

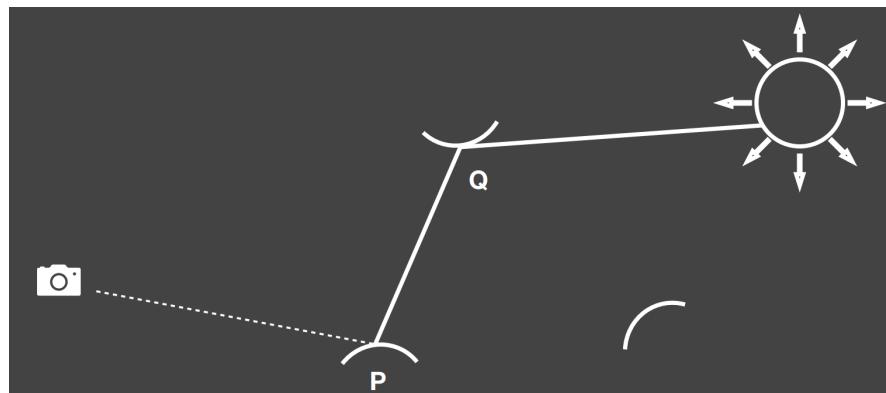
Real-Time Global Illumination

Mengzhu Wang

2024 年 5 月 13 日

1 Introduction

全局光照 = 直接光照 + 间接光照，是指通过模拟光线的传播路径，将物体反射的间接光也计算进去，从而提高真实感。实时渲染中，全局光照考虑直接光照和比直接光照多一次 bounce 的间接光照。



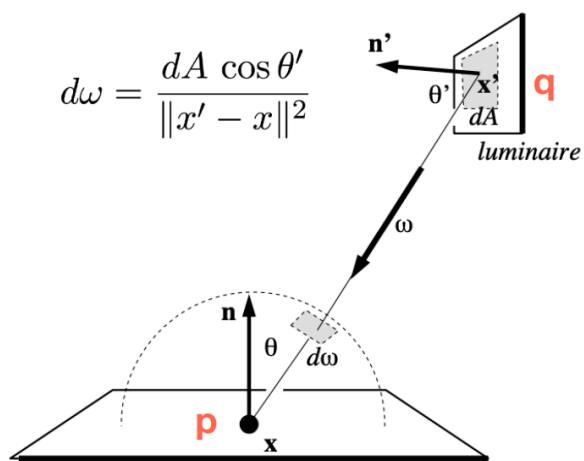
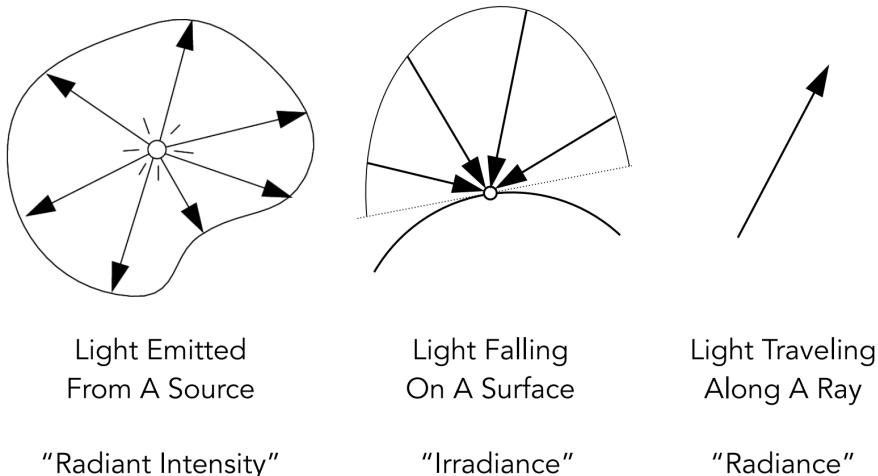
任意被主光源照亮的物体可以认为是次级光源，它们发射光线照亮其他物体。实现间接光照的关键问题在于：

- 寻找被直接光源照亮从而成为次级光源的物体表面
- 计算每个次级光源对 shading point 的光照贡献

2 Image Space

2.1 Reflective Shadow Maps (RSM)

RSM 基于 shadow map，把 shadow map 上的每个像素都看作一个小 surface patch，同时也是一個次级光源。接下来要用这些次级光源照亮点 p，也就是从点 p 观察次级光源，而非从相机观察，所以对于不同点 p 出射方向未知。假设 reflector 或称次级光源是 diffuse 的，出射光线在各方向是均匀的，这样从点 p 还是相机看过去都是一致的。



