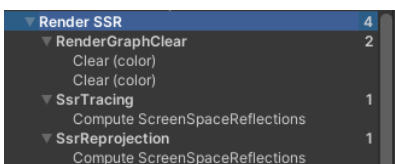


材质

1 Screen Space Reflection (SSR)

以 HDRP 的 SSR 实现为例，其主要流程如下所示：



Render SSR	4
RenderGraphClear	2
Clear (color)	
Clear (color)	
SsrTracing	1
Compute ScreenSpaceReflections	
SsrReprojection	1
Compute ScreenSpaceReflections	

图 1: HDRP Screen Space Reflection

1.1 RenderGraph Clear

清空上一帧 SSR 的计算结果。

1.2 SSR Tracing

1.2.1 计算反射方向

代码中分两种策略：

- (1) 根据相机的观察方向和 GBuffer 的法线计算反射方向。
- (2) 利用 GGX 的 BRDF 计算反射方向（考虑材质的 roughness），此处 GGX 分为 Visible GGX 和普通的 GGX。

1.2.2 Ray March

反射方向作为光线的前进方向，后续操作和 HDRP 的 SSGI 类似，存储 hit point 的信息，此处不做重复描述，见 GI.pdf。

1.3 SSR Reprojection

(1) 获取 hit point 和 motion vector, 计算上一帧 hit point 对应位置。

(2) 计算 opacity

主要是计算一个衰减因子:

(a) 屏幕边缘的衰减 (屏幕空间技术常见操作)

(b) 基于 perceptual roughness 进行衰减

```
1 float PerceptualRoughnessFade(float perceptualRoughness ,
2     float fadeRcpLength, float fadeEndTimesRcpLength)
3 {
4     float t = Remap10(perceptualRoughness , fadeRcpLength ,
5         fadeEndTimesRcpLength);
6     return Smoothstep01(t);
7 }
8
9 // Performs fading at the edge of the screen.
10 float EdgeOfScreenFade(float2 coordNDC, float fadeRcpLength)
11 {
12     float2 coordCS = coordNDC * 2 - 1;
13     float2 t = Remap10(abs(coordCS), fadeRcpLength, fadeRcpLength);
14     return Smoothstep01(t.x) * Smoothstep01(t.y);
15 }
16
17 float opacity = EdgeOfScreenFade(prevFrameNDC, _SsrEdgeFadeRcpLength)
18     * PerceptualRoughnessFade(perceptualRoughness ,
19     _SsrRoughnessFadeRcpLength, _SsrRoughnessFadeEndTimesRcpLength);
```

(3) 采样 color

这里分为两个策略, 以宏 `SSR_APPROX` 进行区分

(3.1) 应用宏 `SSR_APPROX`

对 color 进行采样, 将 color 和 opacity 的乘积作为结果进行存储, 其中采样 color 的 mipmap level 策略如下:

```
1 // TODO: filtering is quite awful. Needs to be non-Gaussian,
2 //     bilateral and anisotropic.
```

```
3 float mipLevel = lerp(0, _SsrColorPyramidMaxMip, perceptualRoughness);
```

(3.2) 不应用宏 *SSR_APPROX*

对周围 3×3 的领域进行加权求和，对每个像素计算 color、opacity 和 opacity，此处 color 采样的 mipmap level 固定为 0。

计算权重的公式如下：

```
1 // Weight for SSR where Fresnel == 1 (returns value/pdf)
2 float GetSSRSampleWeight(float3 V, float3 L, float roughness)
3 {
4     // Simplification:
5     // value = D_GGX / (lambdaVPlusOne + lambdaL);
6     // pdf = D_GGX / lambdaVPlusOne;
7
8     const float lambdaVPlusOne = Lambda_GGX(roughness, V) + 1.0;
9     const float lambdaL = Lambda_GGX(roughness, L);
10
11     return lambdaVPlusOne / (lambdaVPlusOne + lambdaL);
12 }
```

1.4 参考

Screen Space Reflection

2 Visible GGX

2.1 前置说明

GGX 是基于 Microfacet 理论的 BRDF 模型，在常见渲染器和引擎中有广泛的应用（Unity、UE、Blender、PBRT）。最初 GGX 的法线分布函数（Normal Distribution Function, NDF）在 NEE（Next Event Estimation）仅考虑 roughness 和法线对出射方向进行采样，去寻找光源。但是，不是所有光照方向都会对当前观察方向产生贡献，所以我们要基于观察方向进行重要性采样，减少无效采样，因此引入了 Visible GGX。

2.2 GGX 基础知识

详情见pbrtv4。

2.3 Ellipsoid VNDF

2.3.1 GGX 分布

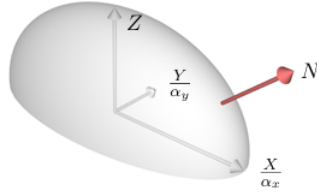


Figure 2. The GGX distribution is the distribution of normals of an ellipsoid.

