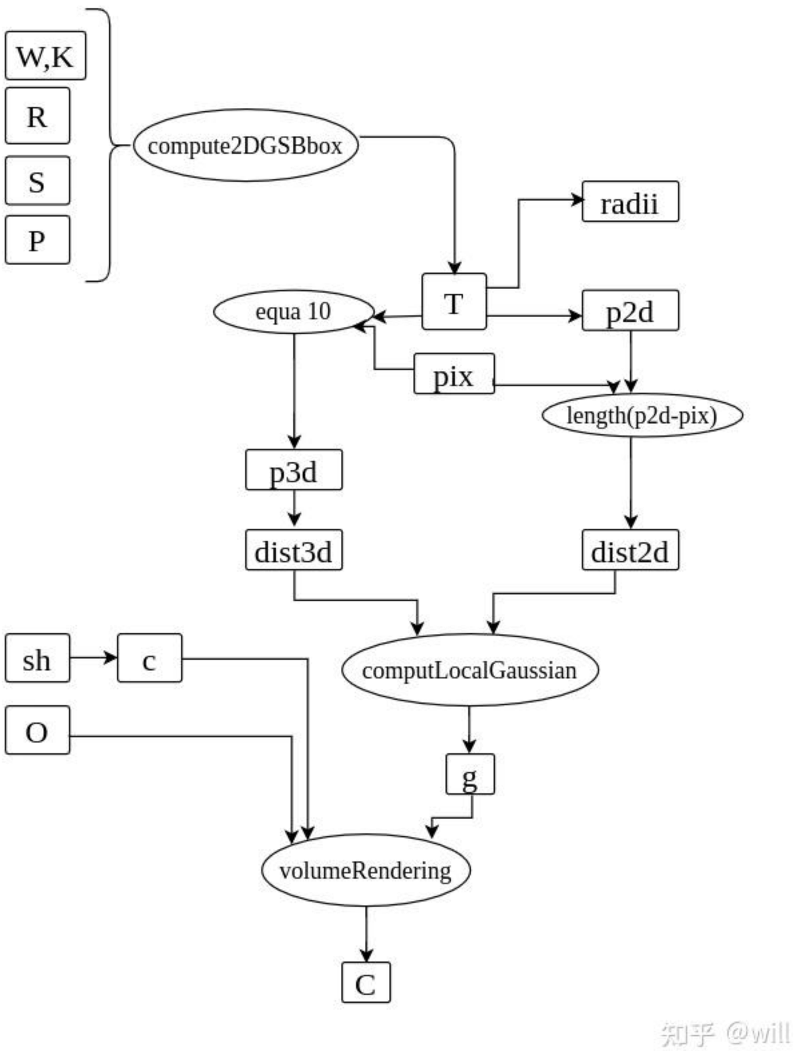
2DGS的非官方实现以及相关推导

本文为个人根据官方[python demo]在cuda上实现的2DGS以及相关推导，知乎上公式不太方便编辑，更规整的排班可以看git项目里的pdf

github：[GitHub - will-zzy/2dgs-non-official: This code is a non-official 2DGS implementation including forward and backward process of 2DGS with cuda.]

2DGS

计算图如下：



前向

> 初始化t_u，t_v：

利用open3d对稀疏点云求法向，得到单位向量t_n

> 在训练过程中2D椭圆图元用四元数q与两个尺度因子s_u，s_v表达，其中q代表的旋转矩阵的第三列初始化为t_n，前两列通过正交化得到，在后续的优化过程中对传到前两列的梯度计算并传递到四元数与scale上

compute2DGSBBox.forward

2DGS在计算图像上高斯的投影点`point_image`和`radii`时和3DGS不一样，2DGS不是forward地将高斯投影到图像上，而是通过在图像上划定一个bbox `B_1`，将其映射到高斯局部平面上，得到另一个bbox `B_2`（`B_2`不一定是长方形，但一定是平行四边型），通过约束`B_2`的各边到原点的距离为1，计算出`B_1`的`x_1,x_2,y_1,y_2`，并取`B_1`的中心为投影点，过程如下：

$$W = W2C, \quad H = \begin{bmatrix} s_u t_u & s_v t_v & 0 & p_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & 0 c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$T = (KWH)^T = [t_0 \quad t_1 \quad t_2 \quad t_2]_{3\times 4}$$