将所有 surface patch 对点 p 的贡献求和就得到间接光照。每一个 surface patch 可以当作 area light，计算 q 对 p 的贡献。直观来说可以对立体角积分，为了简化可以转换为对 light 面积的积分，它们有如下关系：

$$d\omega = \frac{dA \cos \theta'}{\|x' - x\|^2}$$

那么渲染方程也可以改写为

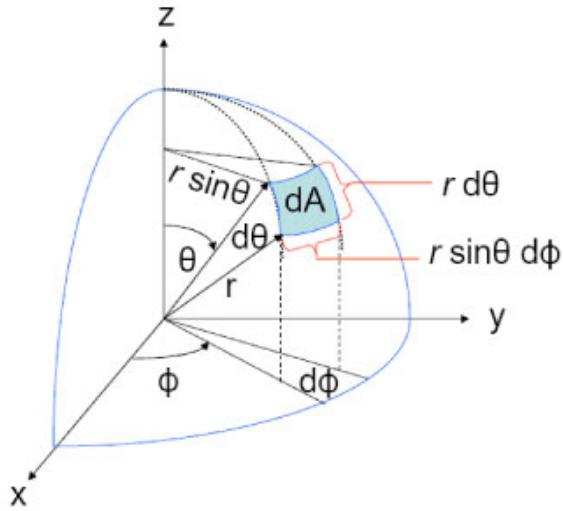
$$\begin{aligned} L_o(p, \omega_o) &= \int_{\Omega_{patch}} L_i(p, \omega_i) V(p, \omega_i) f_r(p, \omega_i, \omega_o) \cos \theta_i d\omega_i \\ &= \int_{A_{patch}} L_i(q \rightarrow p) V(p, \omega_i) f_r(p, q \rightarrow p, \omega_o) \frac{\cos \theta_p \cos \theta_q}{\|q - p\|^2} dA \end{aligned}$$

已经假设 q 是 diffuse 的，可得其为常数

$$f_r(q, \omega_i, \omega_o) = \frac{\rho}{\pi}$$

其中 ρ 表示反照率 albedo, 指辐射度 radiosity 和辐照度 irradiance 之比。该结论的推导如下:

$$\begin{aligned}\rho &= \int_{\Omega} f_r(x, \omega_i, \omega_o) \cos \theta_i d\omega_i \\ &= f_{lambert} \int_{\Omega} \cos \theta_i d\omega_i \\ &\rightarrow f_{lambert} = \frac{\rho}{\int_{\Omega} \cos \theta_i d\omega_i} \\ \int_{\Omega} \cos \theta_i d\omega_i &= \int_{\theta=0}^{\frac{\pi}{2}} \int_{\phi=0}^{2\pi} \cos \theta r d\theta r \sin \theta d\phi = r^2 \int_{\theta=0}^{\frac{\pi}{2}} \sin \theta \cos \theta d\theta \int_{\phi=0}^{2\pi} d\phi = \pi \\ f_{lambert} &= \frac{\rho}{\pi}\end{aligned}$$



BRDF 的定义为

$$f_r(q, \omega_i, \omega_o) = \frac{\text{Radiance}}{\text{Irradiance}} = \frac{dL_o(q, \omega_o)}{dE(q, \omega_i)}$$

则

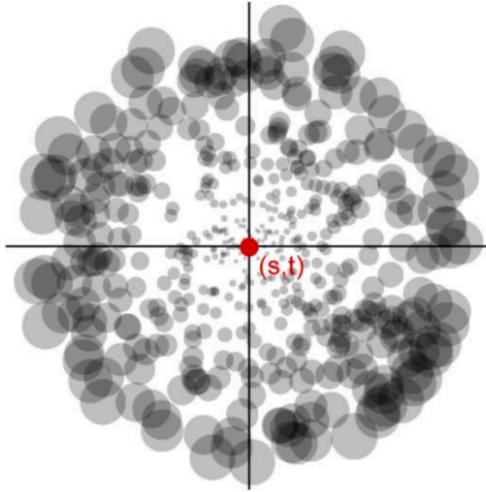
$$L_i(q \rightarrow p) = f_r \frac{\Phi}{dA}$$