图 2: GGX Distribution

如图所示2所示，GGX 分布在局部空间表现为椭球的上半部分，其由

$$\begin{aligned} Z &= (0, 0, 1) \\ \frac{Y}{\alpha_y} &= \frac{(0, 1, 0)}{\alpha_y} \\ \frac{X}{\alpha_x} &= \frac{(1, 0, 0)}{\alpha_x} \end{aligned}$$

构成，其中 α_x 和 α_y 分别表示 x 方向和 y 方向的粗糙度 (roughness)，用于表示各项异性效果。给定法线 $N = (x_n, y_n, z_n)$ ，NDF 的计算结果为

$$D(N) = \frac{1}{\pi \alpha_x \alpha_y \left(\frac{x_n^2}{\alpha_x^2} + \frac{y_n^2}{\alpha_y^2} + z_n^2 \right)^2}.$$

给观察方向 $V = (x_v, y_v, z_v)$ ，对应的 Smith 各向异性 GGX 遮挡函数如下：

$$G_1 = \frac{1}{1 + \Lambda(V)}, \text{ with } \Lambda(V) = \frac{-1 + \sqrt{1 + \frac{\alpha_x^2 x_v^2 + \alpha_y^2 y_v^2}{z_v^2}}}{2}.$$

2.3.2 VNDF

D 项计算公式如下：

$$D_V(N) = \frac{G_1(V) \max(0, V \cdot N) D(N)}{V \cdot Z}$$

PDF 计算公式如下：

$$PDF = \frac{D_V(N)}{4(V \cdot N)}$$

2.3.3 采样 VNDF

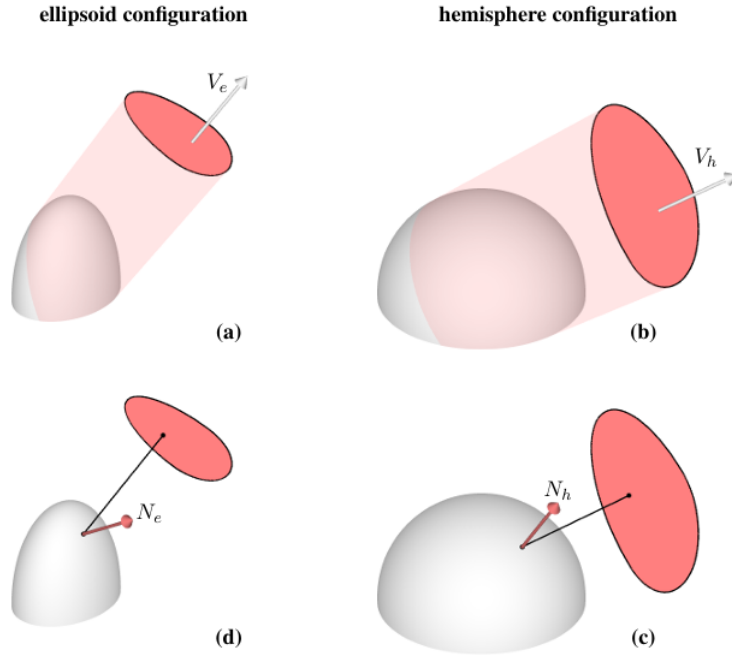


图 3: Ellipsoid-hemisphere transformation

如图3所示，分为四步：

(a) 初始化局部椭球参数，观察方向 V_e ，粗糙度参数 α_x 、 α_y ，均匀随机数 u_1 和 u_2 。

(b) 将观察方向映射到半球面

$$A = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V_h = \frac{AV_e}{\|AV_e\|}$$