代码中的`T`即为公式中的T，均为 3×4的矩阵（省略法向方向的变换）；

由论文，KWH本身将高斯局部平面点的齐次坐标映射到相机平面上，则(KWH)^T将相机平面上的**平面齐次坐标** 映射到高斯局部平面上。

2DGS在相机平面通过x-plane和y-plane确定像素点(x,y)，分别将x-plane和y-plane通过(KWH)^T映射到局部平面上，两个面相交的线与局部平面上的交点即为“采样点”

设像素坐标系为xy坐标系，高斯局部坐标系为uv坐标系，在xy坐标系下的x,y平面齐次坐标分别为：

$$hx = \begin{bmatrix} -1 \\ 0 \\ 0 \\ x \end{bmatrix}, hy = \begin{bmatrix} 0 \\ -1 \\ 0 \\ y \end{bmatrix}$$

则T右乘hx,hy可将xy下的平面齐次坐标映射到uv下，即（上标表示向量第几个元素）：

$$[t_0 \quad t_1 \quad t_2 \quad t_3] h_x = \begin{bmatrix} -t_0^0 + t_3^0 x \\ -t_0^1 + t_3^1 x \\ -t_0^2 + t_3^2 x \end{bmatrix} = h_u$$

应用距离公式：

$$\frac{|h_u^2|}{\sqrt{(h_u^0)^2+(h_u^1)^2}} = 1$$

> 注意aligned axis的x-plane变换到uv后不一定aligned到uv axis；因此需要用距离公式做约束，得到一个一元二次方程

两边平方移项可得：

$$(t_0^0)^2 + (t_0^1)^2 - (t_0^2)^2 + x^2[(t_3^0)^2 + (t_3^1)^2 - (t_3^2)^2] - 2x(t_0^0 t_3^0 + t_0^1 t_3^1 - t_0^2 t_3^2) = 0$$

根 x_1, x_2分别表示了uv平面上bbox的u-plane映射到xy平面上x-plane的位置，则`center`可表示为两根之和/2，也即：

$$\frac{x_1+x_2}{2} = -\frac{b}{a} = \frac{t_0^0 t_3^0 + t_0^1 t_3^1 - t_0^2 t_3^2}{(t_3^0)^2+(t_3^1)^2-(t_3^2)^2}$$

两根之差除以2= $\sqrt{b^2-4ac}/2a$即为图像上bbox的半径，也即splatting过程中的`my_radius`

> **为什么不直接用T.t将(0,0,1,1)^T变换到相机平面得到中心点?**
>
> 因为投影变换不是仿射的，空间gs椭球/椭圆面投影到图像上不一定是对称的，2dgs使用backward的采样方式实际上是为了规避雅克比近似带来的误差，因此在确定相机平面上2dgs的center时不能直接用透视变换投影2dgs质心，而要用**用bbox确定中心** 保证采样的对称性

通过以上方法得到**投影点** 和**半径** 后，即可使用3D gaussians splatting的tile-based排序方法

如果距离为3（3\sigma）

则：

$$\frac{|h_u^2|}{\sqrt{(h_u^0)^2+(h_u^1)^2}} = 3$$

$$9(t_0^0)^2 + 9(t_0^1)^2 - (t_0^2)^2 + x^2[9(t_3^0)^2 + 9(t_3^1)^2 - (t_3^2)^2] - 2x(9t_0^0 t_3^0 + 9t_0^1 t_3^1 - t_0^2 t_3^2$$

如果有滤波，则需要计算高斯在图像上的投影点：

滤波的前向为 $KWH \Rightarrow p_{2d}$，注意 $T_t = KWH$，由于K最后一行是齐次位，故T_t第四行和第三行一样

$$T_t = \begin{bmatrix} T_{00} & T_{01} & T_{02} \\ T_{10} & T_{11} & T_{12} \\ T_{20} & T_{21} & T_{22} \\ T_{20} & T_{21} & T_{22} \end{bmatrix}$$

$$a = \sigma^2 T_{20}^2 + \sigma^2 T_{21}^2 - T_{22}^2$$

$$b = -2(\sigma^2 T_{00} T_{02} + \sigma^2 T_{01} T_{12} - T_{02} T_{22})$$

$$c = \sigma^2 T_{00}^2 + \sigma^2 T_{01}^2 - T_{02}^2$$

$$ax^2 + bx + c = 0$$

则两根之和

$p_{2d} \rightarrow x = \frac{(x_1+x_2)}{2} = -\frac{b}{2a} = \frac{T_{00}T_{02}+T_{10}T_{12}-T_{20}T_{22}}{T_{02}^2+T_{12}^2-T_{22}^2}$

computeLocalGaussian.forward

该函数目的是知道像素点与某个高斯参数，获得该高斯对像素点的权重。该过程有两种方式，第一种发生在**三维空间**，也即在2DGS局部空间上：将像素点变换到uv坐标系，查询高斯权重；第二种发生在**二维空间**，也即在相机平面上：高斯投影点对该像素点的贡献（也即文中的滤波），如果有滤波，则输入需要加入高斯的投影点`point_image`