其中 Φ 表示 incident flux 或 energy。

把上述 lighting 部分代入原来的渲染方程。需要注意的是 visibility 项，点 p 的间接光照来自很多个次级光源，每个都需要计算一次 shadow map 来判断是否遮挡，这是十分困难的，因此去掉该项，最终得到

$$\begin{aligned}E_p(q, n) &= \Phi \cdot f_r^2 \cdot \frac{\cos \theta_p \cos \theta_q}{\|q - p\|^2} \\ &= \Phi_p \cdot \frac{\cos \theta_p \cos \theta_q}{\|q - p\|^2} \\ &= \Phi_p \cdot \frac{\max\{0, n_p \cdot \frac{q-p}{\|q-p\|}\} \cdot \max\{0, n_q \cdot \frac{p-q}{\|p-q\|}\}}{\|q - p\|^2} \\ &= \Phi_p \cdot \frac{\max\{0, n_p \cdot (q-p)\} \cdot \max\{0, n_q \cdot (p-q)\}}{\|q - p\|^4}\end{aligned}$$

根据上述方法，计算场景中一点需要将 shadow map 中的每个像素都设为一个次级光源，这个数量将是十分庞大的；而且次级光源只能贡献到自身的法线平面，有一部分不会对着色点有贡献；此外很远的次级光源贡献也很小。综合以上问题，为加速 RSM，首先定位着色点在 shadow map 中的位置，多选取距离着色点近的次级光源，并且引入权重弥补稀疏采样的误差（权重近小远大，密小稀大），一般一个着色点采样 400 次。



RSM 需要存储深度 (shadow map)、世界坐标 (距离比较)、法线 (计算方向性) 和光源光强 (flux)。

优点：

- 易于实现（多用于手电筒效果）

缺点：

- 性能随光源数量增加而降低，因为需要存储更多的 shadow map
- 对于间接光照没有做可见性检查
- 假设次级光源是 diffuse 的，深度作为距离等
- 采样率和质量需要做平衡

3 3D Space

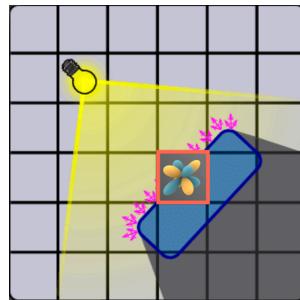
3.1 Light Propagation Volumes (LPV)

LPV 的核心思想是将场景划分为一系列三维网格（体素化）来模拟光线的传播，随后利用 radiance 在直线传播过程中保持恒定这一特征，计算出每个 shading point 上的间接光照。

LPV 的主要步骤如下：

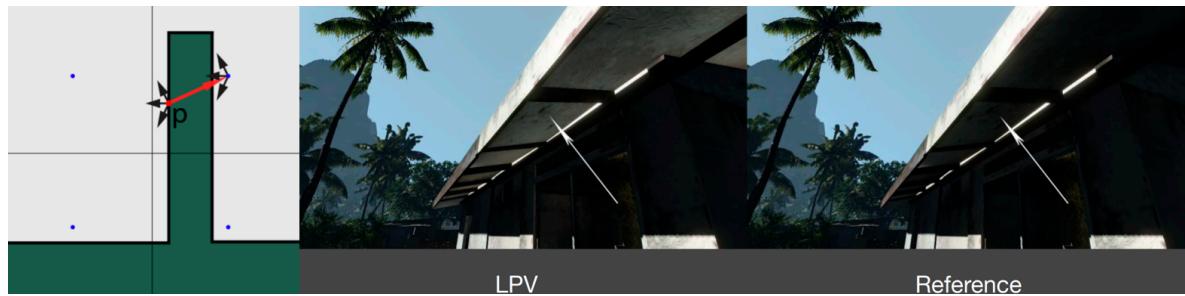
- 生成 (Generation)：采用 RSM 基于 shadow map 找到接受直接光照的点即次级光源，通过采样减少虚拟光源 (virtual light sources)。

- 注入 (Injection): 预先将场景划分为三维网格，找到每个网格中的虚拟光源，对其不同方向的 radiance 求和，投影到两阶 SH 上。



- 传播 (Propagation): 每个网格从周围的 6 个面上求和 radiance，并用 SH 表示，作为下一次迭代的定向 radiance 分布。重复该过程直至辐射亮度分布稳定。
- 渲染 (Rendering): 找到任意着色点所在的网格，根据网格所有方向的 radiance 计算结果。

LPV 可能存在漏光 (light leaking) 问题。当网格单元过大时，物体本身的粒度小于格网单元的粒度，而算法认为网格单元内部各点的辐射亮度分布是均匀的，于是从物体正面注入的辐射亮度可能会照亮本不该被照亮的景物背面。



