基于重要性采样的自适应 Monte Carlo 光线追踪

什么是 Monte Carlo 光线追踪

光线追踪（ray tracing）是三维计算机图形学中的一种渲染算法，跟踪从眼睛发出的光线而不是光源发出的光线，通过这样一项技术生成编排好的场景的数学模型显现出来。这样得到的结果类似于光线投射与扫描线渲染方法的结果，但是这种方法有更好的光学效果，例如对于反射与折射有更准确的模拟效果，所以当追求高质量的效果时经常使用这种方法。当计算一个物体表面对 o 方向的反射光时，根据基于物理的光照计算公式，该反射光计算公式为：

$$L_o = \int_\Omega f_r L_i cos\theta_i d\omega_i = \int_\Omega (k_d \frac{c}{\pi} + k_s \frac{DFG}{4cos\theta_i cos\theta_o}) L_i cos\theta_i d\omega_i$$

通过对物体表面周围半球面的入射光照和反射系数的积分计算可以获得其在 o 方向的反射光。

然而真正去精确计算这些积分带来的效果过大，于是 Monte Carlo 方法被引入，它是一种通过采样来估计积分的方式，计算如公示 1 所示

$$F_n(X) = \frac{1}{n} \sum_{k=1}^{n} \frac{f(X_k)}{pdf(X_k)}$$

公式 1

通过 n 次采样的结果除于采样的概率密度再做平均来得到积分的估计，这是一个无偏估计，并且随着 n 的增加对积分的估计会越来越准确，既方差边小，这就引出了本文讨论的一个重要问题，如果在 n 有限的情况下，如何提高估计的准确度，降低方差（降噪），这其中一个重要的优化方法便是重要性采样以及基于重要性采

样引出的一系列优化策略。

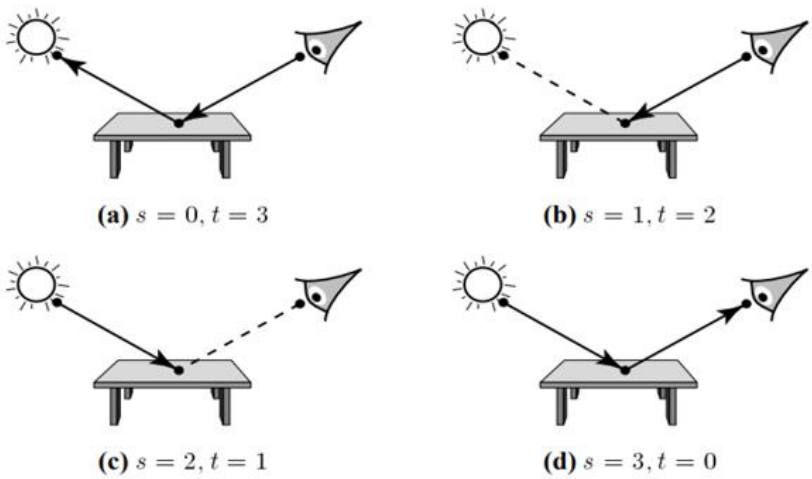
重要性采样

重要性采样[Veach and Guibas 1995a]是通过改变 Monte Carlo 采样的概率密度来降低采样结果的方差，具体论证这里简略就结论而言，当 Monte Carlo 采样的概率密度与所需要求解的积分形状越是相似，方差越小，而关于如何构造出与积分方程相似的概率密度以及如何基于这些构造出来的概率密度进行采样又是一系列被广泛研究的话题，其中重要性重采样是解决非解析解概率密度采样困难的做法之一，而一系列的拟合方式是解决构造积分方程相似概率密度的做法之一。

双向光线追踪 BDPT

常规的光线追踪的光路从摄像机出发，在场景中不断弹射直到达到弹射上限或者打出场景外，但这会导致一个弊端，当场景的光源体非常小时，通过这种弹射方式去弹射到场景中光源的概率非常小，对应到重要性采样的概念上就是我们采样的概率密度和实际的积分场大小非常不一致，带来了极大的方差。

既然从摄像机发射光线打到光源非常困难，那么我们扩展下采样方法，使得光源也向外打出光线，随后连接摄像机的光线和光源发出的光线来构成光路，这便是双向光线追踪 BDPT，示意图如图表一，效果对比如图表二。



