才对）——当路径发生变异时，均衡得以保持（正如上一种情况一样），这导致了对 I_j 的无偏估计。

这种移除启动偏差的方式并不特定用于光传输，而对光传输来说，要求存在一种采样技术 p_0，这在一些情况中很难得到。

2 3　初始化阶段

实践上来说，初始化 MLT 算法时，如果只是随机只产生一条路径，那么可能由于 $W_0 = 0$ 导致所有后续序列的权重都是 $W_i = W_0 = 0$，从而得到完全黑的图像。初始权重也有可能比所有其他期望值都要大，导致结果过亮。这些并不违背算法的无偏性，只是特定某一次运行时的偏差。

解决方案是并行地运行 n 份算法（随机初始路径各不相同），得到初始样本集 $\overline{X}_{0,1}, \overline{X}_{0,2}, ..., \overline{X}_{0,n}$，并把这些样本生成的随机序列来估计图像得到的贡献都聚集到一张图像上（所有这 n 份的总贡献除以 n）。

我们用两步来实现这个策略，首先采样一大堆 $\overline{X}_{0,1}, \overline{X}_{0,2}, ..., \overline{X}_{0,n}$，然后让 $\overline{W}_{0,1}, \overline{W}_{0,2}, ..., \overline{W}_{0,n}$ 分别作为相应的权重。然后，我们从这些路径中选择一个具有代表性的样本集（n' 个样本，n' 远小于 n），并分配给它们相同的权重。这些都作为 Metropolis 算法的独立的初始种子。

每个代表路径 $\overline{X'}_{0,i}$ 都是从初始路径 $\overline{X}_{0,j}$ 中抽取的，抽取是根据正比于 $W_{0,j}$ 的离散的概率密度，然后这些所有的路径 $\overline{X'}_{0,i}$ 都分配同样的权重：

$$W'_{0,i} = \frac{1}{n}\sum_{j=1}^{n} W_{0,j} \tag{二.2}$$

这种方式也都是无偏的（无偏性的保证在于对权重取平均了）。当然，也可以以相等间隔从 $W_{0,j}$ 的分布中抽取路径，这是为了避免同一个路径被采样到两次（这也是无偏的）。

取 $n' = 1$ 也是合理的，此时，采样 n 个路径的意义就是估计平均值 W_0（$E[W_0] = b$）。

如果图像只需要达到常量的比例因子，则只要找到 $f(\overline{x}) > 0$ 的单个路径就可以终止第一阶段。保留多个种子路径（即 $n' > 1$）的主要原因是进行收敛测试（后面会介绍）或镜头子路径的变异（后面会介绍）。

把渲染求解分为两个子问题的方式更有效，初始化阶段估计整个图像的亮度，Metropolis 估计相对的像素值强度。这两个阶段可以被分开决策，实践上，初始化阶段占总的计算时间是很多的（即使初始化阶段采样 100000 个样本，对于一般尺寸的图像来说甚至一个像素都平均分配不到一个样本）。

2 4　收敛测试

马尔科夫链蒙特卡洛方法依赖于模拟问题的收敛性，需要一定的手段来验证算法是收敛的，否则能否应用它不一定能让人信服。

并行运行多份算法（$\overline{X'}_{0,i}$）的好处是有利于收敛测试 (convergence testing)，我们并不能对单个运行的 Metropolis 算法的样本进行方差测试，因为后续样本都是高度相关的。

为了测试收敛性，Metropolis 阶段会开始 n' 个独立的种子路径，它们对图像的贡献都被分别记录（记录到 n' 张图像），当然只记录图像的一小部分，因为全都保留占用的内存和时间都会过多。对于每幅图像我们都会有 n' 个独立的无偏的样本，当 Metropolis 算法进行时，样本值也会根据路径的变异而变化。这些样本的方差也可以每运行 Metropolis 几轮以后测试一次，直到方差局限在指定的大小内（跟以前其他图形学算法不同，MLT 中每个像素的样本量不变，而只是样本值在变化；其他算法是样本总量在变）。

如果对给定像素有贡献的辐射值可以被预先限定，理论上就可以应用更先进的收敛技术。初始化阶段可以限定像素内样本的方差，而 Metropolis 阶段则可以限定渲染结果的误差。

2 5　光谱采样

到目前为止，我们的讨论仅限于单色图像，但对彩色图像的修改很简单。我们将 BSDF 和光源表示为点采样光谱（尽管使用其他表示方式很容易），给定路径，我们计算每个采样频率（对光谱的不同频率有不同的采样值）下传递到透镜的能量，然后将生成的光谱转换为三色刺激颜色值（tristimulus color value，我们使用 RGB），将其累积在当前图像中。

重新定义图像贡献函数 f 以计算相应路径的光谱亮度 (luminance)（比如转换为 XYZ 三原色，然后用 Y 值表示亮度等），这意味着路径样本将根据理想图像的亮度分布，并且每个滤波的图像的样本亮度将相同（我理解的是用不同的初始 $\overline{X}_{0,j}$ 估计出多幅图像，然后最终合并到一起，每幅图像在计算贡献时的 W_i 都是同一种亮度，无论其颜色如何）。每个颜色组件 c_i 都可以用 c_i/p（p 是正比于场景亮度函数的概率密度，这意味着三原色的采样可以相互独立）的估计器形式来采样。由于人眼对亮暗比对其他颜色变化会更敏感，这种方式可以帮助降低感受到的噪声。

