Credits 孤心不寒@bilibili
https://space.bilibili.com/623475709

球谐函数颜色和位置的梯度

```
// Use PyTorch rule for clamping: if clamping was applied,
// gradient becomes 0.
glm::vec3 dL_dRGB = dL_dcolor[idx];
dL_dRGB.x *= clamped[3 * idx + 0] ? 0 : 1;
dL_dRGB.y *= clamped[3 * idx + 1] ? 0 : 1;
dL_dRGB.z *= clamped[3 * idx + 2] ? 0 : 1;
```

```
// Target location for this Gaussian to write SH gradients to
glm::vec3* dL_dsh = dL_dshs + idx * max_coeffs;

// No tricks here, just high school-level calculus.
float dRGBdsh0 = SH_C0;
dL_dsh[0] = dRGBdsh0 * dL_dRGB;
if (deg > 0)
{
    float dRGBdsh1 = -SH_C1 * y;
    float dRGBdsh2 = SH_C1 * z;
    float dRGBdsh3 = -SH_C1 * x;
    dL_dsh[1] = dRGBdsh1 * dL_dRGB;
    dL_dsh[2] = dRGBdsh2 * dL_dRGB;
    dL_dsh[3] = dRGBdsh3 * dL_dRGB;

    dRGBdx = -SH_C1 * sh[3];
    dRGBdy = -SH_C1 * sh[1];
    dRGBdz = SH_C1 * sh[2];

    if (deg > 1)
    {
        float xx = x * x, yy = y * y, zz = z * z;
        float xy = x * y, yz = y * z, xz = x * z;

        float dRGBdsh4 = SH_C2[0] * xy;
        float dRGBdsh5 = SH_C2[1] * yz;
        float dRGBdsh6 = SH_C2[2] * (2.f * zz - xx - yy);
        float dRGBdsh7 = SH_C2[3] * xz;
        float dRGBdsh8 = SH_C2[4] * (xx - yy);
        dL_dsh[4] = dRGBdsh4 * dL_dRGB;
        dL_dsh[5] = dRGBdsh5 * dL_dRGB;
        dL_dsh[6] = dRGBdsh6 * dL_dRGB;
        dL_dsh[7] = dRGBdsh7 * dL_dRGB;
        dL_dsh[8] = dRGBdsh8 * dL_dRGB;

        dRGBdx += SH_C2[0] * y * sh[4] + SH_C2[2] * 2.f * -x * sh[6] + SH_C2[3] * z * sh[7] + SH_C2[4] * 2.f * x * sh[8];
        dRGBdy += SH_C2[0] * x * sh[4] + SH_C2[1] * z * sh[5] + SH_C2[2] * 2.f * -y * sh[6] + SH_C2[4] * 2.f * -y * sh[8];
        dRGBdz += SH_C2[1] * y * sh[5] + SH_C2[2] * 2.f * 2.f * z * sh[6] + SH_C2[3] * x * sh[7];
```

```
// Differentiable Point-Based Radiance Fields for
// Efficient View Synthesis" by Zhang et al. (2022)
glm::vec3 pos = means[idx];
glm::vec3 dir = pos - campos;
dir = dir / glm::length(dir);

glm::vec3* sh = ((glm::vec3*)shs) + idx * max_coeffs;
glm::vec3 result = SH_C0 * sh[0];

if (deg > 0)
{
    float x = dir.x;
    float y = dir.y;
    float z = dir.z;
    result = result - SH_C1 * y * sh[1] + SH_C1 * z * sh[2] - SH_C1 * x * sh[3];

    if (deg > 1)
    {
        float xx = x * x, yy = y * y, zz = z * z;
        float xy = x * y, yz = y * z, xz = x * z;
        result = result +
            SH_C2[0] * xy * sh[4] +
            SH_C2[1] * yz * sh[5] +
            SH_C2[2] * (2.0f * zz - xx - yy) * sh[6] +
            SH_C2[3] * xz * sh[7] +
            SH_C2[4] * (xx - yy) * sh[8];

        if (deg > 2)
        {
            result = result +
                SH_C3[0] * y * (3.0f * xx - yy) * sh[9] +
                SH_C3[1] * xy * z * sh[10] +
                SH_C3[2] * y * (4.0f * zz - xx - yy) * sh[11] +
                SH_C3[3] * z * (2.0f * zz - 3.0f * xx - 3.0f * yy) * sh[12] +
                SH_C3[4] * x * (4.0f * zz - xx - yy) * sh[13] +
                SH_C3[5] * z * (xx - yy) * sh[14] +
                SH_C3[6] * x * (xx - 3.0f * yy) * sh[15];
        }
    }
}
result += 0.5f;
```

用RGB来替换$result$

对于$\deg = 0$:

$$RGB = C_0 \cdot sh_0$$
$$\frac{\partial RGB}{\partial sh_0} = C_0$$
$$\frac{\partial L}{\partial sh_0} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_0} = C_0 \cdot \frac{\partial L}{\partial RGB}$$

对于$\deg > 0$:

对于球谐函数系数的梯度:

$$RGB = RGB - C_1 \cdot y \cdot sh1 + C_1 \cdot z \cdot sh2 - C_1 \cdot x \cdot sh3$$
$$\frac{\partial L}{\partial sh_1} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_1} = -C_1 \cdot y \cdot \frac{\partial L}{\partial RGB}$$
$$\frac{\partial L}{\partial sh_2} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_2} = C_1 \cdot z \cdot \frac{\partial L}{\partial RGB}$$
$$\frac{\partial L}{\partial sh_3} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_3} = -C_1 \cdot x \cdot \frac{\partial L}{\partial RGB}$$

对于通道（方向）的梯度:

$$\frac{\partial RGB}{\partial x} = -C_1 \cdot sh_3$$
$$\frac{\partial RGB}{\partial y} = C_1 \cdot sh_1$$
$$\frac{\partial RGB}{\partial z} = -C_1 \cdot sh_2$$

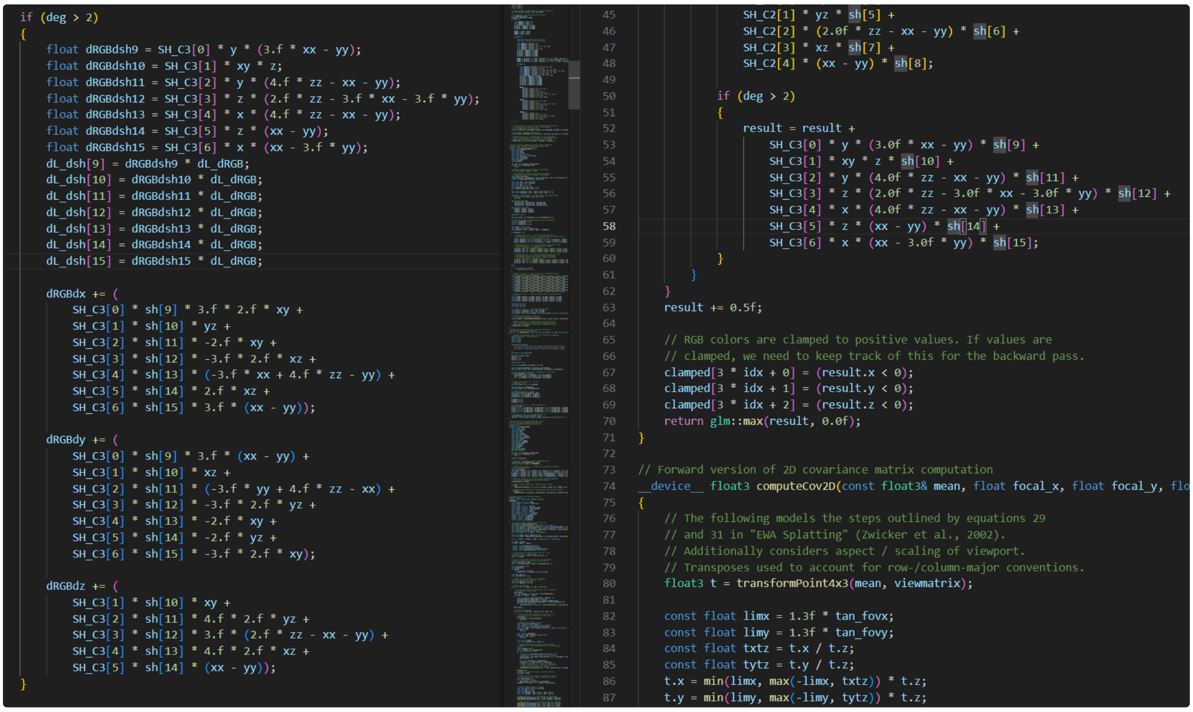
对于$deg > 1$:

对于球谐函数系数的梯度:

$$
\begin{gathered}
RGB = RGB + C_2[0] \cdot xy \cdot sh4 + C_2[1] \cdot yz \cdot sh5 + C_2[2] \cdot (2.0 \cdot zz - xx - yy) \cdot sh6 + \\
C_2[3] \cdot xz \cdot sh7 + C_2[4] \cdot (xx - yy) \cdot sh8 \\
\frac{\partial L}{\partial sh_4} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_4} = C_2[0] \cdot xy \cdot \frac{\partial L}{\partial RGB} \\
\frac{\partial L}{\partial sh_5} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_5} = C_2[1] \cdot yz \cdot \frac{\partial L}{\partial RGB} \\
\frac{\partial L}{\partial sh_6} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_6} = C_2[2] \cdot (2z^2 - x^2 - y^2) \cdot \frac{\partial L}{\partial RGB} \\
\frac{\partial L}{\partial sh_7} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_7} = C_2[3] \cdot xz \cdot \frac{\partial L}{\partial RGB} \\
\frac{\partial L}{\partial sh_8} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_8} = C_2[4] \cdot (x^2 - y^2) \cdot \frac{\partial L}{\partial RGB}
\end{gathered}
$$