图表 1

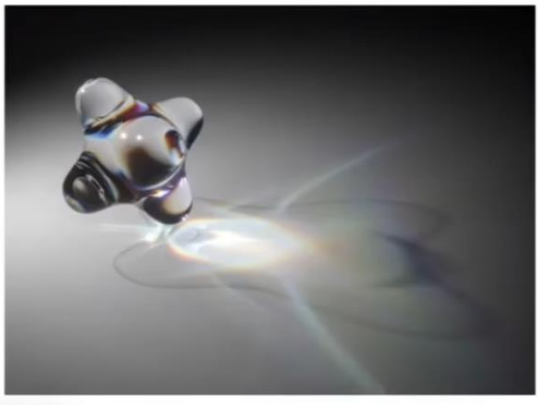
(a) Bidirectional path tracing with 25 samples per pixel

(b) Standard path tracing with 56 samples per pixel (the same computation time as (a))

图表 2

光子映射

尽管我们有了 BDPT 可以解决光源难以被弹射到的问题，但是还存在一种场景，即使使用 BDPT 依然得不到较好的效率，它在自然界中被称为 Caustic 现象，现象如图三所示，具体到光线路径的类型表示为 SDS 路径，既 Specular 到 Diffuse 再到 Specular 的光路。



图表 3

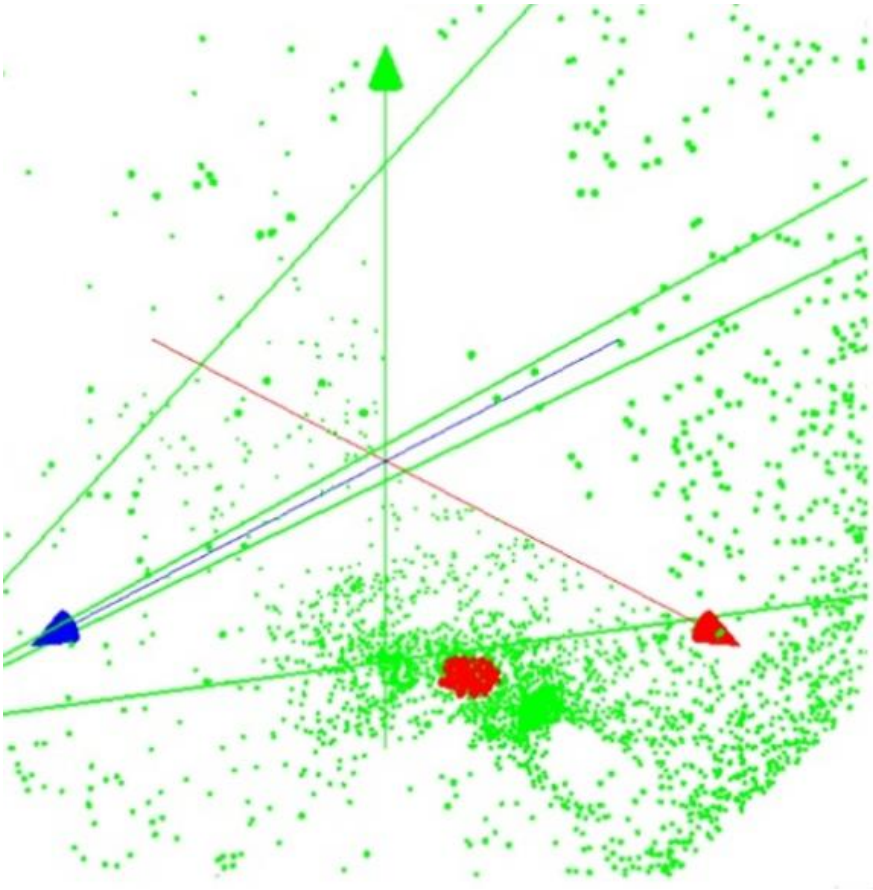
对于这种现象，当前普遍的做法叫做光子映射，光子映射被分为两个 pass

第一阶段，构建一张光子图，存储从光源发射的所有光子的通量信息。

第二阶段，从相机进行传统的路径追踪（path tracing），在追踪到漫反射表面的时候，统计附近的光子信息，并根据这些信息计算出最终的辐射率（radiance）。

