

AO

1 Ambient Occlusion (环境光遮蔽)

环境光遮蔽 (Ambient Occlusion, AO) 是一种用于模拟光线在场景中传播时的遮蔽效果的技术。它通过计算场景中每个点被周围几何体遮挡的程度，从而为渲染图像添加深度和细节感。

2 SSAO

SSAO (Screen Space Ambient Occlusion) 是一种基于屏幕空间的环境光遮蔽技术。它通过在屏幕空间内计算每个像素的遮蔽程度来模拟环境光遮蔽效果。SSAO 通常使用深度缓冲区和法线缓冲区来计算每个像素周围的几何体遮挡情况。

SSAO 的基本步骤如下：

- (1) 根据 GBuffer 中的深度信息，计算每个像素的世界空间位置。
- (2) 使用 GBuffer 中的法线信息以及随机单位向量，在着色点的法向半球内进行采样，计算遮蔽值（范围为 $[0, 1]$ ）。

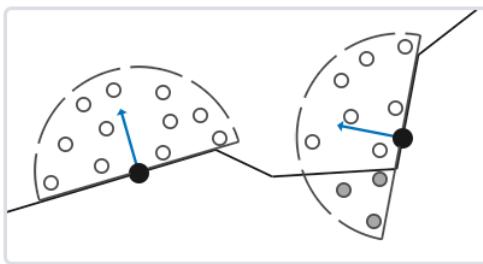


图 1: SSAO 的工作原理示意图

2.1 Unity URP 的 SSAO 简述

SSAO 设置见环境光遮挡 (Ambient Occlusion)，此处不做多余描述。

完整的管线主要包含两部分，第一部分为计算 AO，第二部分为 blur 操作。

注：代码实现还包含对 VR/XR 的适配，如注意力渲染等技术，此处不做多余描述

2.1.1 计算 AO

(1) 根据 uv 坐标采样深度，如果过小，返回 AO 的结果为 0；将深度转化为半精度的线性深度，若大于自定义的 Falloff，返回 AO 的结果为 0

(2) 生成法线

(2.1) depthNormal：用 uv 采样 depthNormal 通道生成的法线

(2.2) depth：用深度重建法线

(a) 根据屏幕空间坐标 uv 和观察空间深度 linearDepth 重建 viewPos

(b) 根据 uv、linearDepth 和 viewPos 重建法线，具体方法见参考链接

(3) 采样计算 AO

(a) 采样方向向量（单位向量乘以半径），分为两种方法：一种根据 blue noise 生成，另一种根据内置的 InterleavedGradientNoise 生成

(b) 计算采样点的 viewPos，计算对应的 screenPos（包含正交相机和透视相机的不同处理），根据深度比较判是否在指定半径内或者是否采样到天空

(c) 根据屏幕坐标和线性空间深度重建观察空间坐标，根据法线计算 AO 值

(d) 在 AO 结果应用 falloff、intensity 和 contrast

2.1.2 blur

主要分为三种类型：

(1) Bilateral Blur，对应 Blur Quality High

(1.1) Bilateral Horizontal Blur

双边过滤的权重，是根据对应法线的点乘结果作为插值因子，在 [0.8, 1.0] 内进行插值计算

(1.2) Bilateral Vertical Blur

(1.3) Final Blur

用更小的 kernel 再进行一次过滤，返回 1.0 - AO 的结果

(2) Gaussian Blur, 对应 Blur Quality Medium

(2.1) Gaussian Horizontal Blur

用 7x7 的 gaussian kernel 进行滤波，使用的优化算法见参考链接，每次调用对应的循环只需要两次，极大减少了计算次数

(2.2) Gaussian Vertical Blur

(3) Kawase Blur, 对应 Blur Quality Low

对左上角、右上角、左下角、右下角进行 Kawase Blur（取平均值）

注：每种类型都会区分是否有 AfterOpaque，如果有，会将 ao 结果存储在 alpha 通道中；如果没有，则将 ao 结果存储在 R 通道中。

2.1.3 参考

Accurate Normal Reconstruction from Depth Buffer

Improved normal reconstruction from depth

An Investigation Of Fast Real-Time GPU-based Image Blur Algorithms

3 HBAO

HBAO (Horizon-based Ambient Occlusion) 是一种基于地平线的环境光遮蔽技术，它通过计算每个像素周围的地平线遮挡情况来模拟环境光遮蔽效果。