三　好的变异策略

Metropolis 算法的不利条件主要是后续样本都是相关的，导致比独立样本有更高的方差。比如每次路径的变异都非常小，或者太多变异被拒绝时，方差就会很大。

通过选择合适的变异策略，就可以最小化相关性。我们首先考虑变异策略应该有的特性，意图最小化最终图像的误差。然后我们描述三种具体的变异策略：双向变异 (bidirectional mutations)、扰动 (perturbations) 和镜头子路径变异 (lens subpath mutations)。这些策略都被设计来满足不同的目标子集，最终实现方案是综合使用这三种变异方法。

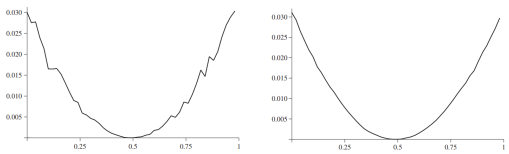
如何计算变异以及转移概率其实都挺麻烦的，要想将代码复现，难度不是一点半点（这也说明有些时候没有很好的资源和平台，无法去跟算法实现者获取源码都会给后续研究带来很大的困难）。

3 1　理想的变异方法的特性

高接收率。如果一个变异的接受率太小，那么由于高拒绝率，得到的路径序列就会出现一连串相同的值，导致高方差。

对路径有大的改变。即使大部分变异的接受概率都很高，但是如果路径变化太小（导致路径之间有很高的相关性），仍然会有很高的噪声（也相当于相似路径的重复）。

遍历性 (Ergodicity，各态历经性)。如果可允许的变异过于严格，那么有些时候就会被“卡住”，比如：



如果只允许增加或者删除顶点，那么上面的路径就无法变异得到其他有效路径（尤其是跨越到遮挡物对面的路径）。我们希望随机游走能够收敛于遍历性状态，这就要求对于任意的 $\bar{x}$ 和 $\bar{y}$ 当 $f(\bar{x}) > 0$ 且

$f(\overline{y}) > 0$ 时，要保证 $T(\overline{x} \to \overline{y}) > 0$（保证变异策略可以使得能够从一条有效路径转移到另一条有效路径上）。

改变对图像贡献的位置。为了最小化样本在图像平面位置的相关性，变异要能够改变镜头边 $x_{k-1}x_k$。注意路径的其他部分的突变并不会提供关于图像平面上的路径分布的信息。

分层性 (Stratification)。Metropolis 方法的另一个潜在问题在于样本在图像平面是随机分布的，并不能保证一个格子一个硬币（随机往 n 个格子抛 n 个硬币，不能保证每个格子都有硬币）。有许多像素可能都是空的，而有的像素可能会有不少样本，这种不均匀性会带来噪声。对于某些类型的变异，这种影响很难避免。然而，值得考虑的是变异是可能存在某种形式的分层的。

低计算消耗。变异要尽量计算不那么复杂，比如如果要重新变异出一条完全新的路径，那么计算量就会比只变化一两个顶点要大很多。我们现在考虑解决这些目标的一些特定的变异策略。注意，Metropolis 算法允许我们在设计采样策略时比标准蒙特卡洛算法有着更大的自由度，这是因为我们只需要计算每个突变的条件概率 $T(\overline{x} \to \overline{y})$：换句话说，允许变异策略可以只取决于当前的路径。

3 2 双向变异策略

双向变异策略可以算是 MLT 算法的基石，可以对路径做出较大改变（比如改变路径长度），其基本方法比较简单，即从当前路径 $\overline{x}$ 中选择一条子路径，并用另一条子路径替换它。我们把这个过程分为如下几步：（1）子路径顶点的删除。（2）产生新的子路径，将删除子路径后的路径连为一个完整路径。

3 2.1 子路径的删除

设当前路径是 $\overline{x} = \mathbf{x}_0, ..., \mathbf{x}_k$，我们设置一个删除子路径 $\mathbf{x}_l, ..., \mathbf{x}_m$ 的概率 $p_d[l, m]$。这条子路径的端点并没有被包括进来，因此 $\mathbf{x}_l, ..., \mathbf{x}_m$ 包括有 $m - l - 1$ 个顶点以及 $m - l$ 条边（$-1 \le l < m \le k + 1$）。

删除概率 $p_d[l, m]$ 是两个因素的乘积，第一个因素 $p_{d,1}$ 仅仅取决于子路径长度，它的目的是支持短的子路径的删除，替换路径计算量较小，而且接受率较高（变异程度较小）。第二个因素 $p_{d,2}$ 是为了避免变异的低接受率（会在后面的章节介绍）。

********** 上面有个遗留问题 **********

$p_d[l, m]$ 是归一化的，通过采样来决定要删除的子路径，删除后，$\overline{x}$ 变为两个部分：$\mathbf{x}_0, ..., \mathbf{x}_l$ 和 $\mathbf{x}_m, ..., \mathbf{x}_k$，我们需要生成一个新的子路径来把这两部分连接起来。