输入：`KWH_t`（`3x4`），`pix_xy`，`point_image`

三维空间：

$$Tt = KWH_{4\times3}$$

$$k = -Tt_{0,:} + xTt_{3,:} = \begin{bmatrix} -T_{00} + xT_{30} \\ -T_{01} + xT_{31} \\ -T_{02} + xT_{32} \end{bmatrix}$$

$$l = -Tt_{1,:} + yTt_{3,:} = \begin{bmatrix} -T_{10} + xT_{30} \\ -T_{11} + xT_{31} \\ -T_{12} + xT_{32} \end{bmatrix}$$

$$p = k \times l$$

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} k_2 l_3 - k_3 l_2 \\ k_3 l_1 - k_1 l_3 \\ k_1 l_2 - k_2 l_1 \end{bmatrix}$$

$$d = (\frac{p^1}{p^3})^2 + (\frac{p^2}{p^3})^2$$

$$g = \exp\left(-\frac{d}{2}\right)$$

二维空间：

令投影点为 p_{2d}

$$\hat{g} = \exp\left(-\frac{(p_{2d} - \begin{bmatrix} x \\ y \end{bmatrix})^2}{2\sigma^2}\right), \quad \sigma = \frac{\sqrt{2}}{2}$$

反向

对于

$\hat{c} = \sum_{i=1}^{N} c_i \bar{\alpha}_i T_i \quad \text{where } \bar{\alpha}_i = \alpha_i g_i, T_i = \prod_{j=1}^{i-1}(1 - \bar{\alpha}_j)$

梯度传递路径如下：

$$\begin{cases} c_i \to sh_i \\ \alpha_i \to o_i \\ T_i \to \begin{cases} \alpha_j \to o_j \\ g_j \to R, S \end{cases} \\ g_i \to R, S \end{cases}$$

则梯度 $\frac{\partial L}{\partial \alpha}, \frac{\partial L}{\partial g}$ 计算如下：

$$\begin{aligned} \frac{\partial \hat{c}}{\partial \bar{\alpha}_i} &= c_i T_i - \frac{\sum_{j=i+1}^{N} c_j \bar{\alpha}_j T_j}{1 - \bar{\alpha}_i} \\ &= (c_i - \frac{\sum_{j=i+1}^{N} c_j \bar{\alpha}_j T_j}{T_{i+1}}) T_i \\ &= (c_i - A_i) T_i \end{aligned}$$

其中A_i是有递推公式的，因此可以节省计算时间

$$\begin{aligned} A_i &= \frac{\sum_{j=i+1}^{N} c_j \bar{\alpha}_j T_j}{T_{i+1}} \\ &= c_{i+1} \bar{\alpha}_{i+1} + \frac{\sum_{j=i+2}^{N} c_j \bar{\alpha}_j T_j}{T_{i+1}} \\ &= c_{i+1} \bar{\alpha}_{i+1} + \frac{\sum_{j=i+2}^{N} c_j \bar{\alpha}_j T_j}{T_{i+2}} * (1 - \bar{\alpha}_{i+1}) \\ &= c_{i+1} \bar{\alpha}_{i+1} + A_{i+1}(1 - \bar{\alpha}_{i+1}) \end{aligned}$$

> A即代码中的`accum_rec`

则可以从后往前递推 A_i以及 $\partial L / \partial \bar{\alpha}_i$

从而 $\partial L/\partial \alpha_i = \partial L/\partial \bar{\alpha}_i * g_i$， $\partial L/\partial g_i = \partial L/\partial \bar{\alpha}_i * \alpha_i$