光子贴图的构建

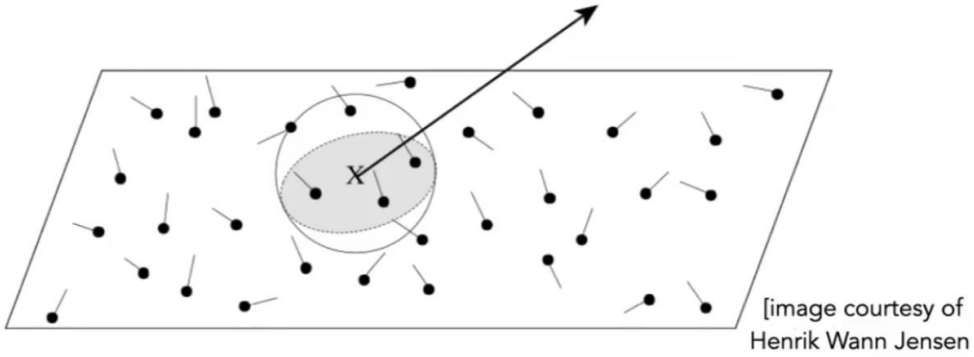
光子图的构建规则就是：从光源发射光子，并让光子在场景中反复弹射，每次击中漫反射表面都进行一次光子的纪录，直到被某个漫反射表面彻底吸收掉为止，如图四所示。



图表 4

利用光子计算全局光照

光线追踪(ray tracing)，在光线弹射的时候，根据 photon map 统计光路顶点周围的光子密度，从而计算全局光照，如图五所示。



图表 5

其公式为公式二表达：

$$L(x,\omega) \approx \sum_{p=1}^{n} \frac{f_r(x,\omega,\omega_p)\phi_p(x_p,\omega_p)}{\pi r^2}$$

公式 2

其中 n 代表的是，我们在光线交点向外搜寻最近的 n 个光子，r 代表搜寻到最近 n 个光子所需要的最大包围球半径，通过这种方式来近似该点周围的光照辐射密度。

最新相关论文介绍

利用重要性采样进行降噪的一种典型的做法就是 the vertex connection and merging(VCM),他组合了大量的采样技术，暂时我们可以简单理解为是 BDPT 和光子映射的结合，其中每一种采样策略都是可以对某一些特定的场景渲染取得非常高的效率[Georgiev et al. 2012; Hachisuka et al. 2012]，这使得 VCM 的健壮性非常高，在各种各样的场景下都能取得不错的效果，但是真是因为对于每种技术都兼顾到了，所以导致它在某一种场景下，往往有很多的采样策略比较低效，并且也并非所有相关的生产人员都深入了解图形学，因此在现代很多应用场景中，渲染器反而会选择那些简单朴素的采样策略，以避免去增加使用者的负担，这些简单的采样策略可以对大部分常规场景发挥作用，但是对于一些复杂的，间接的或者焦散现象存在的场景效果较差[Křivánek et al. 2018]。由此引出了接下来要介绍的一篇

2022 年的论文 Efficiency-aware multiple importance sampling for bidirectional rendering algorithms，这篇论文提出了一种自适应的方法，它可以在渲染场景的时候同时对场景进行检测，这种检测会辨析出来在该场景下，哪些采样策略是高效的，从而让我们把更多的性能用在高效采样策略上，降低低效采样数量或者直接不使用哪些低效的采样策略（采样数为 0）。

方法介绍

这篇文章的目标是优化 MIS 估计器中的样本分配，这将提高他们的效率。他们通过暴力比较不同采样策略样本分配情况下的伪方差（用于近似方差），然后选择伪方差最小的采样策略作为后续的采样策略。

VCM 下 Monte Carlo 计算公式如公式三所示：

$$I = \int_{\mathcal{X}} f(x)\,\mathrm{d}x, \qquad \langle I \rangle_{\mathbf{n}} = \sum_{t=1}^{T} \sum_{i=1}^{n_t} w_{\mathbf{n},t}(x_{t,i}) \frac{f(x_{t,i})}{n_t p_t(x_{t,i})}.$$

公式 3

其中 Wn,t 代表的是在第 t 种策略下，第 i 个采样结果的权重函数，该权重函数的作用是使得合并多种采样策略的结果时得到的方差尽可能小，一种比较优秀的权重函数也是本文中采用的权重函数如公式四[Grittmann et al. 2019, 2021]所示：

$$w_{\mathbf{n},t}(x) = \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)},$$

公式 4

其中 Ct 是矫正函数，Nt 代表该策略下的采样数，Pt 代表当 X=x 时，采样概率密度的值。

优化目标函数表示公式五：

$$\arg\min_{\mathbf{n}} C(\mathbf{n}) \sum_{j=1}^{P} \frac{V[\langle I_j \rangle_{\mathbf{n}}]}{I_j^2}.$$