3 2.2 产生新的子路径

我们选择加入到两边的顶点数 l' 和 m'，这个步骤也分为两步：第一步是选择新的子路径长度 $k_a = l' + m' + 1$（如果没有新顶点加入，则子路径就是单独的一条边，将两个路径连接起来）。为了使得以前的路径和现在新生成的路径长度尽可能相似（保证这是较小的变异），根据一个离散概率 $p_{a,1}$ 来生成 k_a，$p_{a,1}$ 中保持总路径长度的概率要相对大一些。第二步就产生相应的 l' 和 m'，这是根据另一个离散概率密度 $p_{a,2}$ 来采样的，生成不同长度的 l' 的概率相同。我们令 $p_a[l', m']$ 表示 $p_{a,1}$ 和 $p_{a,2}$ 的乘积。

可以在端点处采样 BSDF 来扩充顶点，如果删除的子路径包括相机镜头点，就在相机镜头采样一个顶点；如果删除的子路径包括光源点，就在光源上采样一个点。最终，我们构建了一个新的子路径，然后测试子路径的端点之间是否可见，如果不可见，那么子路径就被抛弃了（如果采样顶点时没有与任何场景中的表面相交，就也算失败）。

注意，丢弃整个路径并从头生成新路径的概率是非零的，这将自动确保遍历性条件，因此算法不会被卡在某个路径子区域，除非生成的新路径一直贡献一直都是 0。

[1] 中给出了一些合理的参数值。$p_{d,1}[k_d]$ 的概率是删除一个长度为 $k_d = m - l$ 的子路径：

$$
\begin{aligned}
p_{d,1}[1] &= 0.25 \\
p_{d,1}[2] &= 0.5 \\
p_{d,1}[k_d] &= 2^{-k_d} \quad for\ k_d \ge 3
\end{aligned}
$$

增加子路径的长度为 k_a 的概率 $p_{a,1}[k_a]$：

$$\begin{aligned} &p_{a,1}[k_d] = 0.5 \\ &p_{a,1}[k_d \pm 1] = 0.15 \\ &p_{a,1}[k_d \pm j] = 0.2(2^{-j}) \quad for\ j \geq 2 \end{aligned} \tag{三.1}$$

3 2.3 接受概率的估计

接受概率可以写为：

$$\begin{aligned} R(\overline{x} \to \overline{y}) &= \frac{f(\overline{y})}{T(\overline{x} \to \overline{y})} \\ a(\overline{x} \to \overline{y}) &= \frac{R(\overline{x} \to \overline{y})}{R(\overline{y} \to \overline{x})} \end{aligned} \tag{三.2}$$

注意 R 跟蒙特卡洛估计 $f(\overline{y})/p(\overline{y})$ 很像，只是分母是条件概率。

$T(\overline{x} \to \overline{y})$ 包括了删除点的概率与添加点的概率的乘积，为了计算添加新点的概率，我们必须把所有的 $l' + m' + 1$ 种方式都考虑进来（下面第一列是表示 $\mathbf{x}_0, ..., \mathbf{x}_l$ 路径段上生成的新顶点数，第二列是 $\mathbf{x}_m, ..., \mathbf{x}_k$ 路径段上生成的顶点数，一共有 $l' + m' + 1$ 种可能）：

$$\begin{array}{ll} 0 & l' \\ 1 & l' - 1 \\ 1 & l' - 1 \\ ... & \\ l + m & 0 \end{array} \tag{三.3}$$

同时那些没有变的顶点不需要考虑在内。下面用一个例子来更好地说明整个过程。

3 2.4 一个例子

令 $\overline{x}$ 是一个路径 $\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$，假设随机变异删除了边 $\mathbf{x}_1\mathbf{x}_2$（注意不是删除了顶点，只是删除了这条边），用一个新的顶点 $\mathbf{z}_1$（从 $\mathbf{x}_1$ 采样射线追踪到的该点），新的路径就是 $\overline{y} = \mathbf{x}_0\mathbf{x}_1\mathbf{z}_1\mathbf{x}_2\mathbf{x}_3$，这对应于随机选择 $l = 1, m = 2, l' = 1, m' = 0$。

令 $P_{\sigma^\perp}(\mathbf{x} \to \mathbf{x}')$ 表示从 $\mathbf{x}$ 生成 $\mathbf{x}'$ 的立体角相关的概率密度（在双向方法中已经描述过很多次了）。转换为表面积相关的概率密度就是：$P_{\sigma^\perp}(\mathbf{x} \to \mathbf{x}')G(\mathbf{x} \leftrightarrow \mathbf{x}')$。于是我们目前就有了计算 R 的全部需要的量。

分子是（$R(\overline{x} \to \overline{y})$ 和 $R(\overline{y} \to \overline{x})$ 之间贡献的部分都给去掉了）：

$$f(\overline{y}) = f_s(\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{z}_1)G(\mathbf{x}_1 \leftrightarrow \mathbf{z}_1)f_s(\mathbf{x}_1 \to \mathbf{z}_1 \to \mathbf{x}_2)G(\mathbf{z}_1 \leftrightarrow \mathbf{x}_2)f_s(\mathbf{z}_1 \to \mathbf{x}_2 \to \mathbf{x}_3) \tag{三.4}$$

分母是：

$$\begin{aligned} T(\overline{x} \to \overline{y}) = p_d[1,2]\Big\{ &p_a[1,0]P_{\sigma^\perp}(\mathbf{x}_1 \to \mathbf{z}_1)G(\mathbf{x}_1 \leftrightarrow \mathbf{z}_1) \\ &+ p_a[0,1]P_{\sigma^\perp}(\mathbf{x}_2 \to \mathbf{z}_1)G(\mathbf{x}_2 \leftrightarrow \mathbf{z}_1)\Big\} \end{aligned} \tag{三.5}$$