当xy平面上的投影点 p_{2d}到像素点 p_I的距离 $\hat{d}$小于 d时，则使用xy平面的高斯投影点计算高斯权重，此时会产生`dL_dmean2D`以及对 KWH_t的梯度

$$\hat{d} = p_I - p_{2d}$$

$$g = \exp\left(-\frac{\hat{d}^2}{2\sigma^2}\right)$$

从而：

$$\frac{dL}{dp_{2d}} = \begin{bmatrix} -\frac{1}{\sigma^2} L_g g(p_{2d}.x - p_I.x) \\ -\frac{1}{\sigma^2} L_g g(p_{2d}.y - p_I.y) \end{bmatrix}$$

computeLocalGaussian.backward

由g到T

由dL_dp2d到T

计算高斯权重：

$$\frac{dL}{dg_i} = \alpha_i \frac{dL}{d\bar{\alpha}_i}$$

如果有滤波，由于投影点是由bbox的中心点得到的，因此存在一条由`dL_dg`到`dL_dp2d`到`dL_dT`的传播路径

投影点计算公式为：

$$p_{2d} \to x = \frac{(x_1 + x_2)}{2} = -\frac{b_x}{2a} = \frac{T_{00}T_{20} + T_{01}T_{21} - T_{02}T_{22}}{T_{20}^2 + T_{21}^2 - T_{22}^2}$$

$$p_{2d} \to y = \frac{(y_1 + y_2)}{2} = -\frac{b_y}{2a} = \frac{T_{10}T_{20} + T_{11}T_{21} - T_{12}T_{22}}{T_{20}^2 + T_{21}^2 - T_{22}^2}$$

令

$$a = \sigma^2 T_{20}^2 + \sigma^2 T_{21}^2 - T_{22}^2$$

$$b_x = -2(\sigma^2 T_{00}T_{20} + \sigma^2 T_{01}T_{21} - T_{02}T_{22})$$

$$b_y = -2(\sigma^2 T_{10}T_{20} + \sigma^2 T_{11}T_{21} - T_{12}T_{22})$$

从而

$$\frac{dL}{da} = L^0_{p_{2d}}\frac{dp_{2d}.x}{da} + L^1_{p_{2d}}\frac{dp_{2d}.y}{da} = L^0_{p_{2d}}\frac{b_x}{2a^2} + L^1_{p_{2d}}\frac{b_y}{2a^2}$$

$$\frac{dL}{db_x} = -L^0_{p_{2d}}\frac{1}{2a}$$

$$\frac{dL}{db_y} = -L^1_{p_{2d}}\frac{1}{2a}$$

由于c不参与投影点计算，故

$$\frac{dL}{dT} = \frac{dL}{da}\frac{da}{dT} + \frac{dL}{db_x}\frac{db_x}{dT} + \frac{dL}{db_y}\frac{db_y}{dT}$$

$$= \frac{dL}{da}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2\sigma^2 T_{20} & 2\sigma^2 T_{21} & -2T_{22} \end{bmatrix} + \frac{dL}{db_x}\begin{bmatrix} -2\sigma^2 T_{20} & -2\sigma^2 T_{21} & 2T_{22} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2\sigma^2 T_{00} & -2\sigma^2 T_{01} & 2T_{02} \end{bmatrix} +$$

另一条路径是由`dL_dg`到`dL_dp3d`到`dL_dT`：

$$\frac{dg_i}{dd_i} = -\frac{g}{2}$$

$$\frac{dd_i}{dp} = \begin{bmatrix} 2p_1/p_3^2 \\ 2p_2/p_3^2 \\ -2((p_1^2 + p_2^2)/p_3^3) \end{bmatrix}$$

$$\frac{\partial p}{\partial k} = \begin{bmatrix} 0 & l_3 & -l_2 \\ -l_3 & 0 & l_1 \\ l_2 & -l_1 & 0 \end{bmatrix}$$

$$\frac{\partial p}{\partial l} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix}$$

$$\frac{dk}{dTt_{0,:}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad \frac{dk}{dTt_{3,:}} = \begin{bmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & x \end{bmatrix}$$

$$\frac{dl}{dTt_{1,:}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad \frac{dl}{dTt_{3,:}} = \begin{bmatrix} y & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & y \end{bmatrix}$$