公式 5

其中 V 代表的是第 n 种策略下，第 j 个像素的方差，I 是对方差进行归一化的一个参数，也可以选择不添加上去。

但是由于 V = E(X2) – （EX）2，EX 正是我们的所求，导致 V 无法得到，因此在本文中特别重要的一个思想便是直接用 E(X2)去近似方差 V，这种近似在绝大部分情况下都是效果很好的，仅有两种情况会导致近似会有比较大的偏差，一种是当多种采样技术中，有一种采样技术的方差非常非常低[Grittmann et al. 2019; Veach 1997]，另一种情况是当采样具有相关性时[Grittmann et al. 2021]，但这篇文章用实验的方式得出这种近似所得到的效果非常好。

于是最终优化目标方程如公式六所示：

$$\mathbf{n} = \arg\min_{\mathbf{n}} C(\mathbf{n}) \sum_{j=1}^{P} \frac{M[\langle I_j \rangle_{\mathbf{n}}]}{I_j^2},$$

公式 6

代码实现总流程

```
1: function FindBestStrategy(m, n^1, ..., n^N, C)
2:     M, I = ComputeMomentImage(m, n^1, ..., n^N)
3:     {p_j} = PixelLevelOptimize(M, n^1, ..., n^N, C)
4:     i = ImageLevelOptimize(M, I, n^1, ..., n^N, C, {p_j})
5:     return {p_j}, i  ← Set of pixel-level sample counts, image-level sample counts
```

图表 6

代码实现总流程如图六所示。

1.生成一个实验性策略 m 和 N 个候选策略{n1 , . . . , nN }.每一个策略可以被划分为两个部分 n = (p, i)，p 代表像素级采样信息，i 代表图像级采样信息

2 计算出 m 策略下的图像信息 I 和 N 个候选策略所对应的（M1.M2..MN）

3.遍历 n 个采样策略的像素级采样信息，选择最佳的一个作为最终像素级采样信息

4.遍历 n 个策略的图像级采样策略，确定最佳图像级采样信息

Pixel-level optimization

Pixel-level optimization 代码流程表示如图七所示

```
function PixelLevelOptimize(M, n^1, ..., n^N, C)
    {M̃_{1,1}, ..., M̃_{P,N}} = FilterImage({⟨M_1⟩_1 ... ⟨M_P⟩_N})
    for pixel index j = 1..P do                    ↰ Reduce noise in moments
        (p_j, _) = arg min M̃_{j,n} C_j(n)          ← Linear search
    return FilterImage({p_1, ..., p_P})
                                   ↰ Avoid abrupt changes in sample counts
```

图表 7

优化目标公式给如公式七所示：

$$(\mathbf{p}_j, \cdot) = \underset{\mathbf{n} \in \{\mathbf{n}^1 \ldots \mathbf{n}^N\}}{\arg\min}\, M[\langle I_j \rangle_{\mathbf{n}}] C_j(\mathbf{n}),$$

公式 7

通过公式七的筛选，我们得到了所有像素对应的最佳采样策略。

Image-level optimization.

function IMAGELEVELOPTIMIZE($\mathbf{M}, \mathbf{I}, \mathbf{n}^1, \ldots, \mathbf{n}^N, C, \{\mathbf{p}_j\}$)
  $\{\tilde{I}_1, \ldots, \tilde{I}_P\}$ = DENOISEIMAGE($\langle I_1 \rangle, \ldots, \langle I_P \rangle$)
  $\{R_\mathbf{i}\} = \{0, \ldots, 0\}$ ← Init. total rel. moment per image-level candidate
  $\{C_\mathbf{i}\} = \{0, \ldots, 0\}$ ← Initialize total cost per image-level candidate
  for pixel index $j = 1..P$ **do**
    for all candidate strategy index $k = 1..N$ **do**
      if $\mathbf{p}_j \neq \mathbf{p}^k$ **then**
        continue ← Skip candidates with different local counts
      $R_{\mathbf{i}k}$ += $\langle M_j \rangle_k \cdot \tilde{I}_j^{-2}$ ← Accumulate relative moments
      $C_{\mathbf{i}k}$ += $C_j(\mathbf{n}^k)$ ← Accumulate cost
  return $\arg\min\{R_\mathbf{i} \cdot C_\mathbf{i}\}$ ← Pick best image-level counts via simple search