其中 $p_a[1,0]$ 是表示 $p_{a,1}[2] \times p_{a,2}[1]$，注意添加新点的概率要把所有的可能都囊括在内，所以有 $p_a[1,0]$ 和 $p_a[0,1]$。如果新的子路径要添加两个点，那么就要计算：$p_a[0,2], p_a[1,1], p_a[2,0]$ 这三项了。

同理，我们也可以得出 $R(\overline{y} \to \overline{x})$（注意现在的 p_d 和 p_a 都指向了路径 $\overline{y}$）：

$$R(\overline{y} \to \overline{x}) = \frac{f_s(\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2)G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)f_s(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3)}{p_d[1,3]p_a[0,0]} \tag{三.6}$$

3 2.5 更一般的实现形式

我们描述一下更一般的实现形式，以及如何在计算上更有效率。

令 $\overline{x} = \mathbf{x}_0...\mathbf{x}_k$，令 $\mathbf{x}_l...\mathbf{x}_m$ 是删除的子路径，令 $\mathbf{z}_1...\mathbf{z}_{k_a-1}$ 是新路径的顶点（注意 $k_a = l' + m' + 1$）。这就产生了一个变异路径 $\overline{y}$ 形式为：

$$\begin{aligned}\overline{y} &= \mathbf{y}_0...\mathbf{y}_{k'} \\ &= \mathbf{x}_0...\mathbf{x}_l\mathbf{z}_1...\mathbf{z}_{k_a-1}\mathbf{x}_m...\mathbf{x}_k\end{aligned}$$

$k' = k - k_d + k_a$ 是新路径的长度（$k_d = m - l$ 表示删除的子路径的长度；$k_a = l' + m' + 1$ 表示新添加的子路径的长度）。

计算 R 的倒数很方便：

$$Q(\overline{x} \to \overline{y}) = \frac{1}{R(\overline{x} \to \overline{y})} = \frac{T(\overline{x} \to \overline{y})}{f(\overline{y})} \tag{三.7}$$

它的计算和之前双向方法中几乎是一样的。将 $\overline{y}$ 分为两个片段，使用新的子路径的第 i 条边作为连接边，考虑光源子路径和相机子路径：

$$\begin{aligned}&\mathbf{y}_0, ..., \mathbf{y}_{l+i-1} = \mathbf{x}_0, ..., \mathbf{x}_l\mathbf{z}_1, ..., \mathbf{z}_{i-1} \\ &\mathbf{y}_{l+i}, ..., \mathbf{y}_{k'} = \mathbf{z}_i, ..., \mathbf{z}_{\mathbf{k_a}-\mathbf{1}}\mathbf{x}_m, ..., \mathbf{x}_k \\ &1 \le i \le k_a\end{aligned} \tag{三.8}$$

第 i 条边就是 $\mathbf{z}_{i-1}\mathbf{z}_i$（第一条边是 $\mathbf{x}_l\mathbf{z}_1$）。两条子路径分别有 $s = l+1+i-1 = l+i$ 和 $t = (k'+1)-(l+i)$ 个顶点（这里的 s 和 t 和双向路径追踪中定义的 s 和 t 是完全一致的，令 C_i^{bd} 表示双向路径追踪中的非加权贡献：

$$C_i^{bd} = C_{s,t}^* \tag{三.9}$$

则 $Q(\overline{x} \to \overline{y})$ 就可以写为：

$$Q(\overline{x} \to \overline{y}) = p_d[l, m] \sum_{i=1}^{k_a} \frac{p_a[i-1, k_a - i]}{C_i^{bd}} \tag{三.10}$$

为了有效地计算上式的和，我们首先先计算 $C_{l'+1}^{bd}$，这意味着光源子路径上有 l' 个新顶点，相机子路径上有 m' 个新顶点（因为 $s = l + i$，光源子路径的顶点数是 $i - 1$，因此当 $i = l' + 1$ 时，光源子路径上的顶点数就是 l' 了）。我们计算 α_s^L 和 α_t^E 以及连接因子 $c_{s,t}$。如果贡献 $C_{l'+1}^{bd}$ 能够趋近于 0（比如可见性测试失败），那么变异就会被马上拒绝；否则我们就计算倒数 $1/C_{l'+1}^{bd}$ 以及其他的 $1/C_i^{bd}$，在一个循环中就可以有效地实现它们。

3 3 扰动策略

有些光照条件双向变异策略总是会被拒绝，比如一些小区域的路径空间路径贡献远大于其他区域，比如焦散、小孔径带来的困难的可见性、或者两个表面相交处的凹角（被积函数中的一种奇异形式，这里的奇异表示不可积分或者积分困难的形式）。问题是双向突变相对较大，因此它们通常试图在高贡献区域之外的路径上进行变异。

使用小变异可以增加接受率，原则是附近路径会对图像有相似贡献，接受率就会高一些，由此可以探索周边的路径。

我们的方法是选择当前路径的一个子路径，然后轻微地移动它的顶点，我们把这种变异叫做扰动 (perturbation)。这种方法可以应用于任意的子路径，我们主要感兴趣的地方在于包括了边 $\mathbf{x}_{k-1}\mathbf{x}_k$ 的扰动（因为其他更改无助于防止同一图像点上的长样本序列）。我们实现了两种扰动方式，一是镜头扰动，二是焦散扰动。