对于通道(方向)的梯度:

$$
\begin{gathered}
\frac{\partial RGB}{\partial x} = C_2[0] \cdot y \cdot sh_4 + C_2[1] \cdot -2x \cdot sh_6 + C_2[3] \cdot z \cdot sh_7 + C_2[4] \cdot 2x \cdot sh_8 \\
\frac{\partial RGB}{\partial y} = C_2[0] \cdot x \cdot sh_4 + C_2[1] \cdot z \cdot sh_5 + C_2[2] \cdot -2y \cdot sh_6 + C_2[4] \cdot -2y \cdot sh_8 \\
\frac{\partial RGB}{\partial z} = C_2[1] \cdot y \cdot sh_5 + C_2[2] \cdot 4z \cdot sh_6 + C_2[3] \cdot x \cdot sh_7
\end{gathered}
$$



```
if (deg > 2)
{
    float dRGBdsh9 = SH_C3[0] * y * (3.f * xx - yy);
    float dRGBdsh10 = SH_C3[1] * xy * z;
    float dRGBdsh11 = SH_C3[2] * y * (4.f * zz - xx - yy);
    float dRGBdsh12 = SH_C3[3] * z * (2.f * zz - 3.f * xx - 3.f * yy);
    float dRGBdsh13 = SH_C3[4] * x * (4.f * zz - xx - yy);
    float dRGBdsh14 = SH_C3[5] * z * (xx - yy);
    float dRGBdsh15 = SH_C3[6] * x * (xx - 3.f * yy);
    dL_dsh[9] = dRGBdsh9 * dL_dRGB;
    dL_dsh[10] = dRGBdsh10 * dL_dRGB;
    dL_dsh[11] = dRGBdsh11 * dL_dRGB;
    dL_dsh[12] = dRGBdsh12 * dL_dRGB;
    dL_dsh[13] = dRGBdsh13 * dL_dRGB;
    dL_dsh[14] = dRGBdsh14 * dL_dRGB;
    dL_dsh[15] = dRGBdsh15 * dL_dRGB;

    dRGBdx += (
        SH_C3[0] * sh[9] * 3.f * 2.f * xy +
        SH_C3[1] * sh[10] * yz +
        SH_C3[2] * sh[11] * -2.f * xy +
        SH_C3[3] * sh[12] * -3.f * 2.f * xz +
        SH_C3[4] * sh[13] * (-3.f * xx + 4.f * zz - yy) +
        SH_C3[5] * sh[14] * 2.f * xz +
        SH_C3[6] * sh[15] * 3.f * (xx - yy));

    dRGBdy += (
        SH_C3[0] * sh[9] * 3.f * (xx - yy) +
        SH_C3[1] * sh[10] * xz +
        SH_C3[2] * sh[11] * (-3.f * yy + 4.f * zz - xx) +
        SH_C3[3] * sh[12] * -3.f * 2.f * yz +
        SH_C3[4] * sh[13] * -2.f * xy +
        SH_C3[5] * sh[14] * -2.f * yz +
        SH_C3[6] * sh[15] * -3.f * 2.f * xy);

    dRGBdz += (
        SH_C3[1] * sh[10] * xy +
        SH_C3[2] * sh[11] * 4.f * 2.f * yz +
        SH_C3[3] * sh[12] * 3.f * (2.f * zz - xx - yy) +
        SH_C3[4] * sh[13] * 4.f * 2.f * xz +
        SH_C3[5] * sh[14] * (xx - yy));
}
```

```
                    SH_C2[1] * yz * sh[5] +
                    SH_C2[2] * (2.0f * zz - xx - yy) * sh[6] +
                    SH_C2[3] * xz * sh[7] +
                    SH_C2[4] * (xx - yy) * sh[8];

                if (deg > 2)
                {
                    result = result +
                        SH_C3[0] * y * (3.0f * xx - yy) * sh[9] +
                        SH_C3[1] * xy * z * sh[10] +
                        SH_C3[2] * y * (4.0f * zz - xx - yy) * sh[11] +
                        SH_C3[3] * z * (2.0f * zz - 3.0f * xx - 3.0f * yy) * sh[12] +
                        SH_C3[4] * x * (4.0f * zz - xx - yy) * sh[13] +
                        SH_C3[5] * z * (xx - yy) * sh[14] +
                        SH_C3[6] * x * (xx - 3.0f * yy) * sh[15];
                }
            }
        }
    result += 0.5f;

    // RGB colors are clamped to positive values. If values are
    // clamped, we need to keep track of this for the backward pass.
    clamped[3 * idx + 0] = (result.x < 0);
    clamped[3 * idx + 1] = (result.y < 0);
    clamped[3 * idx + 2] = (result.z < 0);
    return glm::max(result, 0.0f);
}

// Forward version of 2D covariance matrix computation
__device__ float3 computeCov2D(const float3& mean, float focal_x, float focal_y, flo
{
    // The following models the steps outlined by equations 29
    // and 31 in "EWA Splatting" (Zwicker et al., 2002).
    // Additionally considers aspect / scaling of viewport.
    // Transposes used to account for row-/column-major conventions.
    float3 t = transformPoint4x3(mean, viewmatrix);

    const float limx = 1.3f * tan_fovx;
    const float limy = 1.3f * tan_fovy;
    const float txtz = t.x / t.z;
    const float tytz = t.y / t.z;
    t.x = min(limx, max(-limx, txtz)) * t.z;
    t.y = min(limy, max(-limy, tytz)) * t.z;
```

对于$deg > 2$:

对于球谐函数系数的梯度:

$$
RGB = RGB + SH_C3[0] \cdot y \cdot (3.0f \cdot xx - yy) \cdot sh_9 + SH_C3[1] \cdot xy \cdot z \cdot sh_{10} + SH_C3[2] \cdot y \cdot (4.0f \cdot zz - xx - yy) \cdot sh_{11} + \\
SH_C3[3] \cdot z \cdot (2.0f \cdot zz - 3.0f \cdot xx - 3.0f \cdot yy) \cdot sh_{12} + SH_C3[4] \cdot x \cdot (4.0f \cdot zz - xx - yy) \cdot sh_{13} + \\
SH_C3[5] \cdot z \cdot (xx - yy) \cdot sh_{14} + SH_C3[6] \cdot x \cdot (xx - 3.0f \cdot yy) \cdot sh_{15}
$$

$$
\frac{\partial L}{\partial sh_9} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_9} = SH_C3[0] \cdot y \cdot (3.f \cdot xx - yy) \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{10}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{10}} = SH_C3[1] \cdot xy \cdot z \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{11}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{11}} = SH_C3[2] \cdot y \cdot (4.f \cdot zz - xx - yy) \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{12}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{12}} = SH_C3[3] \cdot z \cdot (2.f \cdot zz - 3.f \cdot xx - 3.f \cdot yy) \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{13}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{13}} = SH_C3[4] \cdot x \cdot (4.f \cdot zz - xx - yy) \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{14}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{14}} = SH_C3[5] \cdot z \cdot (xx - yy) \cdot \frac{\partial L}{\partial RGB}
$$

$$
\frac{\partial L}{\partial sh_{15}} = \frac{\partial L}{\partial RGB} \cdot \frac{\partial RGB}{\partial sh_{15}} = SH_C3[6] \cdot x \cdot (xx - 3.f \cdot yy) \cdot \frac{\partial L}{\partial RGB}
$$

对于通道（方向）的梯度:

$$
\frac{\partial RGB}{\partial x} = SH_C3[0] \cdot sh_9 \cdot 3.f \cdot 2.f \cdot xy + SH_C3[1] \cdot sh_{10} \cdot yz + SH_C3[2] \cdot sh_{11} \cdot -2.f \cdot xy + \\
SH_C3[3] \cdot sh_{12} \cdot -3.f \cdot 2.f \cdot xz + SH_C3[4] \cdot sh_{13} \cdot (-3.f \cdot xx + 4.f \cdot zz - yy) + \\
SH_C3[5] \cdot sh_{14} \cdot 2.f \cdot xz + SH_C3[6] \cdot sh_{15} \cdot 3.f \cdot (xx - yy)
$$

$$
\frac{\partial RGB}{\partial y} = SH_C3[0] \cdot sh_9 \cdot 3.f \cdot (xx - yy) + SH_C3[1] \cdot sh_{10} \cdot xz + SH_C3[2] \cdot sh_{11} \cdot (-3.f \cdot yy + 4.f \cdot zz - xx) + \\
SH_C3[3] \cdot sh_{12} \cdot -3.f \cdot 2.f \cdot yz + SH_C3[4] \cdot sh_{13} \cdot -2.f \cdot xy + \\
SH_C3[5] \cdot sh_{14} \cdot -2.f \cdot yz + SH_C3[6] \cdot sh_{15} \cdot -3.f \cdot 2.f \cdot xy
$$

$$
\frac{\partial RGB}{\partial z} = SH_C3[1] \cdot sh_{10} \cdot xy + SH_C3[2] \cdot sh_{11} \cdot 4.f \cdot 2.f \cdot yz + \\
SH_C3[3] \cdot sh_{12} \cdot 3.f \cdot (2.f \cdot zz - xx - yy) + SH_C3[4] \cdot sh_{13} \cdot 4.f \cdot 2.f \cdot xz + \\
SH_C3[5] \cdot sh_{14} \cdot (xx - yy)
$$