体素划分至少比场景分辨率少一个数量级。LPV 使用了 SH 压缩替代，所以会假设次级光源都是 diffuse 的因为 glossy 用 SH 会产生强烈模糊。

3.2 Voxel Global Illumination (VXGI)

VXGI 和 RSM 一样是一个 2-pass 算法，能得到近似光线追踪的效果，但是速度较慢。

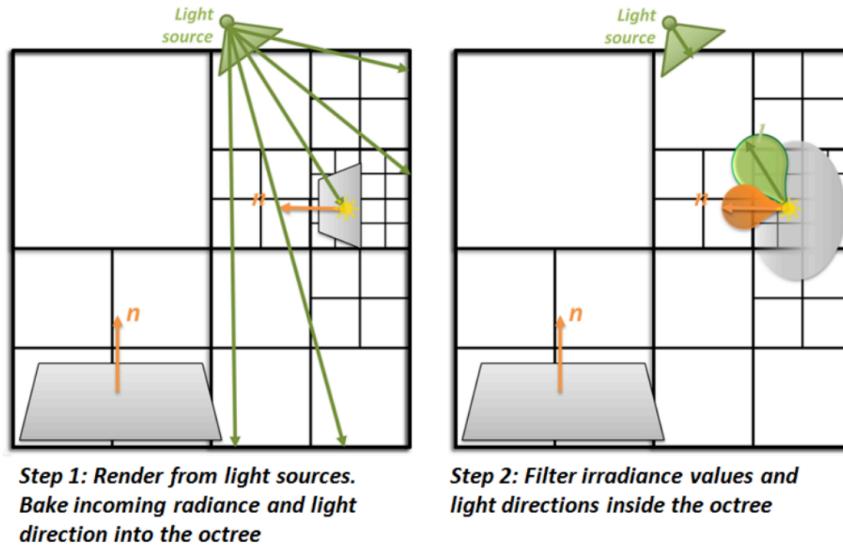
VXGI 与 RSM 的主要区别是

- RSM 的次级光源是 shadow map 中每个像素代表的 small surface patch，而 VXGI 的次级光源是离散化的网格，并划分为 hierarchical voxels 即稀疏八叉树结构。
- RSM 传播只做一次，LPV 的每个 shading point 只需要一次传播就能得出间接光照度的 radiance，而 VXGI 对每个像素要做 cone tracing。

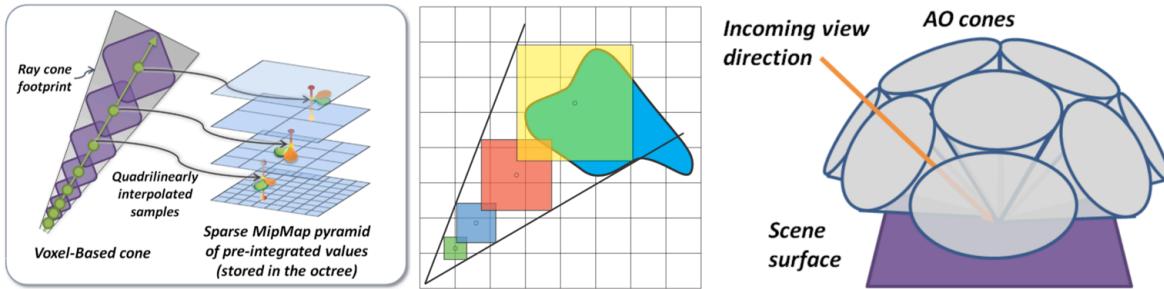
VXGI 的步骤为：

- Pass 1 (light pass): 记录直接光源的入射光线和法线的分布，结合材质计算间接光照分布，包括方向和锥角。这里同时支持 glossy 和 diffuse，比 LPV 仅支持 diffuse 更优。在层级结

构上，计算完低层级体素后，将八个叶子节点的结果相加就得到高一级的间接光照分布。



- Pass 2 (camera pass): 对于 glossy 材质，在反射方向追踪一个 cone，基于追踪出的圆锥面的大小，对其层级进行查询。对于 diffuse 材质，需要追踪若干个 cone。



4 Screen Space

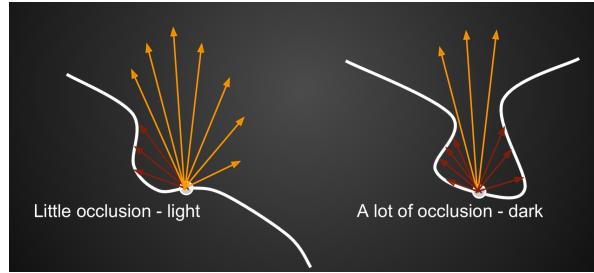
全局光照算法在生成间接光照时，对只有直接光照的渲染结果进行后处理以加上间接光照，这种方法称为屏幕空间算法。