3 3.1 镜头扰动

我们删除具有 $(L|D)DS^*E$ 形式的子路径 $\mathbf{x}_m...\mathbf{x}_k$（全路径公式中 DSE 表示针孔相机，DDE 表示有限孔径相机；但这里的符号用的只是基本的 Heckbert 正则表达式，默认相机是 $D(D|S)E$ 类型，光源射入到相机 E 时的最后一段路径可以都是镜面反射），该子路径称为镜头子路径，有 $k-m$ 条边和 $k-m-1$ 个顶点（不包括顶点 $\mathbf{x}_m$）。我们要求 $\mathbf{x}_m$ 和 $\mathbf{x}_{m+1}$ 都不能是镜面的，否则就会导致生成的扰动路径 $f(\bar{y})=0$。

我们把旧路径在图像上的位置沿着随机方向 ϕ 移动随机距离 R，角度 ϕ 是均匀随机选择的，而 R 是两个变量 r_1 和 r_2 之间指数分布的值：

$$R = r_2 \exp(-\ln(r2/r1)U) \tag{三.11}$$

U 是在 $[0,1]$ 之间的均匀分布。

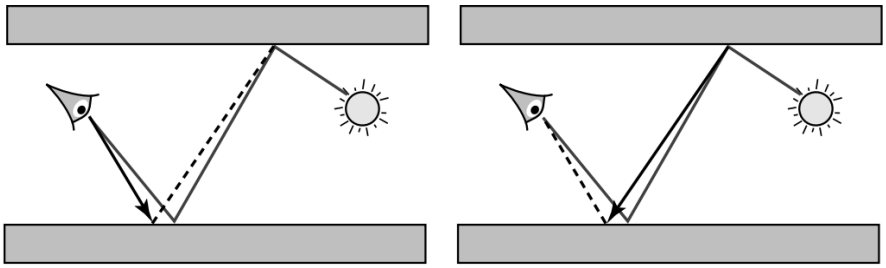
然后，我们从新的图像位置上发射一条光线，并通过额外的镜面反射将子路径延伸到与原始图像相同的长度。每次镜面反射时的散射模式都会被保留（即镜面反射或透射），而不是进行新的随机选择（也就是说，一开始的子路径最后包含几次镜面散射，那么变异以后还需要包含相应的镜面散射次数，以保证相似的路径可以贡献到其他像素中，如果扰动将顶点从镜面材料移动到非镜面材料，则变异将立即被拒绝。）这允许我们有效地采样罕见的事件组合，例如透明表面 99% 的光穿透了过去，而 1% 的光被镜面反射——当只有其中某些组合对图像有贡献时，这一点很重要：例如，考虑一个包含玻璃窗的场景模型，其中窗外的环境是黑暗的，在这种情况下，只有来自窗口的反射才会对图像产生显著影响。

接受概率与双向变异的计算是相似的，主要区别在于这里使用了在图像平面采样一个点的概率。

3 3.2 焦散扰动

焦散路径一般表示为 LS^+DE，镜头扰动并不能接受这种变异。

扰动这种路径需要另一种方法，事实上任何形似 $(D|L)S^*DE$ 的路径 $\mathbf{x}_m...\mathbf{x}_k$ 都可以这么变异。



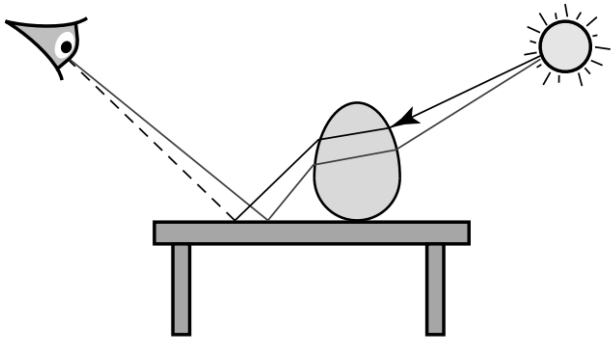
Lens perturbation　　Caustic perturbation

我们从 $\mathbf{x}_m$ 产生一条随机方向 $\mathbf{x}_m \to \mathbf{x}_{m+1}$，其中，$\theta=0$ 的轴对应于先前光线的方向，ϕ 是随机选择的，θ 是两个值 θ_1 到 θ_2 之间指数分布的：