HBAO 的基本步骤如下：

(1) 对深度缓冲进行采样，每个 pixel 固定采样数量（4 或 8），采样方向进行随机旋转或者添加随机 offset，如图2所示。

(2) 计算 horizon angle 和 tangent angle，如图3和图4所示。

(3) 计算 AO，如图5所示。

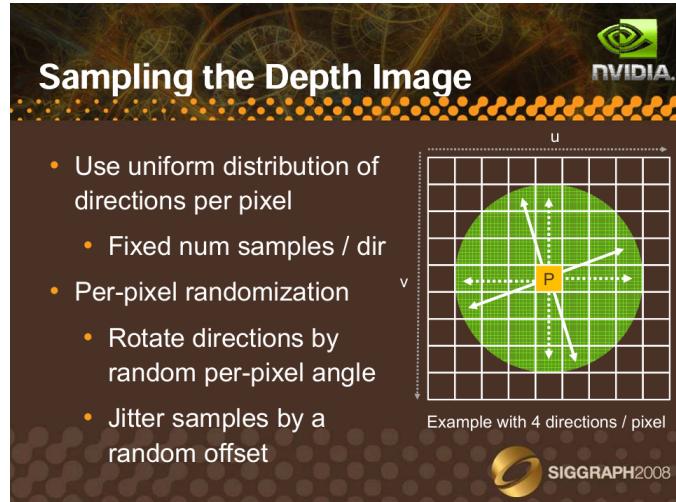


图 2: HBAO 的深度采样

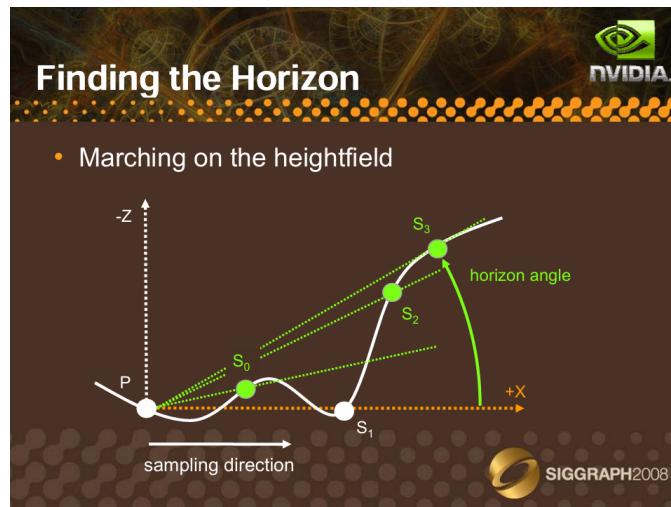


图 3: 计算 horizon angle

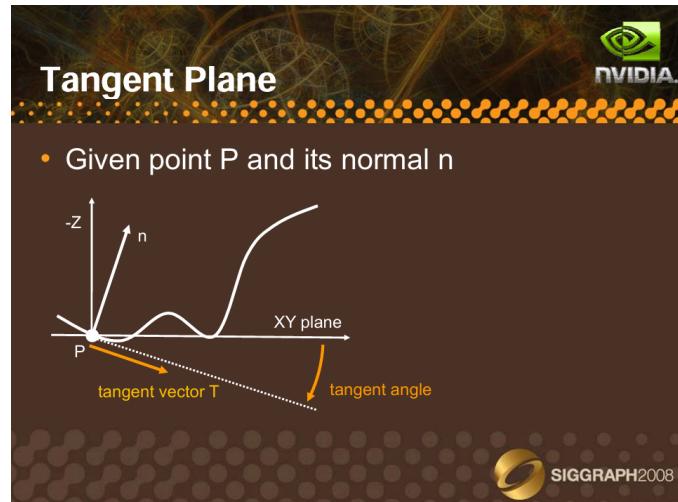


图 4: 计算 tangent angle

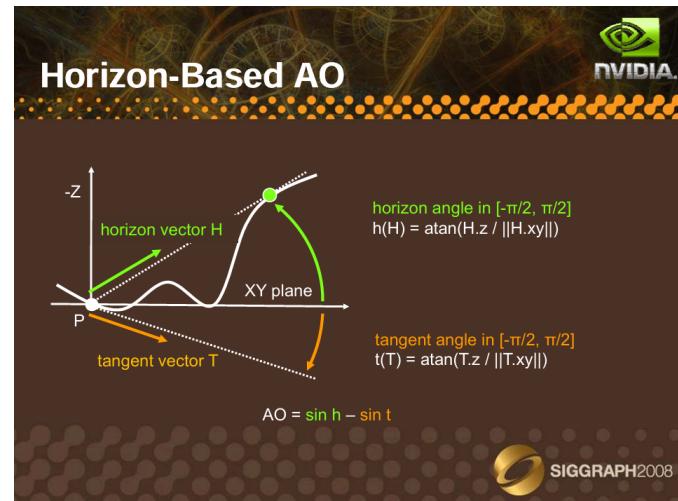


图 5: 计算 AO

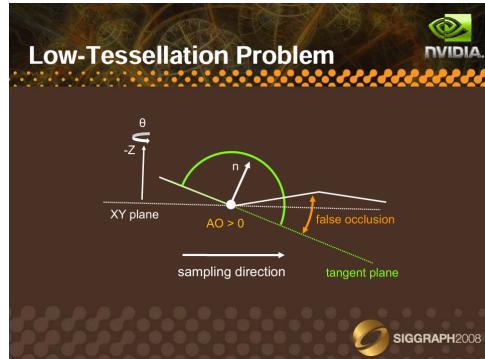


图 6: Low-Tessellation Problem

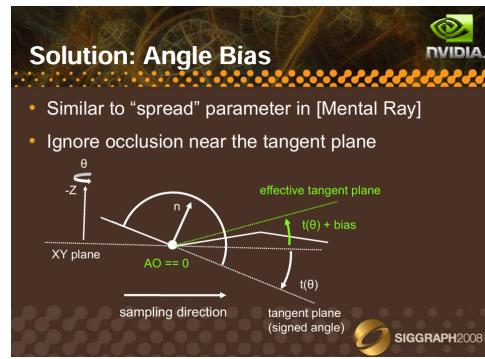


图 7: Angle Bias

3.1 Low-Tessellation Problem

如图6所示，会出现错误遮挡的情况。可以通过添加 Angle Bias 来解决，如图7所示。

3.2 Discontinuity Problem

如图8所示，相邻像素的 AO 采样可能会带来不连续的问题。如图9和图10所示，HBAO 通过衰减操作来解决这个问题。

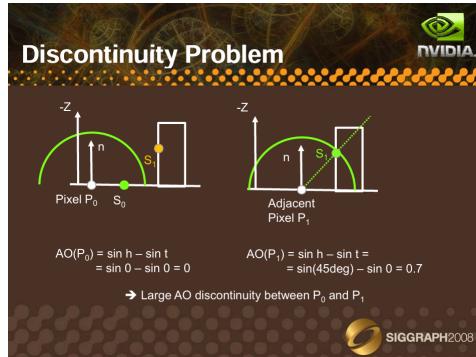


图 8: Discontinuity Problem

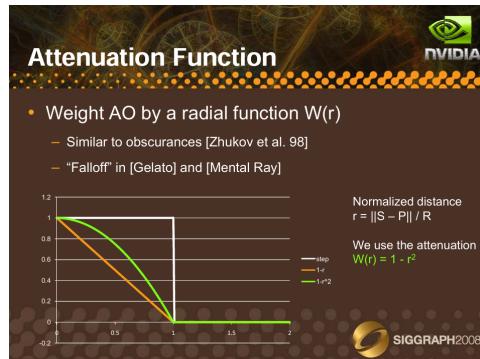


图 9: Attenuation Function

3.3 Noise

AO 会带来噪声问题，如图11所示，通过添加双边滤波或基于深度的高斯滤波来处理。

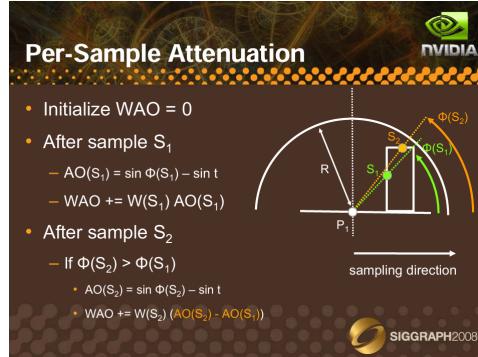