4.1 Screen Space Ambient Occlusion (SSAO)

4.1.1 AO

环境光遮蔽 AO 是在屏幕空间中对全局光照的近似，能够使得距离较近的物体之间相互遮挡的暗部更加明显。

AO 假设任意 shading point 上来自任何方向的间接光照是常数，类似 Phong 模型的环境光。但是不同位置的 visibility 不同，且材质为 diffuse。



从渲染方程入手，根据近似公式 $\int_{\Omega} f(x)g(x)dx \approx \frac{\int_{\Omega_G} f(x)dx}{\int_{\Omega_G} dx} \cdot \int_{\Omega} g(x)dx$ 可以推得

$$L_o^{indir}(p, \omega_o) \approx \frac{\int_{\Omega^+} V(p, \omega_i) \cos \theta_i d\omega_i}{\int_{\Omega^+} \cos \theta_i d\omega_i} \cdot \int_{\Omega^+} L_i^{indir}(p, \omega_i) f_r(p, \omega_i, \omega_o) \cos \theta_i d\omega_i$$

其中前半部分记作 k_A ，表示当前 shading point 的 AO 在所有方向上以 \cos 为权重的加权平均。后半部分由于间接光照为常数且 BRDF 是 diffuse 的，对 AO 来说整个部分同样为常数。这两部分的推导如下：

$$k_A = \frac{\int_{\Omega^+} V(p, \omega_i) \cos \theta_i d\omega_i}{\pi}$$

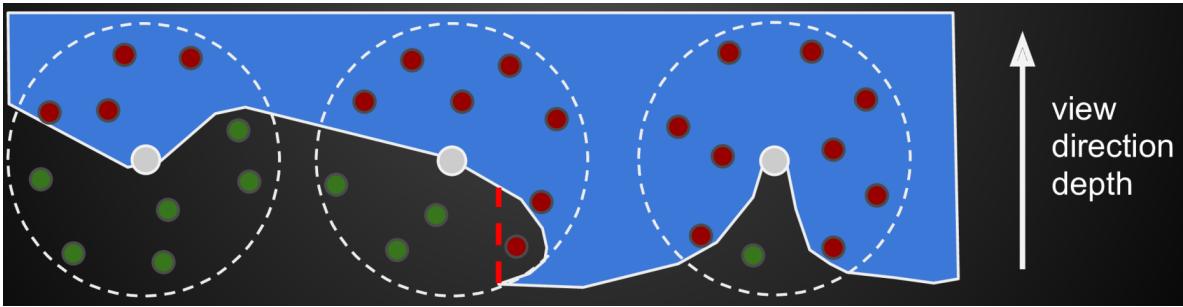
$$L_i^{indir}(p) \cdot \frac{\rho}{\pi} \cdot \pi = L_i^{indir}(p) \cdot \rho$$

对近似公式的一种理解是拆分出的项就是 $f(x)$ 在 $g(x)$ support 下的平均值，即 $\int_{\Omega} f(x)g(x)dx \approx \overline{f(x)} \cdot \int_{\Omega} g(x)dx$ 。该近似在 $g(x)$ 的 support 比较小或者 $g(x)$ 比较 smooth 的情况下比较准确，而 AO 的假设使得 $g(x)$ 是常数，因此是绝对准确的。

近似公式的另一种理解是关于 k_A 需要求一点向任意方向看去以 \cos 为权重的加权平均作为 visibility 的问题。此处 $\cos \theta d\omega$ 的意义是将单位球上对应的面积投影到半球对应的单位圆，这也解释了积分后得到 π 。