$$\theta = \theta_2 \exp(-\ln(\theta_2/\theta_1)U) \tag{三.12}$$

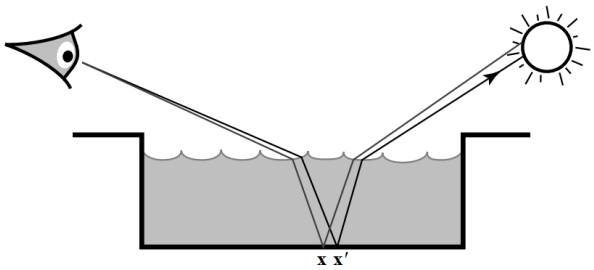
其他部分与镜头扰动相似，每次散射的模式都要保留。

一个焦散扰动的例子：

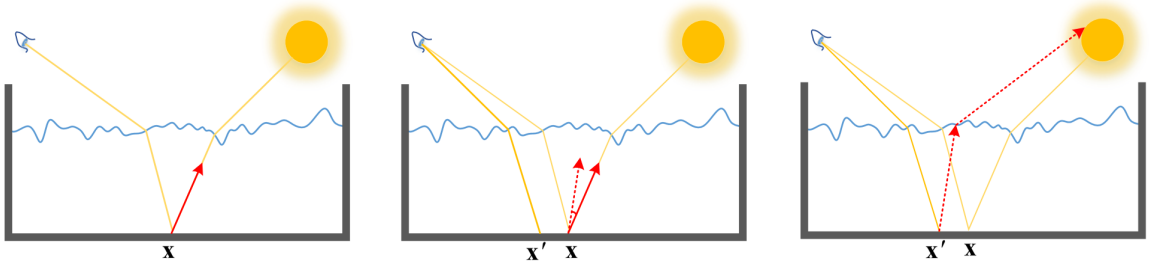


3 3.3 多链扰动 (Multi-chain perturbations)

对于 $(D|L)DS^+DS^+E$ 的情况，上面的方法并不能对其进行变异。这可以通过扰动超过一个镜面链来实现，首先使用一个镜头扰动对第一个链 DS^+E 来产生一个变异；然后通过扰动原来路径的第一条边，来得到 DS^+D 的第一个顶点。下图是一个示例：



过程如下：



第一张图表示一开始的路径；第二张图表示镜头扰动，得到点 $\mathbf{x}'$，且新方向会根据 $\mathbf{x}$ 原来的方向（红色实线）进行轻微扰动，得到新方向（红色虚线）；第三张图表示 $\mathbf{x}'$ 处使用新方向构成完整路径。

前面用到的参数值 θ_1 和 θ_2 的选取在 [1] 有描述。

3 4 镜头子路径变异

镜头子路径变异可以在整个图像平面采样，每次变异包括删除当前镜头子路径并添加一个新的镜头子路径。和之前一样，镜头子路径要有 $(L|D)S^*E$ 的形式，镜头子路径在图像平面是分层的，因此每个像素都能接受到相同数量的镜头子路径变异。

首先，用 n' 个独立的种子路径来初始化算法。我们也要存储一个当前镜头子路径 $\bar{x}_e$，一个镜头子路径变异包括删除当前路径 $\bar{x}$ 的镜头子路径，然后用 $\bar{x}_e$ 去替换它（当此时的变异类型是镜头子路径变异，而不是双向变异或者扰动变异时，就这么操作）。当 $\bar{x}_e$ 被用过一定次数 n_e 后，就丢弃它并产生一个新的镜头子路径。注意 $n' >> n_e$ 避免相同的镜头子路径在同样的路径中被使用超过一次。

每个镜头子路径都是都是通过从图像上某个随机点投射一条光线，通过大于等于零次镜面散射，最终到达一个非镜面顶点（对于同时包含镜面和非镜面组件的材质，我们随机选择一种属性）。为了在图像平面上对样本进行分层，我们对每个像素处生成的镜头子路径的数量进行计数。当产生一个新的子路径时，我们选择一个随机像素并增加该像素上的计数，当一个像素已经达到其镜头子路径数目配额时，我们将使用漫游 (rover) 的概念（以某些内存管理方案中的类似想法命名）来搜索没有达到配额数量的像素。

漫游就是简单地图像像素的伪随机排序的索引，这样每个像素都会出现一次。如果从第一步随机选择的像素是满的，我们检查对应于漫游的像素，如果需要，我们以伪随机顺序（根据索引来）访问其他像素，直到找到没有满的像素。我们还通过计算泊松最小圆盘模式 (Poisson minimum-disc pattern) 并将其平铺在图像平面上，来控制每个像素内的样本分布。

3 5 从变异类型中进行选择

每一步中我们都为三种变异类型分配概率，采样这个离散分布来选择当前变异类型。让三种变异方案相对平衡很重要，因为变异类型是被设计来满足不同目标的，很难预测哪种类型更有效，总目标是让变异程度尽可能大，并保证足够大的接受率。

这和多重重要性采样很像，我们定义了很多种变异策略，并不知道哪种最优，所以一起使用它们。包含特殊情况的变异虽然会增加被拒绝的次数，但是却使得算法更加鲁棒。

四　MLT 算法技术改进

本节描述一些提高 MLT 效率的改进方案。

4 1　直接光照

一般场景中，直接光照用标准方法即可，比 Metropolis 计算消耗更低。

这种优化可以作为镜头子路径变异策略的一部分来完成，该策略已经在每个像素处生成了固定数量的子路径。为了计算直接照明，我们在生成每个透镜子路径时执行标准光线跟踪计算（与当前 MLT 路径无关）。这些样本的贡献聚集方法与 Metropolis 样本相同。我们还需要从算法的 Metropolis 部分删除直接照明路径，但这很容易：当变异生成直接照明路径时，我们只需拒绝它。一个更好的方法是修改变异策略本身，以避免首先生成这些路径。

最后请注意，如果照明特别困难（例如，由于可见性问题，一些场景不存在直接照明），那么直接照明“优化”可能是一个缺点。例如，想象一座有许多房间和灯光的大型建筑，但当前我们只能看到一个房间，除非直接光照策略能够很好地排除所有不重要的灯光，否则 MLT 可以大大提高效率。