二维协方差矩阵三个参数a、b、c的梯度

```
// Gradients of loss w.r.t. entries of 2D covariance matrix,
// given gradients of loss w.r.t. conic matrix (inverse covariance matrix).
// e.g., dL / da = dL / d_conic_a * d_conic_a / d_a
dL_da = denom2inv * (-c * c * dL_dconic.x + 2 * b * c * dL_dconic.y + (denom - a * c) * dL_dconic.z);
dL_dc = denom2inv * (-a * a * dL_dconic.z + 2 * a * b * dL_dconic.y + (denom - a * c) * dL_dconic.x);
dL_db = denom2inv * 2 * (b * c * dL_dconic.x - (denom + 2 * b * b) * dL_dconic.y + a * b * dL_dconic.z);
```

二维协方差矩阵的三个参数 a、b和 c 对应的是以下矩阵形式:

$$
\mathbf{C} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}
$$

损失函数 L对应的是通过逆变换（即矩阵的逆）求出的一个二次曲线矩阵（conic matrix）。假设我们有以下损失函数:

$$
L = f(\mathbf{C}^{-1})
$$

我们要计算损失函数L 对协方差矩阵 $\mathbf{C}$中参数的a、b和 c 梯度。
首先，我们需要知道$\mathbf{C}$的逆矩阵:

$$
\mathbf{C}^{-1} = \frac{1}{\det(\mathbf{C})} \begin{bmatrix} c & -b \\ -b & a \end{bmatrix}
$$

其中$\det(\mathbf{C}) = ac - b^2$是矩阵$\mathbf{C}$ 的行列式。
假设我们有以下损失函数:

$$
L = f(\mathbf{C}^{-1}) = f\left(\frac{1}{\det(\mathbf{C})} \begin{bmatrix} c & -b \\ -b & a \end{bmatrix}\right)
$$

记$\mathbf{C}^{-1}=\begin{bmatrix} C_{11} & C_{12} \\ C_{12} & C_{22} \end{bmatrix}$
损失函数L 对$\mathbf{C}$ 中元素a、b和 c 的梯度分别为:

$$\frac{\partial L}{\partial a}=\frac{\partial L}{\partial C_{11}}\cdot\frac{\partial C_{11}}{\partial a}+\frac{\partial L}{\partial C_{12}}\cdot\frac{\partial C_{12}}{\partial a}+\frac{\partial L}{\partial C_{22}}\cdot\frac{\partial C_{22}}{\partial a}$$

类似地，可以写出 $\frac{\partial L}{\partial b}$ 和$\frac{\partial L}{\partial c}$。
我们分别计算各项的偏导数,同时还需要考虑 $\det(\mathbf{C})$ 对各项的影响，因为$\det(\mathbf{C})=ac-b^2$:

$$\frac{\partial \det(\mathbf{C})}{\partial a}=c,\quad \frac{\partial \det(\mathbf{C})}{\partial b}=-2b,\quad \frac{\partial \det(\mathbf{C})}{\partial c}=a$$

因此我们需要使用链式法则:

$$\frac{\partial C_{11}}{\partial a}=\frac{\partial}{\partial a}\left(\frac{c}{\det(\mathbf{C})}\right)=\frac{1}{\det(\mathbf{C})}\cdot\frac{\partial c}{\partial a}-\frac{c}{\det(\mathbf{C})^2}\cdot\frac{\partial \det(\mathbf{C})}{\partial a}=-\frac{c^2}{\det(\mathbf{C})^2}$$

$$\frac{\partial C_{12}}{\partial a}=\frac{\partial}{\partial a}\left(-\frac{b}{\det(\mathbf{C})}\right)=\frac{-b\cdot(-c)}{\det(\mathbf{C})^2}=\frac{bc}{\det(\mathbf{C})^2}$$

$$\frac{\partial C_{22}}{\partial a}=\frac{\partial}{\partial a}\left(\frac{a}{\det(\mathbf{C})}\right)=\frac{a^2-ac}{\det(\mathbf{C})^2}$$

所以,

$$\frac{\partial L}{\partial a}=\frac{1}{\det(\mathbf{C})^2}\left(-c^2\frac{\partial L}{\partial C_{11}}+2bc\frac{\partial L}{\partial C_{12}}+(ac-a^2)\frac{\partial L}{\partial C_{22}}\right)$$

同理可以得到$\frac{\partial L}{\partial b}$ 和 $\frac{\partial L}{\partial c}$。
这样，我们就得到了以下公式:

$$\frac{\partial L}{\partial c}=\frac{1}{\mathrm{denom}^2}\left(-a^2\cdot\frac{\partial L}{\partial \mathrm{conic}_c}+2ab\cdot\frac{\partial L}{\partial \mathrm{conic}_b}+(\mathrm{denom}-ac)\cdot\frac{\partial L}{\partial \mathrm{conic}_a}\right)$$

$$\frac{\partial L}{\partial b}=\frac{2}{\mathrm{denom}^2}\left(bc\cdot\frac{\partial L}{\partial \mathrm{conic}_a}-(\mathrm{denom}+2b^2)\cdot\frac{\partial L}{\partial \mathrm{conic}_b}+ab\cdot\frac{\partial L}{\partial \mathrm{conic}_c}\right)$$

三维协方差矩阵六个参数cov3D[0,1,2,3,4,5]的梯度

```
// Gradients of loss L w.r.t. each 3D covariance matrix (Vrk) entry,
// given gradients w.r.t. 2D covariance matrix (diagonal).
// cov2D = transpose(T) * transpose(Vrk) * T;
dL_dcov[6 * idx + 0] = (T[0][0] * T[0][0] * dL_da + T[0][0] * T[1][0] * dL_db + T[1][0] * T[1][0] * dL_dc);
dL_dcov[6 * idx + 3] = (T[0][1] * T[0][1] * dL_da + T[0][1] * T[1][1] * dL_db + T[1][1] * T[1][1] * dL_dc);
dL_dcov[6 * idx + 5] = (T[0][2] * T[0][2] * dL_da + T[0][2] * T[1][2] * dL_db + T[1][2] * T[1][2] * dL_dc);

// Gradients of loss L w.r.t. each 3D covariance matrix (Vrk) entry,
// given gradients w.r.t. 2D covariance matrix (off-diagonal).
// Off-diagonal elements appear twice --> double the gradient.
// cov2D = transpose(T) * transpose(Vrk) * T;
dL_dcov[6 * idx + 1] = 2 * T[0][0] * T[0][1] * dL_da + (T[0][0] * T[1][1] + T[0][1] * T[1][0]) * dL_db + 2 * T[1][0] * T[1][1] * dL_dc;
dL_dcov[6 * idx + 2] = 2 * T[0][0] * T[0][2] * dL_da + (T[0][0] * T[1][2] + T[0][2] * T[1][0]) * dL_db + 2 * T[1][0] * T[1][2] * dL_dc;
dL_dcov[6 * idx + 4] = 2 * T[0][2] * T[0][1] * dL_da + (T[0][1] * T[1][2] + T[0][2] * T[1][1]) * dL_db + 2 * T[1][1] * T[1][2] * dL_dc;
```

首先，我们定义三维协方差矩阵$\mathbf{V}_{\mathrm{rk}}$为:

$$\mathbf{V}_{\mathrm{rk}}=\begin{bmatrix} \mathrm{cov3D}[0] & \mathrm{cov3D}[1] & \mathrm{cov3D}[2] \\ \mathrm{cov3D}[1] & \mathrm{cov3D}[3] & \mathrm{cov3D}[4] \\ \mathrm{cov3D}[2] & \mathrm{cov3D}[4] & \mathrm{cov3D}[5] \end{bmatrix}=\begin{bmatrix} V_{00} & V_{01} & V_{02} \\ V_{10} & V_{11} & V_{12} \\ V_{20} & V_{21} & V_{22} \end{bmatrix}$$

二维协方差矩阵$\mathbf{C}$ 的计算公式为:

$$\mathbf{C}=\mathbf{T}^T\mathbf{V}_{\mathrm{rk}}\mathbf{T}$$

其中 $\mathbf{T}$ 矩阵是由视图矩阵 $\mathbf{W}$和 Jacobian 矩阵 $\mathbf{J}$ 相乘得到的，即 $\mathbf{T}=\mathbf{WJ}$。
投影矩阵 $\mathbf{T}$是2×3矩阵:

$$\mathbf{T}=\begin{bmatrix} T_{00} & T_{01} & T_{02} \\ T_{10} & T_{11} & T_{12} \end{bmatrix}$$

二维协方差矩阵$\mathbf{C}$的形式为：

$$\mathbf{C} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

我们需要通过 $\frac{\partial L}{\partial a}$、$\frac{\partial L}{\partial b}$ 和 $\frac{\partial L}{\partial c}$ 计算 $\mathbf{V}$矩阵中各个元素的梯度 $\frac{\partial L}{\partial \mathbf{V}_{ij}}$。根据链式法则，我们有：

$$\frac{\partial L}{\partial \mathbf{V}_{ij}} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial \mathbf{V}_{ij}} + \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial \mathbf{V}_{ij}} + \frac{\partial L}{\partial c} \cdot \frac{\partial c}{\partial \mathbf{V}_{ij}}$$