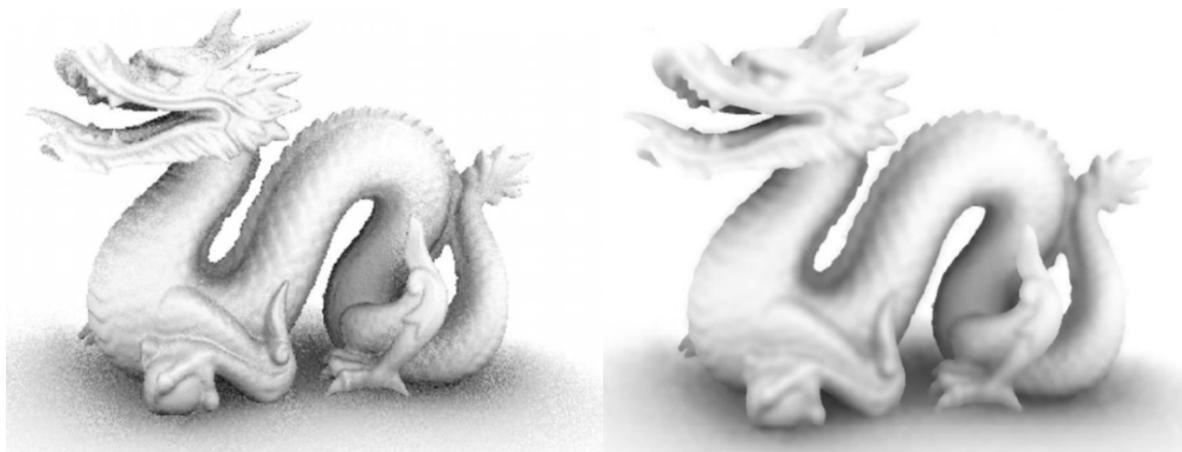
4.1.2 SSAO

剩下的问题就是计算 k_A 。SSAO 以 shading point 为中心，在 R 为半径的求体内随机采样。通过 zbuffer 判断采样点是否可见（可能误判）。AO 概念上是半球的遮挡，但是 SSAO 时代无法得到法线信息，所以考虑整个球体，计算可见点的概率，若大于 0.5 则不用 AO，否则使用 AO。也是因为没有考虑法线， \cos 加权也不考虑，虽然物理上不正确，但结果可行。



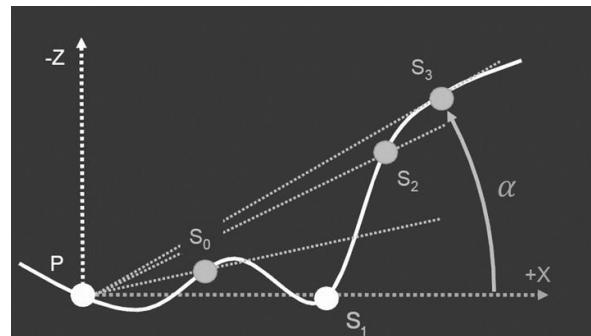
SSAO 由于深度的误判会产生多余的阴影。尽管更多的采样能得到更高的准确度，但是会影响性

能。采样较少的点得到一个 noisy 的结果，施加 edge preserving blur 可以降噪，得到一个较好的结果。

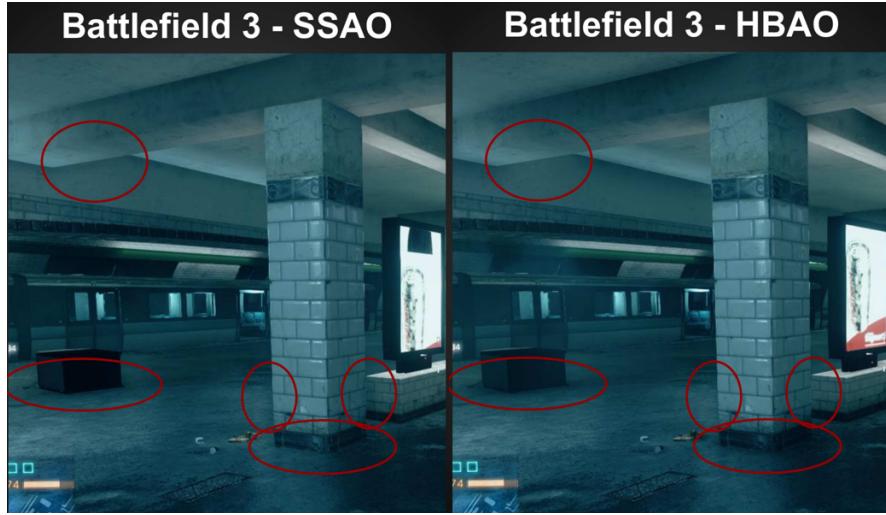


4.1.3 Horizon Based Ambient Occlusion (HBAO)

HBAO 已知法线信息，可以知道在哪个半球采样，使用多次光线步进找到着色点切平面与遮挡物采样点形成的最大角度，再通过这个角度来计算物体与物体间的遮挡程度，从而完成对 AO 的近似。