4 2　使用期望值

在某次变异时，接受概率为 $a(\overline{x} \to \overline{y})$，拒绝概率为 $1 - a(\overline{x} \to \overline{y})$。我们可以通过始终在两个位置累积一个样本，并通过相应的概率进行加权，从而提高效率，即：

$$path\ 1\ :\ a(\overline{x} \to \overline{y}) f(\overline{y})$$
$$path\ 2\ :\ \big(1 - a(\overline{x} \to \overline{y})\big) f(\overline{x})$$

$path\ 1$ 和 $path\ 2$ 分别贡献到它们对应的像素上（如果在同一个像素就都贡献到一个像素中）。实际上，这种优化将随机变量替换为其期望值，这对于对图像的暗淡区域进行采样尤其有用，否则暗淡的地方会接收很少的采样。注意，这种优化不会影响随机游动本身；以与之前相同的方式接受或拒绝每个转换（这只是用来计算贡献，之后，新路径 $\overline{x}'$ 还是按照接受概率来选择是 $\overline{x}$ 还是 $\overline{y}$）。

4 3　两个阶段的 MLT

图像亮度变化较大时，MLT 算法主要在明亮的部分采样，意味着明亮的部分计算的更准确。更准确的说，像素 j 的方差正比于 I_j，标准差正比于 $\sqrt{I_j}$，相对误差正比于 $1/\sqrt{I_j}$。作为第一种近似，所有像素的相对误差最好相同（因为人眼对对比度差异很敏感）。为了实现这一点，我们需要一种算法，该算法在每个像素处生成大致相同数量的样本（样本值根据理想图像的亮度而变化）。

通过以低采样密度预计算测试图像 I_0，可以容易地修改 MLT 算法以接近该目标。然后，我们不是根据图像贡献函数 f 进行采样，而是根据：

$$f'(\overline{x}) = f(\overline{x}) / I_0(\overline{x}) \tag{四.1}$$

$I_0(\overline{x})$ 仅仅与 $\overline{x}$ 在图像上的位置有关。然后再 MLT 算法运行时，需要补偿它，也就是 MLT 的样本值乘以相应的 $I_0(\overline{x})$ 然后再贡献到图像上（不能等图像计算完以后再对每个图像的每个像素乘以 $I_0(\overline{x})$，因为这个权重是对应于每个样本的，而不是对应于每个像素的）。这并不会引入任何偏差，这仅仅意味着更亮的区域可以用少量计算值更大的样本来估计得到。但是有可能生成的测试图像中有不少地方都是黑的（导致后续计算错误），这时可以通过滤波来使这些像素的值大于 0，或者是只抽取测试图像中最亮的那一部分保留原值，其他部分都设置为统一的值。

这种方法对于图像范亮度变化较大时会很有效，比如最亮的区域一般是直接光照或者相机直接能看到光源，这些区域不要太多样本也是有好处的。

4 4　对变异策略进行重要性采样

我们描述一种总体上可以增加 MLT 接受率的方法，它提升的是平均接受率。其思想是在决定尝试哪一个突变时，通过根据删除的子路径可以重新生成的概率对每个可能的变异进行加权，实现一种关于 $a(\overline{x} \to \overline{y})$ 的重要性采样形式。

令 $\overline{x} = \mathbf{x}_0 ... \mathbf{x}_k$，考虑删除了一段子路径 $\mathbf{x}_l ... \mathbf{x}_m$，给定已经删除的子路径，已经可以计算出接受率的一部分内容了。注意 $a(\overline{x} \to \overline{y})$ 正比于：

$$Q(\overline{y} \to \overline{x}) = 1/R(\overline{y} \to \overline{x}) \tag{四.2}$$

给定 $\overline{x}$，我们容易计算出除了 p_d 和 p_a 之外的其他部分（因为这些概率取决于 $\overline{y}$，而 $\overline{y}$ 还没有被产生）。如果我们简单地设置这些未知量为 1，即：

$$p_{d,2} = \sum_{i=1}^{k_a} \frac{1}{C_i^{bd}} \tag{四.3}$$

通过对每种变异类型的概率密度加权，就可以避免一些不太可能被接受的路径。

五　结果与展示

MLT 确实在很多场景都比 BDPT 和 PT 有更好的效果，由于 [1] 上关于本节的内容较为简单，所以不再赘述，而是重点说一些值得关注的问题。

5 1　测试例子 1

下图显示了具有困难间接照明的测试场景。这个场景中的所有光线都来自一个稍微打开的门，这让相邻房间中大约 0.1% 的光线通过。光源是位于房间远端的漫射天花板（相当大），因此大部分光线通过门口已经反射了好几次。

对于相同的计算时间，MLT 比双向路径跟踪的结果要好得多。注意，使用许多光线传输算法很难获得的细节：接触面之间的阴影、玻璃茶壶下的焦散、门下白色瓷砖反射的光线，以及沿地板背面的较亮条带（由于桌子和墙壁之间的间隙很窄）。此场景包含漫反射、光泽和镜面反射曲面，并且墙面未设置纹理以清楚显示噪波级别。

在这种情况下，MLT 通过仅改变部分路径的能力获得了很高的效率。穿过门口的那部分路径可以被保留下来，并重新用于许多突变，直到它成功突变为穿过门口的另一条路径。请注意，扰动并不能很好地使这一过程有效，因为通过门口的路径不需要频繁地改变。