从而

$$\begin{aligned}
\frac{dL}{dT_{0,:}} &= \frac{dL}{dd}\frac{dd}{dp}\frac{dp}{dT_{0,:}} \\
&= \frac{dL}{dd}\frac{dd}{dp}\begin{bmatrix} 0 & -T_{12}+yT_{32} & T_{11}-yT_{31} \\ T_{12}-yT_{32} & 0 & -T_{10}+yT_{30} \\ -T_{11}+yT_{31} & T_{10}-yT_{30}y & 0 \end{bmatrix} \\
&= \frac{dL}{dd}\frac{dd}{dp}\begin{bmatrix} 0 & l_3 & -l_2 \\ -l_3 & 0 & l_1 \\ l_2 & -l_1 & 0 \end{bmatrix} \\
&= \frac{dL}{dd}\begin{bmatrix} dp_2l_3 - dp_3l_2 \\ -dp_1l_3 + dp_3l_1 \\ dp_1l_2 - dp_2l_1 \end{bmatrix}
\end{aligned}$$

同理

$$\begin{aligned}
\frac{dL}{dT_{1,:}} &= \frac{dL}{dd}\frac{dd}{dp}\frac{dp}{dT_{1,:}} \\
\\
&= \frac{dL}{dd}\frac{dd}{dp}\begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \\
&= \frac{dL}{dd}\begin{bmatrix} -dp_2k_3 + dp_3k_2 \\ dp_1k_3 - dp_3k_1 \\ -dp_1k_2 + dp_2k_1 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\frac{dL}{dT_{3,:}} &= \frac{dL}{dd}\frac{dd}{dp}\frac{dp}{dT_{3,:}} \\
\\
&= \frac{dL}{dd}\frac{dd}{dp}\begin{bmatrix} 0 & -x(-T_{12}+yT_{32})+y \\ x(-T_{12}+yT_{32})-y(-T_{02}+xT_{32}) & 0 \\ -x(-T_{11}+yT_{31})+y(-T_{01}+xT_{31}) & x(-T_{10}+yT_{30})-y(\end{bmatrix} \\
&= \frac{dL}{dd}\frac{dd}{dp}\begin{bmatrix} 0 & -xl_3+yk_3 & xl_2-yk_2 \\ xl_3-yk_3 & 0 & -xl_1+yk_1 \\ -xl_2+yk_2 & xl_1-yk_1 & 0 \end{bmatrix} \\
&= \frac{dL}{dd}\begin{bmatrix} dp_2(xl_3-yk_3)+dp_3(-xl_2+yk_2) \\ dp_1(-xl_3+yk_3)+dp_3(xl_1-yk_1) \\ dp_1(xl_2-yk_2)+dp_2(-xl_1+yk_1) \end{bmatrix}
\end{aligned}$$

compute2DGSBBox.backward

输入`dL_KWH_t`（3×4）

由T到R,S,p

前向为： $R, S, W, K \Rightarrow KWH\_t$

$$K=\begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^T$$

$$H = SR$$

$$Y = HW_R^T$$

$$p_v = p_w^T W_R^T + W_T^T$$

$$M=\begin{bmatrix} Y_{0,:} & 0 \\ Y_{1,:} & 0 \\ p_v^T & 1 \end{bmatrix}$$

$$T = MK$$

展开可得：

$$Y=\begin{bmatrix} H_{00}W_{R00}+H_{01}W_{R10}+H_{02}W_{R20} & H_{00}W_{R01}+H_{01}W_{R11}+H_{02}W_{R21} \\ H_{10}W_{R00}+H_{11}W_{R10}+H_{12}W_{R20} & H_{10}W_{R01}+H_{11}W_{R11}+H_{12}W_{R21} \\ H_{20}W_{R00}+H_{21}W_{R10}+H_{22}W_{R20} & H_{20}W_{R01}+H_{21}W_{R11}+H_{22}W_{R21} \end{bmatrix}$$

$$T=\begin{bmatrix} f_xM_{00}+c_xM_{02} & f_yM_{01}+c_yM_{02} & M_{02} \\ f_xM_{10}+c_xM_{12} & f_yM_{11}+c_yM_{12} & M_{12} \\ f_xM_{20}+c_xM_{22} & f_yM_{21}+c_yM_{22} & M_{22} \end{bmatrix}$$