图表 8

Image-level optimization 代码流程如图八所示：

它遍历每一个像素和每一种采样策略，并且累加每一种采样策略对映下的所有图像级消耗，优化目标如公式九所示：

$$\mathbf{i} = \underset{\mathbf{i} \in \{\mathbf{i}^1 \ldots \mathbf{i}^N\}}{\arg\min} \sum_{j=1}^{P} \frac{M[\langle I_j \rangle_{(\mathbf{p}_j, \mathbf{i})}]}{I_j^2} \sum_{j=1}^{P} C_j(\mathbf{p}_j, \mathbf{i}).$$

公式 8

The Key of this algorithm

上诉代码流程里面最关键的一点就是如何能够廉价的计算出每个采样策略下的二阶矩矩阵，如果丹村采用纯粹的暴力求解，那么每计算一次二阶矩的代价近似等于做一次渲染，

为了解决这个问题作者做了多步近似。

First Step

第一步的思想是，对于各种不同二阶矩，是否可以引入一个矫正系数然后通过一个标准二阶矩乘以该矫正系数来得到，公式如公式九，十表示：

$$M[\langle I\rangle_{\mathbf{n}}] = \int_{\mathcal{X}} \frac{f^2(x)\sum_t c_t(x) w_{\mathbf{m},t}(x)}{\sum_t c_t(x) m_t p_t(x)} \delta_{\mathbf{n},\mathbf{m}}(x)\,\mathrm{d}x,$$

公式 9

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \frac{\sum_t c_t(x) m_t p_t(x)}{\sum_t c_t(x) n_t p_t(x)} \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)}$$

公式 10

其中公式九为引入了校正系数后第 n 种二阶矩的计算方程，公式十为校正系数公式。

SecondStep

对上述校正系数公式进行变形，公式如公式十一，十二表示：

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \left(\sum_t \frac{m_t}{n_t} w_{\mathbf{n},t}(x)\right) \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)}.$$

公式 11

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \left(\sum_t \frac{m_t}{n_t} w_{\mathbf{n},t}(x)\right) \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)}.$$

公式 12

然后经过变形以后，关于权重函数 Wn,t 依然于每一种采样技术的每一个采样点，在性能上并没有得到显著提升。

ThirdStep

既然二阶矩可以通过矫正系数来进行表达，那么同样的思路应用于权重函数，我们引入标准权重函数和相应的矫正函数去进行变形。

其如公式十三所示：

$$w_{\mathrm{n},t}(x) = \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{24a}$$

$$= \frac{n_t}{a_t} \frac{c_t(x) a_t p_t(x)}{\sum_k c_k(x) a_k p_k(x)} \frac{\sum_{t'} c_{t'}(x) a_{t'} p_{t'}(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{24b}$$

$$= \frac{\frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)}. \tag{24c}$$

公式 13

最终矫正函数的计算公式如公式十四：

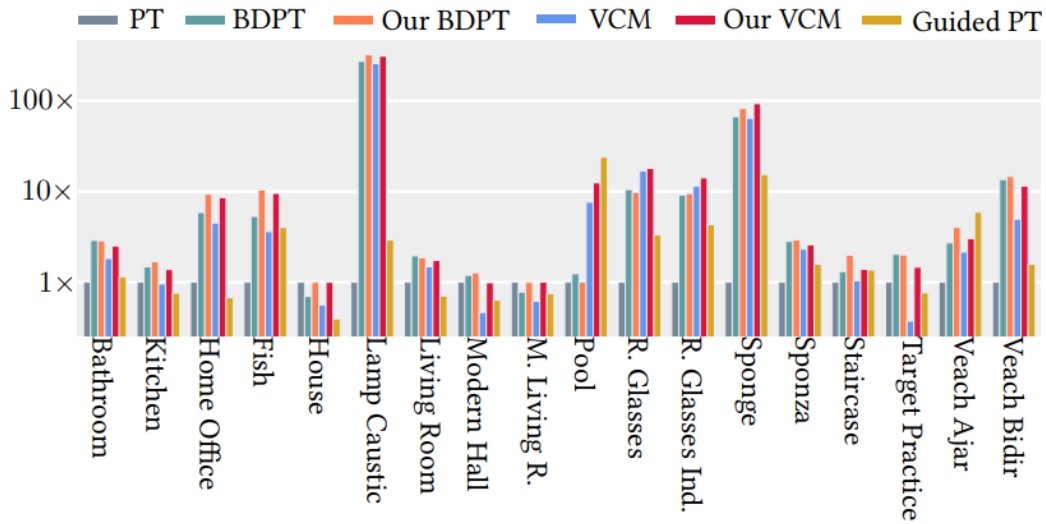
$$\delta_{\mathrm{n,m}}(x) = \frac{\left(\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)\right)^2 \sum_t c_t(x) \frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\left(\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)\right)^2 \sum_t c_t(x) \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}$$