首先，我们需要计算a、b和c 对$\mathbf{V}$矩阵中各个元素的梯度。
其中，二维协方差矩阵的$(0,0)$元素C_{00}是：

$$C_{00} = \mathbf{T}_{00}^2\mathbf{V}_{00} + 2\mathbf{T}_{00}\mathbf{T}_{01}\mathbf{V}_{01} + \mathbf{T}_{01}^2\mathbf{V}_{11} + 2\mathbf{T}_{00}\mathbf{T}_{02}\mathbf{V}_{02} + 2\mathbf{T}_{01}\mathbf{T}_{02}\mathbf{V}_{12} + \mathbf{T}_{02}^2\mathbf{V}_{22}$$

但实际应用中，如果只考虑二维投影，忽略第三维的影响，可以简化计算过程。
那么 a 就是：

$$a = \mathbf{T}_{00}^2\mathbf{V}_{00} + 2\mathbf{T}_{00}\mathbf{T}_{01}\mathbf{V}_{01} + \mathbf{T}_{01}^2\mathbf{V}_{11}$$

因此：

$$\frac{\partial a}{\partial \mathbf{V}_{00}} = \mathbf{T}_{00}^2$$

$$\frac{\partial a}{\partial \mathbf{V}_{01}} = 2\mathbf{T}_{00}\mathbf{T}_{01}$$

$$\frac{\partial a}{\partial \mathbf{V}_{11}} = \mathbf{T}_{01}^2$$

同理，我们可以得到 b和c 的偏导数：

$$b = \mathbf{T}_{00}\mathbf{T}_{10}\mathbf{V}_{00} + (\mathbf{T}_{00}\mathbf{T}_{11} + \mathbf{T}_{01}\mathbf{T}_{10})\mathbf{V}_{01} + \mathbf{T}_{01}\mathbf{T}_{11}\mathbf{V}_{11}$$

$$\frac{\partial b}{\partial \mathbf{V}_{00}} = \mathbf{T}_{00}\mathbf{T}_{10}$$

$$\frac{\partial b}{\partial \mathbf{V}_{01}} = \mathbf{T}_{00}\mathbf{T}_{11} + \mathbf{T}_{01}\mathbf{T}_{10}$$

$$\frac{\partial b}{\partial \mathbf{V}_{11}} = \mathbf{T}_{01}\mathbf{T}_{11}$$

$$c = \mathbf{T}_{10}^2\mathbf{V}_{00} + 2\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{V}_{01} + \mathbf{T}_{11}^2\mathbf{V}_{11}$$

$$\frac{\partial c}{\partial \mathbf{V}_{00}} = \mathbf{T}_{10}^2$$

$$\frac{\partial c}{\partial \mathbf{V}_{01}} = 2\mathbf{T}_{10}\mathbf{T}_{11}$$

$$\frac{\partial c}{\partial \mathbf{V}_{11}} = \mathbf{T}_{11}^2$$

接下来，我们将这些结果代入到计算$\frac{\partial L}{\partial \mathbf{V}_{ij}}$ 的公式中。以$\frac{\partial L}{\partial \mathbf{V}_{00}}$ 为例：

$$\frac{\partial L}{\partial \mathbf{V}_{00}} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial \mathbf{V}_{00}} + \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial \mathbf{V}_{00}} + \frac{\partial L}{\partial c} \cdot \frac{\partial c}{\partial \mathbf{V}_{00}}$$

将其代入代码中得到：

$$\frac{\partial L}{\partial \mathrm{cov}}[6 \cdot \mathrm{idx} + 0] = \mathbf{T}_{00}^2 \cdot \frac{\partial L}{\partial \mathrm{a}} + \mathbf{T}_{00}\mathbf{T}_{10} \cdot \frac{\partial L}{\partial \mathrm{b}} + \mathbf{T}_{10}^2 \cdot \frac{\partial L}{\partial \mathrm{c}}$$

同理，可以计算其他元素的梯度：

$$\frac{\partial L}{\partial \mathrm{cov}}[6\cdot\mathrm{idx}+1]=2\mathbf{T}_{00}\mathbf{T}_{01}\cdot\frac{\partial L}{\partial \mathrm{a}}+(\mathbf{T}_{00}\mathbf{T}_{11}+\mathbf{T}_{01}\mathbf{T}_{10})\cdot\frac{\partial L}{\partial \mathrm{b}}+2\mathbf{T}_{10}\mathbf{T}_{11}\cdot\frac{\partial L}{\partial \mathrm{c}}$$

$$\frac{\partial L}{\partial \mathrm{cov}}[6\cdot\mathrm{idx}+2]=2\mathbf{T}_{00}\mathbf{T}_{02}\cdot\frac{\partial L}{\partial \mathrm{a}}+(\mathbf{T}_{00}\mathbf{T}_{12}+\mathbf{T}_{02}\mathbf{T}_{10})\cdot\frac{\partial L}{\partial \mathrm{b}}+2\mathbf{T}_{10}\mathbf{T}_{12}\cdot\frac{\partial L}{\partial \mathrm{c}}$$

$$\frac{\partial L}{\partial \mathrm{cov}}[6\cdot\mathrm{idx}+3]=\mathbf{T}_{01}^2\cdot\frac{\partial L}{\partial \mathrm{a}}+\mathbf{T}_{01}\mathbf{T}_{11}\cdot\frac{\partial L}{\partial \mathrm{b}}+\mathbf{T}_{11}^2\cdot\frac{\partial L}{\partial \mathrm{c}}$$

$$\frac{\partial L}{\partial \mathrm{cov}}[6\cdot\mathrm{idx}+4]=2\mathbf{T}_{01}\mathbf{T}_{02}\cdot\frac{\partial L}{\partial \mathrm{a}}+(\mathbf{T}_{01}\mathbf{T}_{12}+\mathbf{T}_{02}\mathbf{T}_{11})\cdot\frac{\partial L}{\partial \mathrm{b}}+2\mathbf{T}_{11}\mathbf{T}_{12}\cdot\frac{\partial L}{\partial \mathrm{c}}$$

$$\frac{\partial L}{\partial \mathrm{cov}}[6\cdot\mathrm{idx}+5]=\mathbf{T}_{02}^2\cdot\frac{\partial L}{\partial \mathrm{a}}+\mathbf{T}_{02}\mathbf{T}_{12}\cdot\frac{\partial L}{\partial \mathrm{b}}+\mathbf{T}_{12}^2\cdot\frac{\partial L}{\partial \mathrm{c}}$$

二维投影协方差矩阵和损失函数对中间矩阵T的梯度的关系

```
// Gradients of loss w.r.t. upper 2x3 portion of intermediate matrix T
// cov2D = transpose(T) * transpose(Vrk) * T;
float dL_dT00 = 2 * (T[0][0] * Vrk[0][0] + T[0][1] * Vrk[0][1] + T[0][2] * Vrk[0][2]) * dL_da +
    (T[1][0] * Vrk[0][0] + T[1][1] * Vrk[0][1] + T[1][2] * Vrk[0][2]) * dL_db;
float dL_dT01 = 2 * (T[0][0] * Vrk[1][0] + T[0][1] * Vrk[1][1] + T[0][2] * Vrk[1][2]) * dL_da +
    (T[1][0] * Vrk[1][0] + T[1][1] * Vrk[1][1] + T[1][2] * Vrk[1][2]) * dL_db;
float dL_dT02 = 2 * (T[0][0] * Vrk[2][0] + T[0][1] * Vrk[2][1] + T[0][2] * Vrk[2][2]) * dL_da +
    (T[1][0] * Vrk[2][0] + T[1][1] * Vrk[2][1] + T[1][2] * Vrk[2][2]) * dL_db;
float dL_dT10 = 2 * (T[1][0] * Vrk[0][0] + T[1][1] * Vrk[0][1] + T[1][2] * Vrk[0][2]) * dL_dc +
    (T[0][0] * Vrk[0][0] + T[0][1] * Vrk[0][1] + T[0][2] * Vrk[0][2]) * dL_db;
float dL_dT11 = 2 * (T[1][0] * Vrk[1][0] + T[1][1] * Vrk[1][1] + T[1][2] * Vrk[1][2]) * dL_dc +
    (T[0][0] * Vrk[1][0] + T[0][1] * Vrk[1][1] + T[0][2] * Vrk[1][2]) * dL_db;
float dL_dT12 = 2 * (T[1][0] * Vrk[2][0] + T[1][1] * Vrk[2][1] + T[1][2] * Vrk[2][2]) * dL_dc +
    (T[0][0] * Vrk[2][0] + T[0][1] * Vrk[2][1] + T[0][2] * Vrk[2][2]) * dL_db;
```

二维协方差矩阵的表达为：

$$\mathbf{C}=\mathbf{T}\mathbf{V}_{\mathrm{rk}}\mathbf{T}^T$$

其中，$\mathbf{T}$是一个2×3的矩阵，$\mathbf{V}_{\mathbf{rk}}$是一个3×3的协方差矩阵。假设损失函数$\mathbf{L}$是二维协方差矩阵的一些函数，为了求损失函数对$\mathbf{T}$的梯度，可是使用链式法则来计算：

$$\frac{\partial L}{\partial T_{ij}}=\sum_{k,l}\frac{\partial L}{\partial C_{kl}}\frac{C_{kl}}{\partial T_{ij}}$$

需要计算的是$\frac{C_{kl}}{\partial T_{ij}}$,二维协方差矩阵的元素$C_{kl}$是通过$T$和$V_{rk}$计算得到的。

$$\mathbf{C}_{kl}=\sum_{m,n}\mathbf{T}_{mk}\mathbf{Vrk}_{mn}\mathbf{T}_{ln}$$

其中m,n分别从0到2，需要关注的是对$\mathbf{T}_{ij}$的偏导数：

$$\frac{\partial C_{kl}}{\partial T_{ij}}=\frac{\partial}{\partial T_{ij}}(\sum_{m,n}T_{mk}VrkT_{ln})$$