图 10: Per-Sample Attenuation



图 11: Cross Bilateral Filter

4 GTAO

GTAO (Ground Truth Ambient Occlusion) 同样也是基于地平线的方法，但是与 HBAO 相比，GTAO 更加准确。

4.1 算法步骤

(1) 在一个半圆的 slice 中，如图12所示，每个 slice 对应屏幕空间的前进方向，记为 ϕ ，分别沿着 $+\phi$ 和 $-\phi$ 的方向前进，计算两个最大的 horizon angle，记为 $\theta_1(\phi)$ 和 $\theta_2(\phi)$ 。

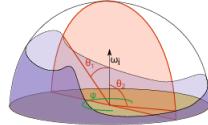


图 12: GTAO Slice

(2) 计算 AO

AO 的计算公式如下：

$$\hat{A}(x) = \frac{1}{\pi} \int_0^{\pi} \int_{\theta_1(\phi)}^{\theta_2(\phi)} \cos(\theta - \gamma)^+ |\sin(\theta)| d\theta d\phi$$

其中 γ 是法线和观察方向的夹角， $\cos(\theta)^+ = \max(\cos(\theta), 0)$ 。

上式中内积分，通过以下方式求解析解：

$$\begin{aligned} \hat{a} &= \frac{1}{4} (-\cos(2\theta_1 - \gamma) + \cos(\gamma) + 2\theta_1 \sin(\gamma)) \\ &\quad + \frac{1}{4} (-\cos(2\theta_2 - \gamma) + \cos(\gamma) + 2\theta_2 \sin(\gamma)) \end{aligned}$$

注：

该公式要求该处的法线位于定义 slice 所处的平面（记为 P ），大多数情况下并不满足，所以我们需要计算法线在该平面上的投影，记为 $\hat{\mathbf{n}}_x$ 。 γ 计算方法如下：