(c) 在半球上的投影面积采样法线 N_h

(d) 将半球上采样的法线映射到半椭球体上

$$N_e = \frac{AN_h}{\|AN_h\|}$$

2.3.4 在半球上的投影面积采样法线

(1) 构建正交基 (V_h, T_1, T_2)

$$T_1 = \frac{Z \times V_h}{\|Z \times V_h\|} = \frac{(-y_v, x_v, 0)}{\sqrt{x_v^2 + y_v^2}},$$

$$T_2 = V_h \times T_1$$

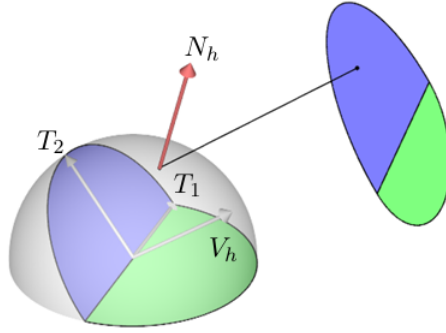


Figure 4. Orthonormal basis for sampling the projected area of the hemisphere.

图 4: Orithogonal Basis

(2) 对投影面积进行参数化表示

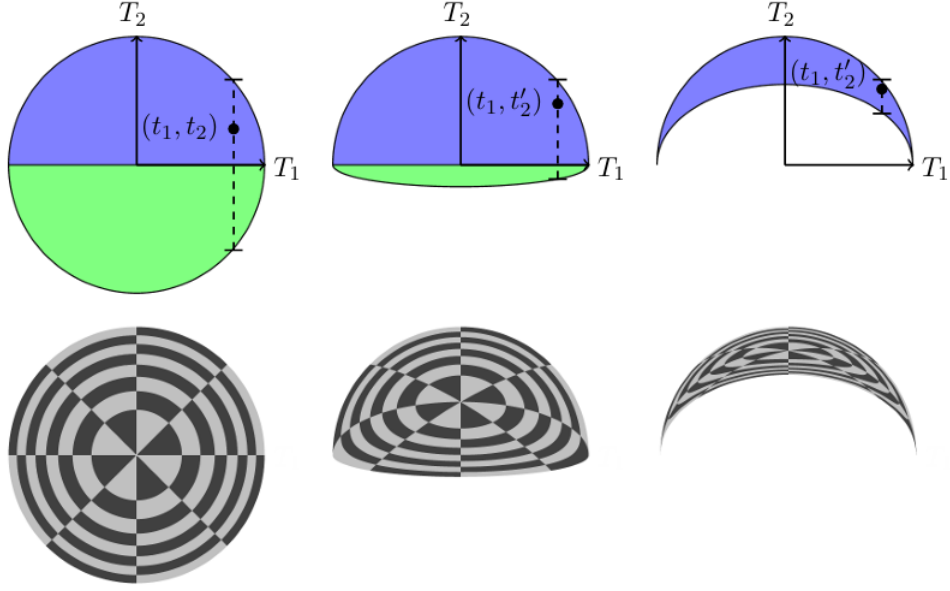


图 5: Parameterization of the projected area of the hemisphere

如图5所示,水平方向取 t_1 ,垂直方向对应的区间为 $[-\sqrt{1-t_1^2}, \sqrt{1-t_1^2}]$, 需要将其映射到区间 $[-(V_h \cdot Z)\sqrt{1-t_1^2}, \sqrt{1-t_1^2}]$, 对应的放缩因子为 $s = \frac{1+(V_h \cdot Z)}{2}$ 。

对单位圆盘上均匀分布的点 (t_1, t_2) , 用极坐标表示:

$$(r, \phi) = (\sqrt{U_1}, 2\pi U_2)$$

$$(t_1, t_2) = (r \cos(\phi), r \sin(\phi))$$

运用下述垂直映射, 将其映射到投影区域内的点 (t_1, t_2') :

$$t_2' = (1-s)\sqrt{1-t_1^2} + s\sqrt{t_2}$$

(3) 重投影到半球上

将点 $t_1 T_1 + t_2 T_2$ 重投影到半球上, 单位化得到法线:

$$N_h = t_1 T_1 + t_2 T_2 + v_h V_h$$

其中

$$v_h = \sqrt{1 - t_1^2 - t_2^2}$$

2.3.5 代码

```

1 // Input Ve: view direction
2 // Input alpha_x, alpha_y: roughness parameters
3 // Input U1, U2: uniform random numbers
4 // Output Ne: normal sampled with NDF
5 //      D_Ve(Ne) = G1(Ve) * max(0, dot(Ve, Ne)) * D(Ne) / Ve.z
6 vec3 sampleGGXVNDF(vec3 Ve, float alpha_x, float alpha_y,
7     float U1, float U2)
8 {
9     // transforming the view direction to the hemisphere configuration
10    vec3 Vh = normalize(vec3(alpha_x * Ve.x, alpha_y * Ve.y, Ve.z));
11    // orthonormal basis (with special case if cross product is zero)
12    float lensq = Vh.x * Vh.x + Vh.y * Vh.y;
13    vec3 T1 = lensq > 0 ?
14        vec3(-Vh.y, Vh.x, 0) * inversesqrt(lensq) : vec3(1,0,0);
15    vec3 T2 = cross(Vh, T1);
16    // parameterization of the projected area
17    float r = sqrt(U1);
18    float phi = 2.0 * M_PI * U2;
19    float t1 = r * cos(phi);
20    float t2 = r * sin(phi);
21    float s = 0.5 * (1.0 + Vh.z);
22    t2 = (1.0 - s) * sqrt(1.0 - t1*t1) + s*t2;
23    // Section 4.3: reprojection onto hemisphere
24    vec3 Nh = t1*T1 + t2*T2 + sqrt(max(0.0, 1.0 - t1*t1 - t2*t2))*Vh;
25    // transforming the normal back to the ellipsoid configuration
26    vec3 Ne = normalize(vec3(alpha_x * Nh.x, alpha_y * Nh.y,
27        std::max<float>(0.0, Nh.z)));
28    return Ne;
29 }

```


2.4 Spherical Caps VNDF

2.4.1 采样步骤

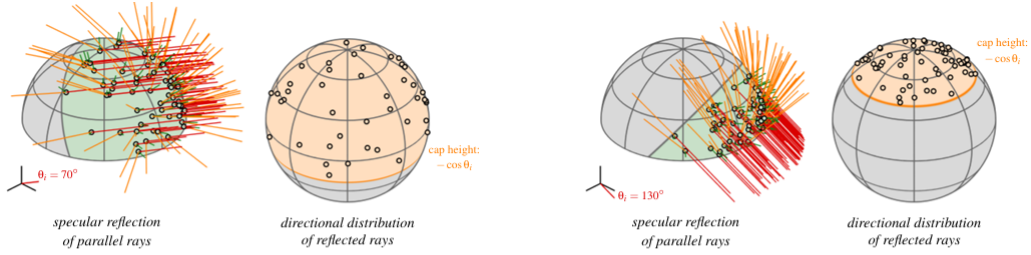


Figure 4: Main insight: a hemispherical mirror reflects parallel light rays towards directions enclosed within a spherical cap.

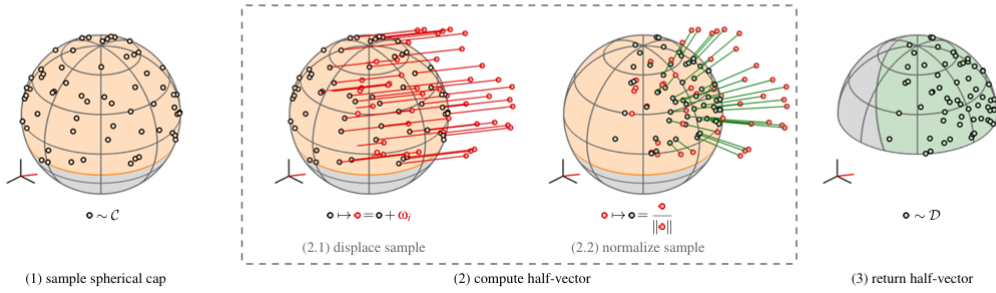


Figure 5: Sampling the visible hemisphere using spherical caps (our method).

图 6: Spherical Caps VNDF 采样步骤

采样步骤如下：

- (1) 采样光线的出射方向，其表现为球的球冠，对应最低高度为 $-z_i$ ，即观察方向的 z 坐标，论文中证明了镜面反射的出射方向为球冠的随机位置。
- (2) 根据出射方向和观察方向计算半程向量，作为 visible normal。

2.4.2 代码

```

1 // Sampling the visible hemisphere as half vectors(our method)
2 vec3 SampleVndf_Hemisphere(vec2 u, vec3 wi)
3 {
4     //sample a spherical cap in  $(-wi.z, 1]$ 
5     float phi= 2.0 f * M_PI * u.x;