需要考虑T_{ij}出现在求和中的位置，假设$i=0$和$k=l=0$,则：

$$\frac{\partial C_{00}}{\partial T_{00}}=\frac{\partial}{\partial T_{00}}(T_{00}Vrk_{00}T_{00}+T_{00}Vrk_{01}T_{01}+T_{01}Vrk_{10}T_{00}+\\ T_{01}Vrk_{11}T_{01}+T_{02}Vrk_{20}T_{00}+T_{02}Vrk_{21}T_{01}+\dots)$$

由此可以看出，C_{00}的梯度包括了对所有涉及$\mathbf{T}_{00}$的项的导数。
而前一部分，$\frac{\partial L}{\partial C_{kl}}$可以直接带入$a,b,c$来表达，比如：$\frac{\partial L}{\partial C_{00}}=\frac{dL}{da}$,$\frac{\partial L}{\partial C_{01}}=\frac{\partial L}{\partial C_{10}}=\frac{dL}{db}$
,$\frac{\partial L}{\partial C_{11}}=\frac{dL}{dc}$。
计算二维协方差矩阵对$\mathbf{T}$ 的梯度
考虑$\mathbf{T}$ 的各个元素T_{ij} 的梯度：

$$\frac{\partial L}{\partial T_{00}}=2\left(T_{00}Vrk_{00}+T_{01}Vrk_{01}+T_{02}Vrk_{02}\right)\frac{dL}{da}+\left(T_{10}Vrk_{00}+T_{11}Vrk_{01}+T_{12}Vrk_{02}\right)\frac{dL}{db}$$

$$\frac{\partial L}{\partial T_{01}} = 2\left(T_{00}Vrk_{10} + T_{01}Vrk_{11} + T_{02}Vrk_{12}\right)\frac{dL}{da} + \left(T_{10}Vrk_{10} + T_{11}Vrk_{11} + T_{12}Vrk_{12}\right)\frac{dL}{db}$$

$$\frac{\partial L}{\partial T_{02}} = 2\left(T_{00}Vrk_{20} + T_{01}Vrk_{21} + T_{02}Vrk_{22}\right)\frac{dL}{da} + \left(T_{10}Vrk_{20} + T_{11}Vrk_{21} + T_{12}Vrk_{22}\right)\frac{dL}{db}$$

$$\frac{\partial L}{\partial T_{10}} = 2\left(T_{10}Vrk_{00} + T_{11}Vrk_{01} + T_{12}Vrk_{02}\right)\frac{dL}{dc} + \left(T_{00}Vrk_{00} + T_{01}Vrk_{01} + T_{02}Vrk_{02}\right)\frac{dL}{db}$$

$$\frac{\partial L}{\partial T_{11}} = 2\left(T_{10}Vrk_{10} + T_{11}Vrk_{11} + T_{12}Vrk_{12}\right)\frac{dL}{dc} + \left(T_{00}Vrk_{10} + T_{01}Vrk_{11} + T_{02}Vrk_{12}\right)\frac{dL}{db}$$

$$\frac{\partial L}{\partial T_{12}} = 2\left(T_{10}Vrk_{20} + T_{11}Vrk_{21} + T_{12}Vrk_{22}\right)\frac{dL}{dc} + \left(T_{00}Vrk_{20} + T_{01}Vrk_{21} + T_{02}Vrk_{22}\right)\frac{dL}{db}$$

计算二维协方差矩阵对J的梯度：

```
// Gradients of loss w.r.t. upper 3x2 non-zero entries of Jacobian matrix
// T = W * J
float dL_dJ00 = W[0][0] * dL_dT00 + W[0][1] * dL_dT01 + W[0][2] * dL_dT02;
float dL_dJ02 = W[2][0] * dL_dT00 + W[2][1] * dL_dT01 + W[2][2] * dL_dT02;
float dL_dJ11 = W[1][0] * dL_dT10 + W[1][1] * dL_dT11 + W[1][2] * dL_dT12;
float dL_dJ12 = W[2][0] * dL_dT10 + W[2][1] * dL_dT11 + W[2][2] * dL_dT12;
```

考虑$\mathbf{T}$ 的各个元素 T_{ij}的梯度,如是一个section所示。
这些公式考虑了每个 T_{ij} 对损失函数 L 的影响，并使用了链式法则将对二维协方差矩阵元素的梯度传递回 $\mathbf{T}$矩阵的梯度。
接下来求上三行两列非零部分的雅可比矩阵
给定$\mathbf{T} = \mathbf{W}\mathbf{J}$，我们可以通过链式法则计算损失函数 L 对雅可比矩阵 $\mathbf{J}$ 的梯度。

$$\frac{\partial T_{ij}}{\partial J_{kj}} = W_{ik}$$

因此：

$$\frac{\partial L}{\partial J_{kj}} = \sum_i \frac{\partial L}{\partial T_{ij}} W_{ik}$$

具体计算如下：

$$\begin{aligned}
\frac{\partial L}{\partial J_{00}} &= W_{00}\frac{\partial L}{\partial T_{00}} + W_{01}\frac{\partial L}{\partial T_{01}} + W_{02}\frac{\partial L}{\partial T_{02}} \\
\frac{\partial L}{\partial J_{02}} &= W_{20}\frac{\partial L}{\partial T_{00}} + W_{21}\frac{\partial L}{\partial T_{01}} + W_{22}\frac{\partial L}{\partial T_{02}} \\
\frac{\partial L}{\partial J_{11}} &= W_{10}\frac{\partial L}{\partial T_{10}} + W_{11}\frac{\partial L}{\partial T_{11}} + W_{12}\frac{\partial L}{\partial T_{12}} \\
\frac{\partial L}{\partial J_{12}} &= W_{20}\frac{\partial L}{\partial T_{10}} + W_{21}\frac{\partial L}{\partial T_{11}} + W_{22}\frac{\partial L}{\partial T_{12}}
\end{aligned}$$

高斯均值 t 的梯度

```
// Gradients of loss w.r.t. transformed Gaussian mean t
float dL_dtx = x_grad_mul * -h_x * tz2 * dL_dJ02;
float dL_dty = y_grad_mul * -h_y * tz2 * dL_dJ12;
float dL_dtz = -h_x * tz2 * dL_dJ00 - h_y * tz2 * dL_dJ11 + (2 * h_x * t.x) * tz3 * dL_dJ02 + (2 * h_y * t.y) * tz3 * dL_dJ12;

// Account for transformation of mean to t
// t = transformPoint4x3(mean, view_matrix);
float3 dL_dmean = transformVec4x3Transpose({ dL_dtx, dL_dty, dL_dtz }, view_matrix);

// Gradients of loss w.r.t. Gaussian means, but only the portion
// that is caused because the mean affects the covariance matrix.
// Additional mean gradient is accumulated in BACKWARD::preprocess.
dL_dmeans[idx] = dL_dmean;
```

首先，，定义相关变量：$tz = \frac{1}{t_z}$ ，这里只涉及到代码，以下推导内容不采用这种表达。

$$tz2 = tz^2$$
$$tz3 = tz^3$$

首先，根据链式法则：$\frac{\partial L}{\partial t_x} = \frac{\partial L}{\partial J_{02}} \cdot \frac{\partial J_{02}}{\partial t_z}$。其中$\frac{\partial J_{02}}{\partial t_z}$是一个由几何变换决定的项，因为试图变换后，$t_x$会受到$t_z$的影响。则有：$\frac{\partial J_{02}}{\partial t_x} = -h_x \cdot \frac{1}{t_z^2}$。再乘以一个标量系数就是该项的梯度。同理，$\frac{\partial L}{\partial t_y} = \frac{\partial L}{\partial J_{12}} \cdot \frac{\partial J_{12}}{\partial t_z}$，根据几何变换的影响，可以得出$\frac{\partial J_{12}}{\partial t_y} = -h_y \cdot \frac{1}{t_z^2}$，再乘以标量系数可以得到对应结果。但是对于$\frac{\partial L}{\partial t_z}$ ，因为t_z会收到t_x, t_y, t_z的影响，所以 $\frac{\partial L}{\partial t_z} = \frac{\partial L}{\partial J_{00}} \cdot \frac{\partial J_{00}}{\partial t_z} + \frac{\partial L}{\partial J_{11}} \cdot \frac{\partial J_{11}}{\partial t_z} + \frac{\partial L}{\partial J_{12}} \cdot \frac{\partial J_{12}}{\partial t_z}$。根据几何变换的影响，有 $\frac{\partial J_{00}}{\partial t_z} = -h_x \cdot \frac{1}{t_z^2}, \frac{\partial J_{11}}{\partial t_z} = -h_y \cdot \frac{1}{t_z^2}, \frac{\partial J_{02}}{\partial t_z} = 2h_x t_x \cdot \frac{1}{t_z^3}, \frac{\partial J_{12}}{\partial t_z} = 2h_y t_y \cdot \frac{1}{t_z^3}$

接下来，计算损失函数 L 相对于变换后高斯均值 $\mathbf{t}$ 的梯度：

$$\frac{\partial L}{\partial t_x} = x\_grad\_mul \cdot \left(-h_x \cdot \frac{1}{t_z^2} \cdot \frac{\partial L}{\partial J_{02}}\right)$$
$$\frac{\partial L}{\partial t_y} = y\_grad\_mul \cdot \left(-h_y \cdot \frac{1}{t_z^2} \cdot \frac{\partial L}{\partial J_{12}}\right)$$
$$\frac{\partial L}{\partial t_z} = -h_x \cdot \frac{1}{t_z^2} \cdot \frac{\partial L}{\partial J_{00}} - h_y \cdot \frac{1}{t_z^2} \cdot \frac{\partial L}{\partial J_{11}} + 2h_x t_x \cdot tz^3 \cdot \frac{\partial L}{\partial J_{02}} + 2h_y t_y \cdot \frac{1}{t_z^3} \cdot \frac{\partial L}{\partial J_{12}}$$