$$\gamma = \arccos\left(\frac{\hat{\mathbf{n}}_x}{\|\hat{\mathbf{n}}_x\|}, \omega_o\right)$$

其中 ω_o 为观察方向，对应的 AO 计算结果需要乘以 $\hat{\mathbf{n}}_x$ 的范数：

$$\hat{A}(x) = \frac{1}{\pi} \int_0^\pi \|\hat{\mathbf{n}}_x\| \hat{a}(\phi) d\phi$$

(3) 双边过滤

4.2 HDRP 中的 GTAO

- (1) 获取当前像素的 depth、viewPos、normal，计算最大步长。
- (2) horizon loop，分为两种模式
 - (2.1) 时序 (temporal) 模式，每次只计算一个方向
 - (2.2) 全向 (full-directional) 模式，每次计算所有方向
- 根据最大步长和前进方向，每次计算两个相反方向的最大 horizon angle。
- (3) 计算投影法线， γ 和 $\cos\gamma$ ，依据之间的公式计算 AO 值。

5 CACAO(Combined Adaptive Compute Ambient Occlusion)

AMD 的 FidelityFx 的 SDK 中提供了 CACAO (Combined Adaptive Compute Ambient Occlusion) 的 AO 算法，是 Intel 提出的 ASSAO (Adaptive Screen Space Ambient Occlusion) 的扩展版本。

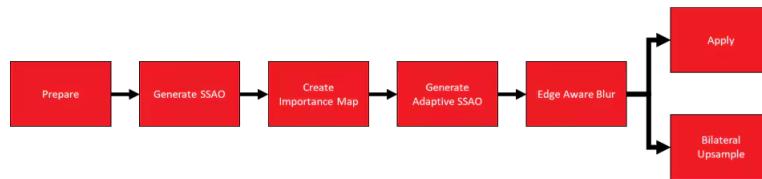


图 13: CACAO

5.1 准备阶段

准备 depth buffer 和 normal buffer，depth buffer 根据质量来选择是否生成 mipmap；除此之外，会进行 de-interleaving 操作。