```

```

6      // in hlsl: fma(a, b, c) = a * b + c
7      float z= fma((1.0f - u.y), (1.0f + wi.z), -wi.z);
8      float sinTheta = sqrt(clamp(1.0f - z * z, 0.0f, 1.0f));
9      float x = sinTheta * cos(phi);
10
11     float y = sinTheta * sin(phi);
12     vec3 c = vec3(x,y,z);
13     // compute halfway direction;
14     vec3 h = c + wi;
15     // return without normalization(as this is done later)
16     return h;
17 }

```

注：此处代码忽略了将观察方向从半椭球体上映射到半球上以及将法线映射回半椭球上的流程，实际使用时需要。

2.5 Bounded VNDF

2.5.1 采样步骤

Bounded VNDF 在 Spherical Caps VNDF 的基础上计算了更准确的出射方向的下界。

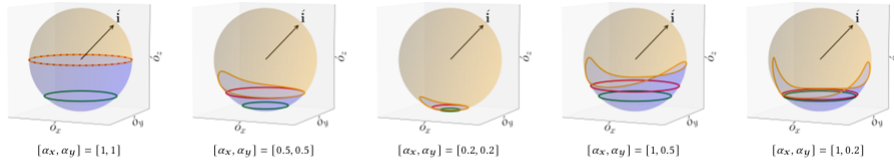


Figure 3: Previous lower bound (green) and our lower bound (red) for the spherical cap in VNDF sampling. The orange and blue regions correspond to reflection vectors in the upper and lower hemispheres, respectively. Since our method bounds the orange region more tightly than the previous method, it reduces the number of reflection vectors occluded by the surface.

图 7: Bounded VNDF 采样步骤

(1) 反射向量的下界

令反射向量为 $\mathbf{o} = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$ ，空间变换后的反射向量为 $\mathbf{o}' = (o'_x, o'_y, o'_z)$ ；观察方向为 $\mathbf{i} = (i_x, i_y, i_z)$ ，空间变换后的观察方向为 $\mathbf{i}' = (i'_x, i'_y, i'_z)$ 。

可以推出 o'_z 的在 ϕ 处的下界（论证在论文中的补充材料中）：

$$\begin{aligned}\min_{\theta \in [0, \pi/2]} o'_z(\theta, \phi) &= o'_z(\pi/2, \phi) \\ &= \left(\frac{(i_x + \cos\phi)^2 + (i_y + \sin\phi)^2 + i_z^2}{\frac{(i_x + \cos\phi)^2}{\alpha_x^2} + \frac{(i_y + \sin\phi)^2}{\alpha_y^2} + i_z^2} - 1 \right) i'_z\end{aligned}$$

其中 $i'_z > 0$ 。进一步有：

$$\min_{\phi} o'_z(\pi/2, \phi) = \min_{\phi} \frac{(i_x + \cos\phi)^2 + (i_y + \sin\phi)^2 + i_z^2}{\frac{(i_x + \cos\phi)^2}{\alpha_x^2} + \frac{(i_y + \sin\phi)^2}{\alpha_y^2} + i_z^2}$$

（2）球冠 Spherical Cap 的下界

（2.1）各项同性，即 $\alpha = \alpha_x = \alpha_y$ ：

$$\min_{\phi} o'_z\left(\frac{\pi}{2}, \phi\right) = \begin{cases} \max_{\phi} r(\phi), \alpha < 1 \\ \min_{\phi} r(\phi), \alpha > 1 \\ \mathbb{R}, \alpha = 1 \end{cases}$$

其中 $r(\phi) = (i_x + \cos\phi)^2 + (i_y + \sin\phi)^2$ ，对应的极值为 $(1 \pm \sqrt{i_x^2 + i_y^2})^2$ 。
则最小值公式转化为：

$$\min_{\phi} o'_z\left(\frac{\pi}{2}, \phi\right) = \begin{cases} \{\phi | r(\phi) = s^2\}, \alpha \neq 1 \\ \mathbb{R}, \alpha = 1 \end{cases}$$

其中 $s = 1 + \text{sgn}(1 - \alpha)\sqrt{i_x^2 + i_y^2}$ 。代入最初的公式可以得到：

$$\min_{\phi} o'_z\left(\frac{\pi}{2}, \phi\right) = -ki'_z, \text{ where } k = \frac{(1 - \alpha^2)s^2}{s^2 + \alpha^2 i_z^2}$$

当 $i'_z > 0$ ，在 $(-ki'_z, 1]$ 上采样 o'_z ；当 $i'_z \leq 0$ ，采取和 Spherical Caps 相同的方法，在 $(-i'_z, 1]$ 上采样 o'_z 。