然后，考虑均值 $\mathbf{t}$ 到 $\mathbf{mean}$ 的变换：

$$\mathbf{dL\_dmean} = \text{transformVec4x3Transpose}\left(\left(\begin{pmatrix} \frac{\partial L}{\partial t_x} \\ \frac{\partial L}{\partial t_y} \\ \frac{\partial L}{\partial t_z} \end{pmatrix}, \mathbf{view\_matrix}\right)\right)$$

最终，损失函数 L 相对于高斯均值 $\mathbf{means}$ 的梯度，仅包含因为均值影响协方差矩阵的部分：

$$dL\_dmeans[idx] = dL\_dmean$$

三维协方差矩阵的梯度

```
// Convert per-element covariance loss gradients to matrix form
glm::mat3 dL_dSigma = glm::mat3(
    dL_dcov3D[0], 0.5f * dL_dcov3D[1], 0.5f * dL_dcov3D[2],
    0.5f * dL_dcov3D[1], dL_dcov3D[3], 0.5f * dL_dcov3D[4],
    0.5f * dL_dcov3D[2], 0.5f * dL_dcov3D[4], dL_dcov3D[5]
);

// Compute loss gradient w.r.t. matrix M
// dSigma_dM = 2 * M
glm::mat3 dL_dM = 2.0f * M * dL_dSigma;
```

定义四元数$q = (r, x, y, z)$，可以得到旋转矩阵R和缩放矩阵S：

$$R = \begin{pmatrix} 1 - 2(y^2 + z^2) & 2(xy - rz) & 2(xz + ry) \\ 2(xy + rz) & 1 - 2(x^2 + z^2) & 2(yz - rx) \\ 2(xz - ry) & 2(yz + rx) & 1 - 2(x^2 + y^2) \end{pmatrix}$$

$$S = \begin{pmatrix} s.x & 0 & 0 \\ 0 & s.y & 0 \\ 0 & 0 & s.z \end{pmatrix}$$

矩阵M是旋转矩阵和缩放矩阵的乘积：

$$M = S \cdot R$$

损失函数L相对于协方差矩阵的梯度为$\frac{\partial L}{\partial \Sigma}$，在代码里写作$\frac{\partial L}{\partial Sigma}$。我们需要先计算$M$对$\Sigma$的梯度，即$\frac{\partial \Sigma}{\partial M} = 2M$这个是怎么来的，啰嗦两句：因为$\Sigma$是$M$去掉了对称部分，或者说，$\Sigma$是$M$的对称部分：$\Sigma = M \cdot M^T$。那么：

$$
\begin{aligned}
\frac{\partial \Sigma}{\partial M} = \frac{\partial (M \cdot M^T)}{\partial M} = \frac{\partial M}{\partial M} \cdot M^T + M \cdot \frac{\partial M^T}{\partial M} & (\text{这里用到一个叫做}Kronecker\text{乘积来化简}) \\
= 2M &
\end{aligned}
$$

损失函数相对于矩阵M的梯度为：

$$
\frac{\partial L}{\partial M} = \frac{\partial L}{\partial \Sigma} \cdot \frac{\partial \Sigma}{\partial M} = \frac{\partial L}{\partial \Sigma} \cdot 2M
$$

缩放参数的梯度

```
// Gradients of loss w.r.t. scale
glm::vec3* dL_dscale = dL_dscales + idx;
dL_dscale->x = glm::dot(Rt[0], dL_dMt[0]);
dL_dscale->y = glm::dot(Rt[1], dL_dMt[1]);
dL_dscale->z = glm::dot(Rt[2], dL_dMt[2]);

dL_dMt[0] *= s.x;
dL_dMt[1] *= s.y;
dL_dMt[2] *= s.z;
```

代码中有些参数进行了进一步的定义：比如Rt定义为R矩阵的转置，dl_dMt定义为$\frac{\partial L}{\partial M}$的转置。
由于：

$$
\begin{aligned}
\frac{\partial M}{\partial s_x} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot R \\
\frac{\partial M}{\partial s_y} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot R \\
\frac{\partial M}{\partial s_z} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R
\end{aligned}
$$

对损失函数 (L) 关于缩放因子的梯度可以写作：

$$
\begin{aligned}
\frac{\partial L}{\partial s_x} &= \text{trace}\left(\frac{\partial L}{\partial M} \cdot \frac{\partial M}{\partial s_x}\right) = dot(R^T[0], \frac{\partial M}{\partial s_x}) \\
\frac{\partial L}{\partial s_y} &= \text{trace}\left(\frac{\partial L}{\partial M} \cdot \frac{\partial M}{\partial s_y}\right) = dot(R^T[1], \frac{\partial M}{\partial s_y}) \\
\frac{\partial L}{\partial s_z} &= \text{trace}\left(\frac{\partial L}{\partial M} \cdot \frac{\partial M}{\partial s_z}\right) = dot(R^T[2], \frac{\partial M}{\partial s_z})
\end{aligned}
$$

旋转四元数的梯度

```
// Gradients of loss w.r.t. normalized quaternion
glm::vec4 dL_dq;
dL_dq.x = 2 * z * (dL_dMt[0][1] - dL_dMt[1][0]) + 2 * y * (dL_dMt[2][0] - dL_dMt[0][2]) + 2 * x * (dL_dMt[1][2] - dL_dMt[2][1]);
dL_dq.y = 2 * y * (dL_dMt[1][0] + dL_dMt[0][1]) + 2 * z * (dL_dMt[2][0] + dL_dMt[0][2]) + 2 * r * (dL_dMt[1][2] - dL_dMt[2][1]) - 4 * x * (dL_dMt[2][2] + dL_dMt[1][1]);
dL_dq.z = 2 * x * (dL_dMt[1][0] + dL_dMt[0][1]) + 2 * r * (dL_dMt[2][0] - dL_dMt[0][2]) + 2 * z * (dL_dMt[1][2] + dL_dMt[2][1]) - 4 * y * (dL_dMt[2][2] + dL_dMt[0][0]);
dL_dq.w = 2 * r * (dL_dMt[0][1] - dL_dMt[1][0]) + 2 * x * (dL_dMt[2][0] + dL_dMt[0][2]) + 2 * y * (dL_dMt[1][2] + dL_dMt[2][1]) - 4 * z * (dL_dMt[1][1] + dL_dMt[0][0]);

// Gradients of loss w.r.t. unnormalized quaternion
float4* dL_drot = (float4*)(dL_drots + idx);
*dL_drot = float4{ dL_dq.x, dL_dq.y, dL_dq.z, dL_dq.w };//dnormvdv(float4{ rot.x, rot.y, rot.z, rot.w }, float4{ dL_dq.x, dL_dq.y, dL_dq.z, dL_dq.w });
```

四元数 q 对应的旋转矩阵 R 的梯度较为复杂，需要考虑每个元素对四元数的偏导数。为了简化表示，我们定义旋转矩阵的元素 R_{ij} 对四元数 q 的梯度为 $\frac{\partial R_{ij}}{\partial q_k}$。
假设 dL_dM_{ij} 是 dL_dM 中的第 i 行第 j 列元素，则四元数梯度的每个分量可以表示为：

$$
\frac{\partial L}{\partial q_x} = \sum_{i,j} \frac{\partial L}{\partial M_{ij}} \cdot \frac{\partial M_{ij}}{\partial q_x}
$$
$$
\frac{\partial L}{\partial q_y} = \sum_{i,j} \frac{\partial L}{\partial M_{ij}} \cdot \frac{\partial M_{ij}}{\partial q_y}
$$
$$
\frac{\partial L}{\partial q_z} = \sum_{i,j} \frac{\partial L}{\partial M_{ij}} \cdot \frac{\partial M_{ij}}{\partial q_z}
$$
$$
\frac{\partial L}{\partial q_w} = \sum_{i,j} \frac{\partial L}{\partial M_{ij}} \cdot \frac{\partial M_{ij}}{\partial q_w}
$$