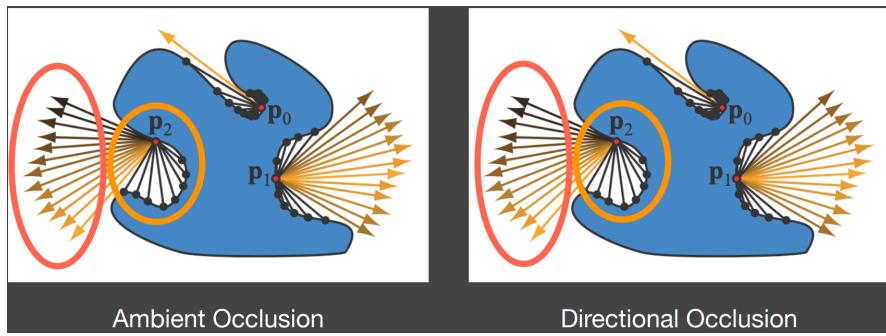


HBAO 有效减少了对遮挡的误判，改善阴影冗余问题，还大大减少采样。



4.2 Screen Space Directional Occlusion (SSDO)

SSDO 类似 path tracing，从 shading point 发射光线，没有遮挡则是直接光照，有遮挡则是间接光照。SSAO 假设间接光照来自远方且恒定，会被 shading point 附近的物体遮挡。相反，SSDO 假设间接光照来自附近，只有被遮挡了该遮挡物才能作为次级光源实现间接光照。SSDO 产生不恒定的间接光照，存在 color bleeding 的效果，使得全局光照更加真实。

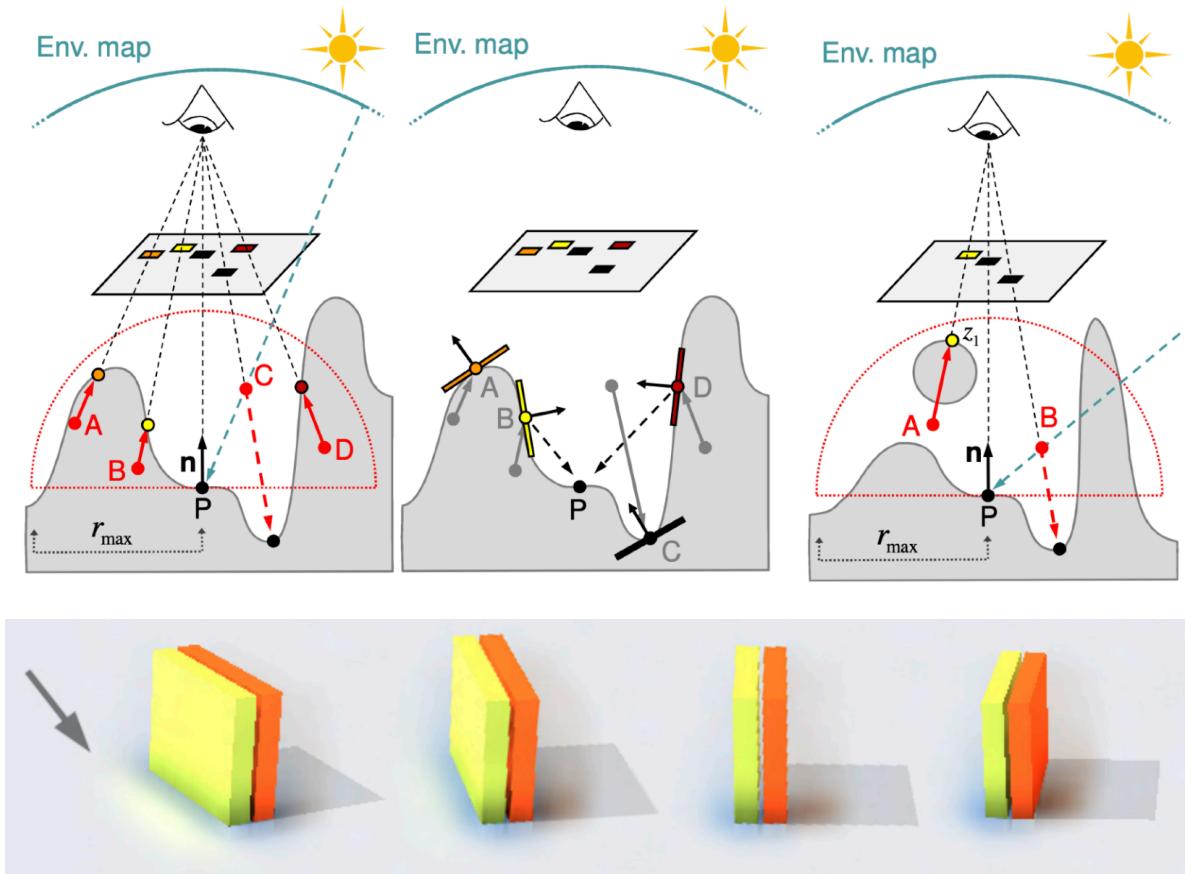


SSDO 在点 p 的半球内采样若干点，判断相机视角下的可见性，被遮挡的点称为潜在的次级光源，然后根据它们的法线方向等信息，判断它们是否对点 p 贡献间接光照并累加贡献。不过也存在误判的情况，比如右图 SSDO 判断 A 对应潜在次级光源，但实际上它是直接光源。