公式 14

至此我们不再需要去深入到每一个像素去进行计算，大大提高了这个算法的运行效率。

结果展示

Speed-up	vs. path tracing		vs. vanilla	
	BDPT	VCM	BDPT	VCM
Average	3.14×	5.00×	1.18×	1.68×
Worst	0.99×	0.98×	0.95×	1.12 ×
Best	97.09×	600.77×	1.60×	5.32×



图表 9

如图表九所示，作者在大量不同特征的场景下对比了使用其方法的路径追踪和 VCM 渲染和不适用的路径追踪和 VCM 渲染的运行效率，使用了其方法的在平均效率和最佳效率上都得到了显著提升，并且最坏情况也与原方法相差无几。

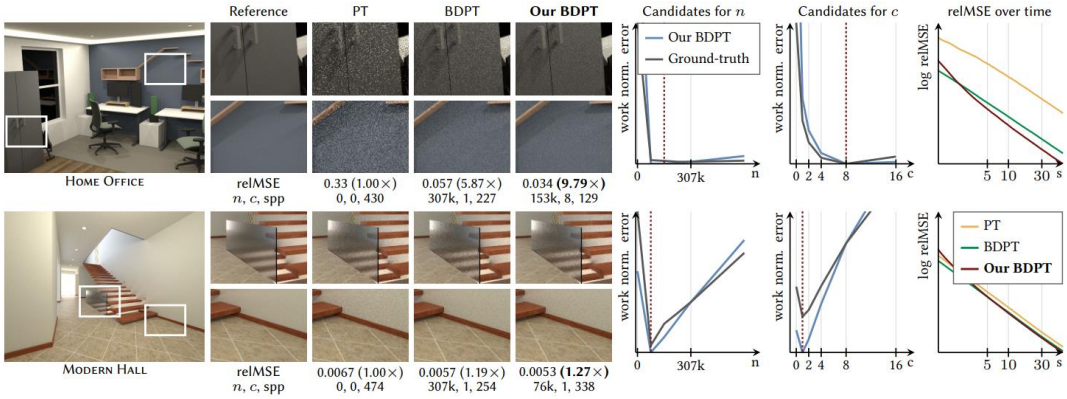


Fig. 7. Equal-time (60s) comparison between our optimized BDPT and two baselines. n and c are the number of light subpaths and bidirectional connections, respectively. The numbers in parentheses are the speed-ups (higher is better) over forward path tracing (PT). The plots compare our estimated work-normalized moments to ground truth work-normalized variances for different choices of n and c. The dashed red line marks the sample count chosen by our optimization. Basing our optimization on the second moments produces similar sample counts as the much more expensive full variances and yields consistent equal-time speed-ups compared to both baselines.

如图表十所示，运用了作者方法的渲染在不同场景下的渲染结果的噪声也要比不采用的要低，最坏持平。

文献参考

Eric Veach and Leonidas J Guibas. 1995a. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In SIGGRAPH '95. ACM, 419–428.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. ACM Trans. Graph. 31, 6 (2012), 10 pages.

Jaroslav Křivánek, Christophe Chevallier, Vladimir Koylazov, Ondřej Karlíik, Hen rik Wann Jensen, and Thomas Ludwig. 2018. Realistic Rendering in Architecture and Product Visualization. In ACM SIGGRAPH 2018 Courses. Article 10, 5 pages. https://doi.org/10.1145/3214834.3214872

Pascal Grittmann, Iliyan Georgiev, and Philipp Slusallek. 2021. Correlation-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms. Comput. Graph. Forum (EG 2021) 40, 2 (2021), 231–238.

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křiv á nek. 2019. Variance-Aware Multiple Importance Sampling. ACM Trans. Graph. (SIGGRAPH Asia 2019) 38, 6 (Nov. 2019), 9 pages.