则：

$$\frac{dL}{dM}=\begin{bmatrix} f_xL_{KWH_t}^{00} & f_yL_{KWH_t}^{01} & c_xL_{KWH_t}^{00}+c_yL_{KWH_t}^{01}+L_{KWH_t}^{02}+L_{KWH_t}^{03} & 0 \\ f_xL_{KWH_t}^{10} & f_yL_{KWH_t}^{11} & c_xL_{KWH_t}^{10}+c_yL_{KWH_t}^{11}+L_{KWH_t}^{12}+L_{KWH_t}^{13} & 0 \\ f_xL_{KWH_t}^{20} & f_yL_{KWH_t}^{21} & c_xL_{KWH_t}^{20}+c_yL_{KWH_t}^{21}+L_{KWH_t}^{22}+L_{KWH_t}^{23} & 0 \end{bmatrix}$$

由M到Y和p_v：

$$\frac{dL}{dp_w}=\begin{bmatrix} L_M^{20} & L_M^{21} & L_M^{22} \end{bmatrix}^T$$

$$\frac{dL}{dY}=\begin{bmatrix} f_xL_{KWH_t}^{00} & f_yL_{KWH_t}^{01} & c_xL_{KWH_t}^{00}+c_yL_{KWH_t}^{01}+L_{KWH_t}^{02}+L_{KWH_t}^{03} \\ f_xL_{KWH_t}^{10} & f_yL_{KWH_t}^{11} & c_xL_{KWH_t}^{10}+c_yL_{KWH_t}^{11}+L_{KWH_t}^{12}+L_{KWH_t}^{13} \\ 0 & 0 & 0 \end{bmatrix}$$

由Y到H（注意这里W矩阵为W2C.T）

$$\frac{dL}{dH}=\begin{bmatrix} L_Y^{00}W_{R00}+L_Y^{01}W_{R01}+L_Y^{02}W_{R02} & L_Y^{00}W_{R10}+L_Y^{01}W_{R11}+L_Y^{02}W_{R12} \\ L_Y^{10}W_{R00}+L_Y^{11}W_{R01}+L_Y^{12}W_{R02} & L_Y^{10}W_{R10}+L_Y^{11}W_{R11}+L_Y^{12}W_{R12} \\ 0 & 0 \end{bmatrix}$$

由H到R和S

注意这里R每一行是一个向量/轴

$$H=SR \quad R=\begin{bmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{bmatrix} \quad S=\begin{bmatrix} s_0 & 0 & 0 \\ 0 & s_1 & 0 \\ 0 & 0 & s_2 \end{bmatrix}$$

$$H=\begin{bmatrix} s_0R_{00} & s_0R_{01} & s_0R_{02} \\ s_1R_{10} & s_1R_{11} & s_1R_{12} \\ s_2R_{20} & s_2R_{21} & s_2R_{22} \end{bmatrix}$$

则

$$\frac{dL}{dS}=\begin{bmatrix} L_H^{00}R_{00}+L_H^{01}R_{01}+L_H^{02}R_{02} \\ L_H^{10}R_{10}+L_H^{11}R_{11}+L_H^{12}R_{12} \\ 0 \end{bmatrix}$$

四元数到旋转矩阵如下（每一行是一个向量，第三行是没用的因为dL/dH第三行为0）：

$$q=[r,x,y,z]$$

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

从而：

$$\frac{dL}{dq}=\begin{bmatrix} 2z(L_R^{01}-L_R^{10})-2yL_R^{02}+2xL_R^{12} \\ 2y(L_R^{01}+L_R^{10})+2zL_R^{02}-4xL_R^{11}+2rL_R^{12} \\ -4yL_R^{00}+2x(L_R^{01}+L_R^{10})-2rL_R^{02}+2zL_R^{12} \\ -4z(L_R^{00}+L_R^{11})+2r(L_R^{01}-L_R^{10})+2xL_R^{02}+2yL_R^{12} \end{bmatrix}$$

Depth Distortion

$$
\begin{aligned}
\mathcal{L}_{dist} &= \sum_{i=0}^{N-1}\sum_{j=0}^{i-1}\omega_i\omega_j(m_i-m_j)^2 \\
&= \sum_{i=0}^{N-1}\omega_i(m_i^2\sum_{j=0}^{i-1}\omega_j+\sum_{j=0}^{i-1}\omega_j m_j^2-2m_i\sum_{j=0}^{i-1}\omega_j m_j) \\
&= \sum_{i=0}^{N-1}\omega_i(m_i^2A_{i-1}+D_{i-1}^2-2m_iD_{i-1})
\end{aligned}
$$

其中 $A_i=\sum_{j=0}^{i}\omega_j, D_i=\sum_{j=0}^{i}\omega_j m_j, D_i^2=\sum_{j=0}^{i}\omega_j m_j^2$