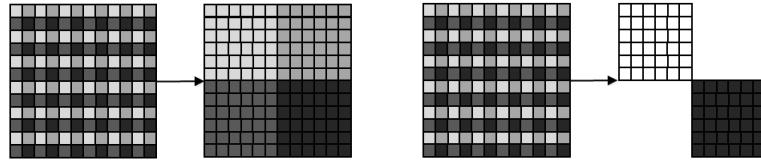


图 14: De-interleaving

如图14所示，每 2×2 的像素块，会被拆分到四个纹理中，每个纹理的分辨率只有原始分辨率的一半；如果选择低质量模式，会抛弃 50% 的像素块。这一操作时为了提高 GPU 上缓存结构的效率。

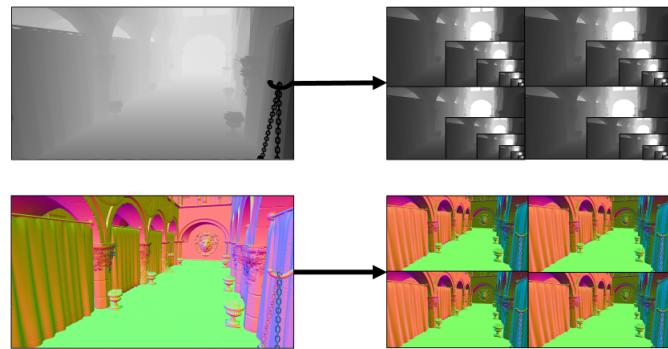


图 15: Depth Buffer and Normal Buffer

5.2 生成 SSAO

计算公式如下：

$$obscurance = \max(0, (\frac{\mathbf{n} \cdot (\mathbf{q} - \mathbf{p})}{|\mathbf{q} - \mathbf{p}|}) - h_{at}) \times \max(0, 1 - f_c \times |\mathbf{q} - \mathbf{p}|^2)$$

5.3 生成 adaptive SSAO

(1) 生成 Importance Map

如图16，计算 8×8 的方块中的 AO 的最大值和最小值，在 Importance Map 中存储最大值和最小值的差，并进行 blur，避免重要区域过渡到非重要区域的的尖锐边缘问题。

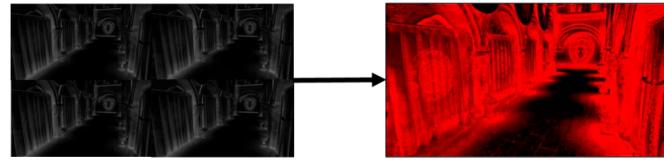


图 16: 生成 Importance Map

(2) 计算 adaptive AO

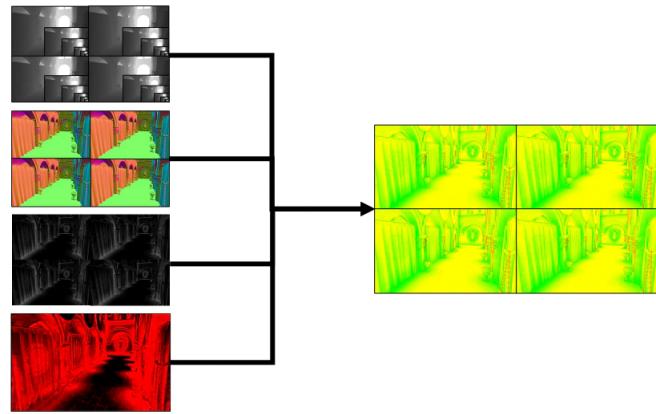


图 17: Adaptive Sample

如图17所示，再进行一次 SSAO 的过程，但是样本的选择基于 Importance Map，计算得到的 AO 和第一次计算得到的 SSAO 进行加权混合，得到最终的 AO。

5.4 Edge-aware blur

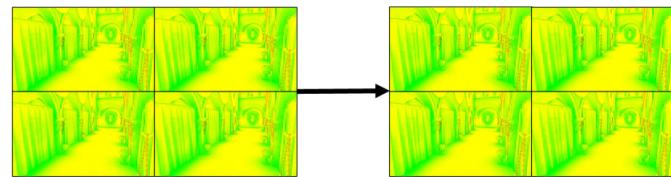


图 18: Edge-aware blur

如图18所示，使用 3×3 的 kernel 对 AO 结果进行降噪。

6 参考

LearnOpenGL - SSAO
HBAO(屏幕空间的环境光遮蔽)
HBAO
GTAO
FidelityFX-SDK
ASSAO