（2.2）各项异性，即 $\alpha_x \neq \alpha_y$

令 $\alpha = \min(\alpha_x, \alpha_y, 1)$ ，代入上式进行计算。

2.5.2 Bounded VNDF 的 PDF 计算

VNDF 的 PDF 计算公式如下:

$$p(\mathbf{m}) = \frac{2D(\mathbf{m})\max(\mathbf{i} \cdot \mathbf{m}, 0)}{i_z + \sqrt{\alpha_x^2 i_x^2 + \alpha_y^2 i_y^2 + i_z^2}},$$

对于 $i_z \leq 0$, Bounded VNDF 采取和 VNDF 相同的 PDF 计算方法;
对于 $i_z > 0$, 将 spherical cap 从 $(-i'_z, 1]$ 映射到 $(-ki'_z, 1]$, 采用如下公式:

$$p_{\text{bounded}}(\mathbf{m}) = \frac{2D(\mathbf{m})\max(\mathbf{i} \cdot \mathbf{m}, 0)}{ki_z + \sqrt{\alpha_x^2 i_x^2 + \alpha_y^2 i_y^2 + i_z^2}} \chi^+(ki'_z + o'_z),$$

除此之外, 上式还需要乘以半程向量和反射向量之间变换对应的 Jacobian 项 $\|d\mathbf{m}/do\| = 1/(4|\mathbf{i} \cdot \mathbf{m}|)$ 。

2.5.3 代码

```
1 float3 SampleGGXReflection(float3 i, float2 alpha, float2 rand) {
2     float3 i_std = normalize(float3(i.xy * alpha, i.z));
3     // Sample a spherical cap
4     float phi = 2.0f * M_PI * rand.x;
5     float a = saturate(min(alpha.x, alpha.y));
6     float s = 1.0f + length(float2(i.x, i.y)); // Omit sgn for a<=1
7     float a2 = a * a; float s2 = s * s;
8     float k = (1.0f - a2) * s2 / (s2 + a2 * i.z * i.z);
9     float b = i.z > 0 ? k * i_std.z : i_std.z;
10    float z = mad(1.0f - rand.y, 1.0f + b, -b);
11    float sinTheta = sqrt(saturate(1.0f - z * z));
12    float3 o_std = {sinTheta * cos(phi), sinTheta * sin(phi), z};
13    // Compute the microfacet normal m
14    float3 m_std = i_std + o_std;
15    float3 m = normalize(float3(m_std.xy * alpha, m_std.z));
16    // Return the reflection vector o
17    return 2.0f * dot(i, m) * m - i;
18 }
```

```

1 float GGXReflectionPDF(float3 i, float3 o, float2 alpha) {
2     float3 m = normalize(i + o);
3     float ndf = D(m, alpha);
4     float2 ai = alpha * i.xy;
5     float len2 = dot(ai, ai);
6     float t = sqrt(len2 + i.z * i.z);
7     if (i.z >= 0.0f) {
8         float a = saturate(min(alpha.x, alpha.y)); // Eq. 6
9         float s = 1.0f + length(float2(i.x, i.y)); // Omit sgn for a<=1
10        float a2 = a * a; float s2 = s * s;
11        float k = (1.0f - a2) * s2 / (s2 + a2 * i.z * i.z); // Eq. 5
12        return ndf / (2.0f * (k * i.z + t)); // Eq. 8 * ||dm/do||
13    }
14    // Numerically stable form of the previous PDF for i.z < 0
15    return ndf * (t - i.z) / (2.0f * len2); // = Eq. 7 * ||dm/do||
16 }

```

2.6 个人应用记录

个人在工作中应用过上述的 VNDF，主要是对于自研渲染引擎的材质模块中的反射部分：

(1) 在此之前，使用的是普通的 GGX NDF，应用过 Ellipsoid VNDF 之后，反射效果有明显提升，尤其是 roughness 较小的时候，减少了噪点。

(2) 后来阅读了 Spherical Caps VNDF 和 Bounded VNDF 的论文后，也在引擎内进行了尝试。但是，相较于 Ellipsoid VNDF，没有明显的效果和性能上的提升。为了引擎效果的稳定性，没有上线该改动。

2.7 参考

Sampling the GGX Distribution of Visible Normals
 Sampling Visible GGX Normals with Spherical Caps
 Bounded VNDF Sampling for Smith-GGX Reflections