则：

$$
\begin{aligned}
\frac{dL}{d\omega_i} &= \sum_{j=0}^{i-1}\omega_j(m_i-m_j)^2+\sum_{j=i+1}^{N-2}\omega_j(m_j-m_i)^2 \\
&= \sum_{j=0}^{N-1}\omega_j(m_j-m_i)^2 \\
&= m_i^2\sum_{j=0}^{N-1}\omega_j-2m_i\sum_{j=0}^{N-1}\omega_j m_j+\sum_{j=0}^{N-1}\omega_j m_j^2 \\
&= m_i^2A_{N-1}-2m_iD_{N-1}+D_{N-1}^2
\end{aligned}
$$

注意 $A_{N-1}, D_{N-1}, D_{N-1}^2$是标量，在代码中可以用一个 $W\times H\times 3$的矩阵同时存下，也即代码中的`ADD_2`

同理，对于 m_i：

$$
\begin{aligned}
\mathcal{L}_{dist} &= \sum_{i=0}^{N-1}\sum_{j=0}^{i-1}\omega_i\omega_j(m_i-m_j)^2 \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{i-1}\omega_i\omega_j m_i^2+\omega_i\omega_j m_j^2-2\omega_i\omega_j m_i m_j \\
&= \sum_{i=0}^{N-1}m_i^2\sum_{j=0}^{i-1}\omega_i\omega_j+\sum_{i=0}^{N-1}\sum_{j=0}^{i-1}\omega_i\omega_j m_j^2-2\sum_{i=0}^{N-1}m_i\sum_{j=0}^{i-1}\omega_i\omega_j m_j
\end{aligned}
$$

则：

$$
\begin{aligned}
\frac{dL}{dm_i} &= 2m_i\sum_{j=0}^{i-1}\omega_i\omega_j + 2\sum_{j=i+1}^{N-1}\omega_j\omega_i m_i - 2\sum_{j=0}^{i-1}\omega_i\omega_j m_j - 2\sum_{j=i+1}^{N-1}m_j\omega_j\omega_i \\
&= 2m_i\omega_i\sum_{j=0}^{i-1}\omega_j + 2m_i\omega_i\sum_{j=i+1}^{N-1}\omega_j - 2\omega_i\sum_{j=0}^{i-1}\omega_j m_j - 2\omega_i\sum_{j=i+1}^{N-1}m_j\omega_j \\
&= 2\omega_i\sum_{j=0}^{N-1}(m_i - m_j)\omega_j \\
&= 2\omega_i m_i A_{N-1} - 2\omega_i D_{N-1}
\end{aligned}
$$

由于权重 $\omega_i = \alpha_i T_i = \alpha_i\prod_{j=0}^{i-1}(1-\alpha_j)$,

$$
\frac{d\omega_i}{d\alpha_j} = \begin{cases} T_j, & j = i \\ -\dfrac{T_i\alpha_i}{1-\alpha_j}, & j < i \\ 0, & j > i \end{cases}
$$

从而

$$
\begin{aligned}
\frac{d\mathcal{L}_{dist}}{d\alpha_i} &= \sum_{j=i+1}^{N-1}\frac{d\mathcal{L}_{dist}}{d\omega_j}\frac{d\omega_j}{d\alpha_i} + \frac{d\mathcal{L}_{dist}}{d\omega_i}\frac{d\omega_i}{d\alpha_i} \\
&= \sum_{j=i+1}^{N-1}\frac{d\mathcal{L}_{dist}}{d\omega_j}(-\frac{T_j\alpha_j}{1-\alpha_i}) + \frac{d\mathcal{L}_{dist}}{d\omega_i}\frac{T_i}{1-\alpha_i}(1-\alpha_i) \\
&= \sum_{j=i}^{N-1}\frac{d\mathcal{L}_{dist}}{d\omega_j}(-\frac{T_j\alpha_j}{1-\alpha_i}) + \frac{d\mathcal{L}_{dist}}{d\omega_i}\frac{T_i}{1-\alpha_i} \\
&= \frac{L_{\omega_i}T_i - \sum_{j=i}^{N-1}L_{\omega_j}T_j\alpha_j}{1-\alpha_i}
\end{aligned}
$$

此处可根据递推式`accum_wTa`$= \sum_{j=i}^{N-1}L_{\omega_j}T_j\alpha_j$加入到`renderCUDA`的递推中

Normal Consistency

这一部分还未实现