最终的梯度表达式代码中给出的四元数梯度可以由下式表示:

$$
\begin{gathered}
\frac{\partial L}{\partial q_x} = 2z\left((\frac{\partial L}{\partial M})^T[0][1] - (\frac{\partial L}{\partial M})^T[1][0]\right) + 2y\left((\frac{\partial L}{\partial M})^T[2][0] - (\frac{\partial L}{\partial M})^T[0][2]\right) + \\
2x\left((\frac{\partial L}{\partial M})^T[1][2] - (\frac{\partial L}{\partial M})^T[2][1]\right) \\
\frac{\partial L}{\partial q_y} = 2y\left((\frac{\partial L}{\partial M})^T[1][0] + (\frac{\partial L}{\partial M})^T[0][1]\right) + 2z\left((\frac{\partial L}{\partial M})^T[2][0] + (\frac{\partial L}{\partial M})^T[0][2]\right) + \\
2r\left((\frac{\partial L}{\partial M})^T[1][2] - (\frac{\partial L}{\partial M})^T[2][1]\right) - 4x\left((\frac{\partial L}{\partial M})^T[2][2] + (\frac{\partial L}{\partial M})^T[1][1]\right) \\
\frac{\partial L}{\partial q_z} = 2x\left((\frac{\partial L}{\partial M})^T(\frac{\partial L}{\partial M})^T[1][0] + (\frac{\partial L}{\partial M})^T[0][1]\right) + 2r\left((\frac{\partial L}{\partial M})^T[2][0] - (\frac{\partial L}{\partial M})^T[0][2]\right) + \\
2z\left((\frac{\partial L}{\partial M})^T[1][2] + (\frac{\partial L}{\partial M})^T[2][1]\right) - 4y\left((\frac{\partial L}{\partial M})^T[2][2] + (\frac{\partial L}{\partial M})^T[0][0]\right) \\
\frac{\partial L}{\partial q_w} = 2r\left((\frac{\partial L}{\partial M})^T[0][1] - (\frac{\partial L}{\partial M})^T[1][0]\right) + 2x\left((\frac{\partial L}{\partial M})^T[2][0] + (\frac{\partial L}{\partial M})^T[0][2]\right) + \\
2y\left((\frac{\partial L}{\partial M})^T[1][2] + (\frac{\partial L}{\partial M})^T[2][1]\right) - 4z\left((\frac{\partial L}{\partial M})^T[1][1] + (\frac{\partial L}{\partial M})^T[0][0]\right)
\end{gathered}
$$

再把结果进行存储:

$$
\frac{\partial L}{\partial rot} = \begin{pmatrix} \frac{\partial L}{\partial q_x} \\ \frac{\partial L}{\partial q_y} \\ \frac{\partial L}{\partial q_z} \\ \frac{\partial L}{\partial q_w} \end{pmatrix}
$$

损失函数对三维均值的梯度

```
// Compute loss gradient w.r.t. 3D means due to gradients of 2D means
// from rendering procedure
glm::vec3 dL_dmean;
float mul1 = (proj[0] * m.x + proj[4] * m.y + proj[8] * m.z + proj[12]) * m_w * m_w;
float mul2 = (proj[1] * m.x + proj[5] * m.y + proj[9] * m.z + proj[13]) * m_w * m_w;
dL_dmean.x = (proj[0] * m_w - proj[3] * mul1) * dL_dmean2D[idx].x + (proj[1] * m_w - proj[3] * mul2) * dL_dmean2D[idx].y;
dL_dmean.y = (proj[4] * m_w - proj[7] * mul1) * dL_dmean2D[idx].x + (proj[5] * m_w - proj[7] * mul2) * dL_dmean2D[idx].y;
dL_dmean.z = (proj[8] * m_w - proj[11] * mul1) * dL_dmean2D[idx].x + (proj[9] * m_w - proj[11] * mul2) * dL_dmean2D[idx].y;
```

为了清楚理解，我们首先从三维点经过投影矩阵得到二维点的过程开始，然后利用链式法则计算梯度。

假设有一个3D点 $\mathbf{m} = (m_x, m_y, m_z)$，通过一个投影矩阵 $\mathbf{P}$ 将其投影到2D空间。投影矩阵 $\mathbf{P}$ 是一个4x4矩阵，包含视图和透视变换。齐次坐标变换后的结果是:

$$
\mathbf{m}_{hom} = \mathbf{P} \cdot \begin{pmatrix} m_x \\ m_y \\ m_z \\ 1 \end{pmatrix}
$$

$$
\mathbf{m}_{hom} = \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{pmatrix} \begin{pmatrix} m_x \\ m_y \\ m_z \\ 1 \end{pmatrix} = \begin{pmatrix} P_{00}m_x + P_{01}m_y + P_{02}m_z + P_{03} \\ P_{10}m_x + P_{11}m_y + P_{12}m_z + P_{13} \\ P_{20}m_x + P_{21}m_y + P_{22}m_z + P_{23} \\ P_{30}m_x + P_{31}m_y + P_{32}m_z + P_{33} \end{pmatrix}
$$

为了得到规范化设备坐标，我们进行透视除法:

$$\mathbf{m}_{ndc}=\begin{pmatrix}x_{ndc}\\y_{ndc}\\z_{ndc}\end{pmatrix}=\begin{pmatrix}\frac{P_{00}m_x+P_{01}m_y+P_{02}m_z+P_{03}}{P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33}}\\\frac{P_{10}m_x+P_{11}m_y+P_{12}m_z+P_{13}}{P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33}}\\\frac{P_{20}m_x+P_{21}m_y+P_{22}m_z+P_{23}}{P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33}}\end{pmatrix}$$

令 $m_w=\frac{1}{P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33}}$，我们有：

$$\begin{aligned}x_{ndc}&=m_w(P_{00}m_x+P_{01}m_y+P_{02}m_z+P_{03})\\y_{ndc}&=m_w(P_{10}m_x+P_{11}m_y+P_{12}m_z+P_{13})\end{aligned}$$

我们知道损失函数关于2D均值的梯度 $\frac{dL}{dmean2D}$，我们希望通过链式法则计算损失函数关于3D均值的梯度 $\frac{dL}{dmean}$。
损失函数 L 关于3D均值 $mean.x$ 的梯度可以通过链式法则表示为：

$$\frac{dL}{dmean.x}=\frac{dL}{dmean2D.x}\cdot\frac{dmean2D.x}{dmean.x}+\frac{dL}{dmean2D.y}\cdot\frac{dmean2D.y}{dmean.x}$$

我们需要计算 $\frac{dmean2D.x}{dmean.x}$ 和 $\frac{dmean2D.y}{dmean.x}$。

从上述投影过程可以看出：

$$\begin{aligned}mean2D.x&=m_w(P_{00}m_x+P_{01}m_y+P_{02}m_z+P_{03})\\mean2D.y&=m_w(P_{10}m_x+P_{11}m_y+P_{12}m_z+P_{13})\end{aligned}$$

对 $mean2D.x$ 关于 $mean.x$ 求导：
m_w 也是 m_x 的函数，因此需要用到乘积法则。首先，我们对 m_w 求导：

$$\begin{aligned}\frac{d(m_w)}{dmean.x}&=\frac{d}{dmean.x}\left(\frac{1}{P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33}}\right)\\&=-\frac{1}{(P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33})^2}\cdot P_{30}=-m_w^2\cdot P_{30}\end{aligned}$$

利用求导乘积法则，我们得到：

$$\begin{aligned}\frac{d(mean2D.x)}{d(mean.x)}&=m_w\cdot\frac{d(P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33})}{dmean.x}+\frac{d(m_w)}{dmean.x}\cdot(P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33})\\&=m_w\cdot P_{00}-m_w^2\cdot(P_{30}m_x+P_{31}m_y+P_{32}m_z+P_{33})\cdot P_{30}\\&=m_w\cdot P_{00}-P_{30}\cdot mean2D.x\cdot m_w\end{aligned}$$

对 $mean2D.y$ 关于 $mean.x$ 求导：

$$\frac{d(mean2D.y)}{d(mean.x)}=m_w\cdot P_{10}-P_{30}\cdot mean2D.y\cdot m_w$$

因此，损失函数关于3D均值的梯度可以表示为：

$$\frac{dL}{dmean.x}=(m_w\cdot P_{00}-P_{30}\cdot mean2D.x\cdot m_w)\cdot\frac{dL}{dmean2D.x}+(m_w\cdot P_{10}-P_{30}\cdot mean2D.y\cdot m_w)\cdot\frac{dL}{dmean2D.y}$$

由于：

$$\mathrm{mul1}=(P_{00}m_x+P_{01}m_y+P_{02}m_z+P_{03})\cdot m_w^2$$

我们有：

$$\frac{dL}{dmean.x}=(P_{00}\cdot m_w-P_{30}\cdot mul1)\cdot\frac{dL}{dmean2D.x}+(P_{10}\cdot m_w-P_{30}\cdot mul2)\cdot\frac{dL}{dmean2D.y}$$

类似地，可以推导出损失函数关于3D均值 $mean.y$ 和 $mean.z$ 的梯度：

$$\begin{aligned}\frac{dL}{dmean.y}&=(P_{01}\cdot m_w-P_{31}\cdot mul1)\cdot\frac{dL}{dmean2D.x}+(P_{11}\cdot m_w-P_{31}\cdot mul2)\cdot\frac{dL}{dmean2D.y}\\\frac{dL}{dmean.z}&=(P_{02}\cdot m_w-P_{32}\cdot mul1)\cdot\frac{dL}{dmean2D.x}+(P_{12}\cdot m_w-P_{32}\cdot mul2)\cdot\frac{dL}{dmean2D.y}\end{aligned}$$

最终结果与代码中的公式一致：

$$\frac{dL}{dmean.x} = (P_{00} \cdot m_w - P_{30} \cdot mul1) \cdot \frac{dL}{dmean2D.x} + (P_{10} \cdot m_w - P_{30} \cdot mul2) \cdot \frac{dL}{dmean2D.y}$$
$$\frac{dL}{dmean.y} = (P_{01} \cdot m_w - P_{31} \cdot mul1) \cdot \frac{dL}{dmean2D.x} + (P_{11} \cdot m_w - P_{31} \cdot mul2) \cdot \frac{dL}{dmean2D.y}$$
$$\frac{dL}{dmean.z} = (P_{02} \cdot m_w - P_{32} \cdot mul1) \cdot \frac{dL}{dmean2D.x} + (P_{12} \cdot m_w - P_{32} \cdot mul2) \cdot \frac{dL}{dmean2D.y}$$

前向传播的渲染部分

初始化透明度累计值，1.0代表完全不透明，再把每个颜色通道的颜色都清零：$T = 1.0$，$C[ch] = 0$。

对于每个高斯点j:

计算距离矢量d:

$$d = xy - pixf$$

计算二次型幂次$power$:

$$power = -0.5(con\_o.x \cdot d.x^2 + con\_o.z \cdot d.y^2) - con\_o.y \cdot d.x \cdot d.y$$

计算高斯函数G:

$$G = e^{power} = e^{-0.5(con\_o.x \cdot d.x^2 + con\_o.z \cdot d.y^2) - con\_o.y \cdot d.x \cdot d.y}$$

计算alpha值

$$\alpha = min(0.99, con\_o.w \cdot G)$$

检查α值是否太小: $if(\alpha < \frac{1.0}{255.0})$
更新透明度T的候选值:

$$test\_T = T \cdot (1 - \alpha)$$

检查透明度是否接近零: $if(test\_T < 0.0001)$
更新颜色值,$f[ch]$是第j个高斯点的第ch个通道的特征值，即$features[collected_id[j] \cdot CHANNELS + ch]$:

$$C[ch] = C[ch] + \alpha \cdot T \cdot f[ch]$$

反向传播的渲染部分（$alpha$、颜色、二维高斯位置与协方差的梯度）

对颜色的梯度

颜色递推公式

在前向传播中，假设我们有多个高斯函数，每个高斯函数的颜色用c_i表示，第i 个高斯函数的透明度用α_i表示。设最终的像素颜色为C，递推公式如下:

$$C_i = \alpha_i c_i + (1 - \alpha_i) C_{i+1}$$

其中C_i 表示第i个高斯函数及其后的所有高斯函数共同贡献的颜色。

透明度递推公式

设T_i 表示第i个高斯函数对像素点的透光率（即前i个高斯函数的透明度的累积），则有：

$$T_i = \prod_{j=1}^{i}(1-\alpha_j)$$

颜色梯度推导

假设我们已经计算出像素的损失L，现在我们需要计算每个高斯函数的颜色梯度，即$\frac{\partial L}{\partial c_i}$。
为了推导这个梯度，我们需要使用链式法则从最终的损失L传递回每个高斯函数的颜色c_i。

1. **递推颜色的梯度**

从最终的颜色 C开始，我们可以递推每个高斯函数的颜色梯度。

$$\frac{\partial L}{\partial C_i} = \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial C_i}$$

其中：

$$\frac{\partial C}{\partial C_i} = \prod_{k=i+1}^{n}(1-\alpha_k)$$

因此，有：

$$\frac{\partial L}{\partial C_i} = \frac{\partial L}{\partial C} \cdot \prod_{k=i+1}^{n}(1-\alpha_k) = T_i \frac{\partial L}{\partial C}$$

2. **计算颜色梯度**

使用链式法则，我们可以将损失对颜色c_i的梯度分解为损失对最终颜色C 的梯度以及颜色递推公式中的各个分量：

$$\frac{\partial L}{\partial c_i} = \frac{\partial L}{\partial C_i} \cdot \frac{\partial C_i}{\partial c_i}$$

其中：

$$\frac{\partial C_i}{\partial c_i} = \alpha_i$$

因此，有：

$$\frac{\partial L}{\partial c_i} = T_i \frac{\partial L}{\partial C} \cdot \alpha_i = \alpha_i T_i \frac{\partial L}{\partial C}$$

总结

最终的梯度公式如下：

$$\frac{\partial L}{\partial c_i} = \alpha_i T_i \frac{\partial L}{\partial C}$$

$\alpha = min(0.99, con\_o.w \cdot G)$
$T = \frac{T_{final}}{\prod_{k=0}^{n-1}(1-\alpha)}$

那么颜色的梯度就是：

$$\frac{\partial L}{\partial color} = \alpha \cdot T \cdot \frac{\partial L}{\partial channel}$$

这里的$\partial channel$是在处理像素的时候对像素的通道颜色值进行采样获得的。

对$alpha$的梯度

$$dL\_dalpha = \frac{\partial L}{\partial alpha} = T \cdot \sum_{ch=0}^{C-1} (c - accum\_rec[ch]) \cdot \frac{\partial L}{\partial channel}$$

如果考虑背景颜色的影响，就再加上一个乘数$bg_color[i]$

高斯分布的衰减指数$G = e^{power}$的梯度

$$\frac{\partial L}{\partial G} = con\_o.w \cdot \frac{\partial L}{\partial alpha}$$

二维高斯分布的均值的梯度

$$\begin{aligned}
\frac{\partial L}{\partial mean2D.x} &= \frac{\partial L}{\partial G} \cdot (-G \cdot d.x \cdot con_o.x - G \cdot d.y \cdot con_o.y) \cdot 0.5W \\
\frac{\partial L}{\partial mean2D.y} &= \frac{\partial L}{\partial G} \cdot (-G \cdot d.y \cdot con_o.y - G \cdot d.x \cdot con_o.x) \cdot 0.5H
\end{aligned}$$

二维高斯分布的协方差的梯度

首先回顾一下二维高斯分布的密度函数：

$$G(x, y) = \exp\left(-\frac{1}{2}\mathbf{d}^T \Sigma^{-1} \mathbf{d}\right)$$

其中，$\mathbf{d} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$ 是位置偏移向量，Σ 是协方差矩阵。假设协方差矩阵为：

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

其中，我们可以定义三个变量：

$$a = \sigma_x^2, \quad b = \rho\sigma_x\sigma_y, \quad c = \sigma_y^2$$

高斯分布的梯度

为了计算协方差矩阵 (\Sigma) 的梯度，我们需要求出$\frac{\partial G}{\partial a}$、$\frac{\partial G}{\partial b}$ 和$frac\partial G\partial c$ 的值。

计算$\frac{\partial G}{\partial a}$

对$a = \sigma_x^2$求导：

$$\frac{\partial}{\partial a}\left(-\frac{1}{2}\mathbf{d}^T \Sigma^{-1} \mathbf{d}\right) = -\frac{1}{2} \cdot \frac{\partial}{\partial a}(\mathbf{d}^T \Sigma^{-1} \mathbf{d})$$

假设Σ^{-1}的元素为：

$$\Sigma^{-1} = \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$$

那么：

$$\mathbf{d}^T \Sigma^{-1} \mathbf{d} = \alpha d_x^2 + 2\beta d_x d_y + \gamma d_y^2$$

对$a = \sigma_x^2$ 求导：

$$\frac{\partial}{\partial a}\left(\alpha d_x^2+2\beta d_x d_y+\gamma d_y^2\right)=\frac{\partial\alpha}{\partial a}d_x^2+2\frac{\partial\beta}{\partial a}d_x d_y+\frac{\partial\gamma}{\partial a}d_y^2$$

由于Σ^{-1} 的具体形式比较复杂，我们可以简化计算，直接使用导数性质得到：

$$\frac{\partial G}{\partial a}=-0.5\cdot G\cdot\frac{\partial}{\partial a}\left(\mathbf{d}^T\Sigma^{-1}\mathbf{d}\right)$$

经过推导可得：

$$\frac{\partial G}{\partial a}=-0.5\cdot G\cdot d_x^2$$

计算$\frac{\partial G}{\partial b}$

对$b=\rho\sigma_x\sigma_y$ 求导：

$$\frac{\partial}{\partial b}\left(-\frac{1}{2}\mathbf{d}^T\Sigma^{-1}\mathbf{d}\right)=-0.5\cdot\frac{\partial}{\partial b}\left(\alpha d_x^2+2\beta d_x d_y+\gamma d_y^2\right)$$

我们得到：

$$\frac{\partial G}{\partial b}=-0.5\cdot G\cdot d_x d_y$$

计算$\frac{\partial G}{\partial c}$

对$c=\sigma_y^2$求导：

$$\frac{\partial}{\partial c}\left(-\frac{1}{2}\mathbf{d}^T\Sigma^{-1}\mathbf{d}\right)=-0.5\cdot\frac{\partial}{\partial c}\left(\alpha d_x^2+2\beta d_x d_y+\gamma d_y^2\right)$$

我们得到：

$$\frac{\partial G}{\partial c}=-0.5\cdot G\cdot d_y^2$$

总结上述推导结果，我们可以得到：

$$\frac{\partial L}{\partial a}=\frac{\partial L}{\partial G}\cdot\frac{\partial G}{\partial a}=-0.5\cdot d_x^2\cdot G\cdot\frac{\partial L}{\partial G}$$
$$\frac{\partial L}{\partial b}=\frac{\partial L}{\partial G}\cdot\frac{\partial G}{\partial b}=-0.5\cdot d_x d_y\cdot G\cdot\frac{\partial L}{\partial G}$$
$$\frac{\partial L}{\partial c}=\frac{\partial L}{\partial G}\cdot\frac{\partial G}{\partial c}=-0.5\cdot d_y^2\cdot G\cdot\frac{\partial L}{\partial G}$$

代入具体变量，可以得到：

$$\frac{\partial L}{\partial \text{conic2D.x}}=-0.5\cdot(G\cdot d_x)\cdot d_x\cdot\frac{\partial L}{\partial G}$$
$$\frac{\partial L}{\partial \text{conic2D.y}}=-0.5\cdot(G\cdot d_x)\cdot d_y\cdot\frac{\partial L}{\partial G}$$
$$\frac{\partial L}{\partial \text{conic2D.w}}=-0.5\cdot(G\cdot d_y)\cdot d_y\cdot\frac{\partial L}{\partial G}$$

这就是最终的梯度公式。

不透明度的梯度

$$\frac{\partial L}{\partial opacity} = G \cdot \frac{\partial L}{\partial alpha}$$