SSDO 的质量接近离线渲染，但它只在 shading point 附近采样，只考虑小范围的全局光照，且由于屏幕空间的限制会丢失信息。

4.3 Screen Space Reflection (SSR)

SSR 是在屏幕空间做光线追踪，光线与相机看到的一层场景求交，找到交点后计算对 shading point 的贡献。

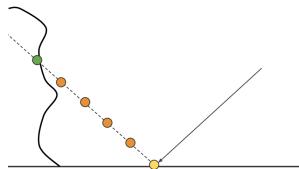


SSR 算法做镜面反射对于每个 fragment 的步骤是

- 计算反射光线
- 用 depth buffer 沿着光线方向追踪，步进查找深度直至相交
- 用交点颜色作为反射颜色

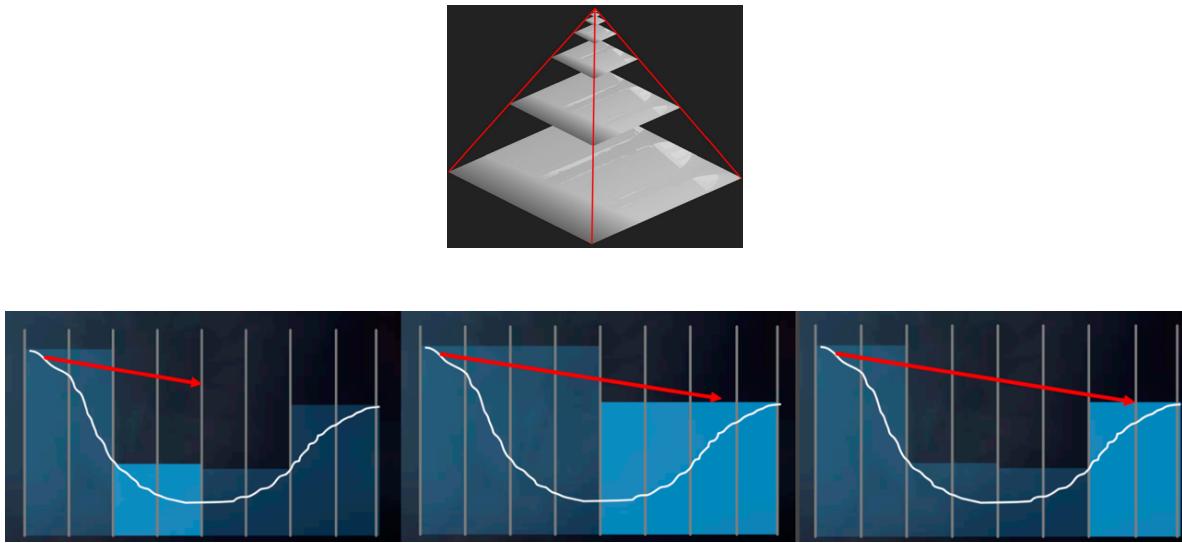
4.3.1 Intersection

求反射光线与场景的相交是其中最为关键的问题之一。最基础的方法是线性光线步进(Linear Ray-march)。该方法沿着反射方向以固定的步长前进，每次都将当前深度和场景的深度比较。若当前深度更浅，则继续前进，否则停止。这种方法的质量依赖于步长，步长越小越精确，但也导致计算量增大。



层级式光线追踪 (Hierarchical ray trace) 是上述方法的改进。首先将场景的深度图做 mipmap，高一级的 mipmap 存储低一级周围四个深度中的最小值。这样光线与 mipmap 中高级的节点不相

交时，必然不会与其子节点相交。开始步长较小，然后逐步试探。若没有交点，就增大步长；若出现交点，退回更低的层级，直至出现交点停止或始终没有交点再停止。



由于屏幕空间的限制，会导致 hidden geometry problem, edge cutoff (edge fading 改善) 等问题。



4.3.2 Shading

SSR 的 shading 与 path tracing 无异，对 shading point 对应的半球积分即可。SSR 不用考虑光线的平方衰减，并且可以保证交点的可见性，这是因为 SSR 的采样对象是 BRDF 本身，而只有在对光源采样的时候才要考虑衰减和遮挡。SSR 可以实现 sharp and blurry reflections、contact hardening、specular elongation、per-pixel roughness and normal 等许多效果。

4.3.3 Summary

优点：

- glossy 和 specular 反射效果很好很快
- 没有 spike 和 occlusion 问题

缺点：

- 对于 diffuse 不高效

- 丢失屏幕外的信息