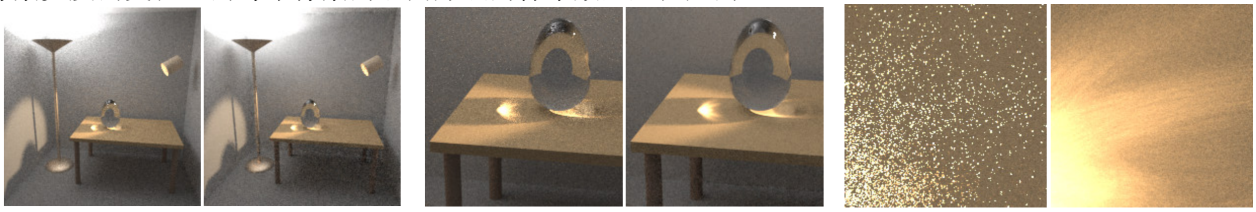
5 2　测试例子 2

本小节比较了具有强间接照明和焦散的场景的 MLT 和双向路径跟踪。这两种方法在图像的上面部分（落地灯的间接照明占主导地位）给出了类似的结果。然而，当我们放大焦散时（注意不是放大图像，是相机去靠近焦散使得它占据视野的范围变大），MLT 表现得更好，因为它能够通过扰动现有路径来生成新路径。

在相同的计算时间内，图像质量随着放大而降低，但只会缓慢下降。这是因为当我们放大焦散时，平均突变成本会上升（因为每次成功的扰动都需要至少四次光线投射操作）。一旦焦散填充整个图像，图像质量随着继续放大几乎保持不变。

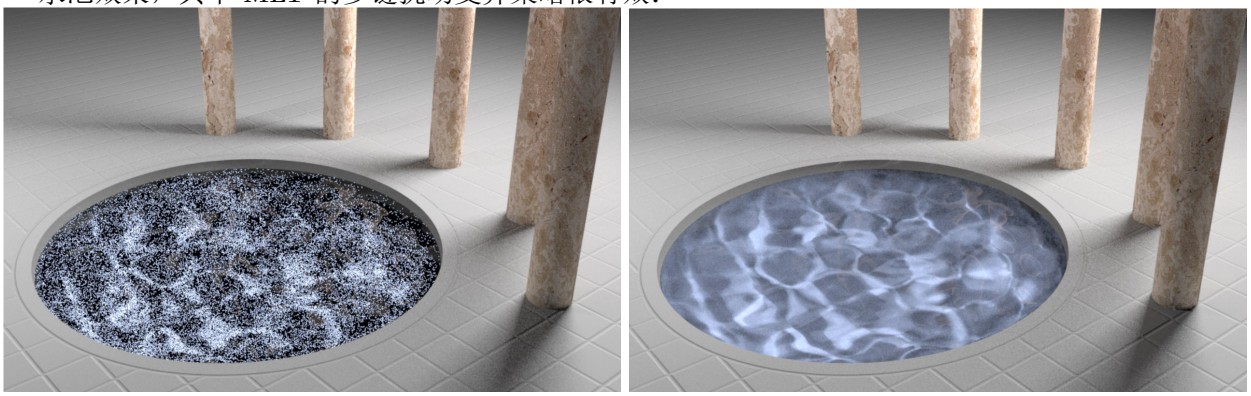
请注意在最高放大率下噪波的条纹外观，这是由于焦散扰动：来自聚光灯的每一条光线都在一个狭窄的锥体内受到扰动；然而，透镜将这个方向锥映射成细长形状。这些条纹是由于长串的焦散变异造成的，这些变异没有被其他类型的成功的变异所打破。

即使在图像的顶行，这两种方法之间也有细微的差异。MLT 算法在图像的亮区域中产生较低的噪声，而双向算法在暗区域中产生较小的噪声。这是我们所期望的，因为 Metropolis 算法生成样本的数量根据像素亮度而变化，而每个像素的双向方法的样本数量是恒定的。



5 3　测试例子 3

水池效果，其中 MLT 的多链扰动变异策略很有效：



六　总结

这样介绍下来，大家可能对 MLT 算法仍然有很多不太理解的地方，比如一开始如何初始化，路径变异时对各个像素的贡献如何计算等。现在大部分渲染器都是使用 PSSMLT 算法（一种实现更简单的 MLT），Mitsuba 中 MLT 和 PSSMLT 两种算法都实现了，大家有兴趣可以参考一下 Mitsuba 中的实现原理（Mitsuba0.6 中 MLT 会借助 BDPT 中的路径采样功能，而不是单独全部实现新的代码）。以后有时间我会继续更新 Mitsuba 渲染器的解读，可能预计得一两年以后了。

参考文献

[1] Veach E . Robust Monte Carlo Methods for Light Transport Simulation[J]. Ph.d.thesis Stanford University Department of Computer Science, 1998.

[2] Arvo, J. [1995]. Analytic Methods for Simulated Light Transport, PhD thesis, Yale University.

[3] 秦春林. 全局光照技术：从离线到实时渲染. 电子科技大学出版社.

[4] https://zhuanlan.zhihu.com/p/459580639

[5] https://zhuanlan.zhihu.com/p/28345852

[6] https://www.cnblogs.com/starfallen/p/3509234.html