The Forward Process of 3D Gaussian Splatting

This document describes the **forward process** of 3D Gaussian Splatting, which is the process of rendering trained 3D Gaussians onto a 2D scene.

First, the required input data is defined as parameters and settings.

Parameters

The parameters are the data that can be obtained from the training process:

- q_i: Rotation 3D Gaussian (quaternion)
- s_i: Scale of 3D Gaussian (3D vector)
- h_i: Spherical harmonics parameters of 3D Gaussian
- α_i: Opacity of 3D Gaussian
- ${p_w}_i$: The location of 3D Gaussian in the world frame

where i is the index of 3D Gaussian. For ease of reading, i is omitted in the following text.

Settings

The settings are values related to the configuration of a virtual camera, which do not need to be trained:

- R_{cw}: The rotation of the camera (local frame)
- t_{cw}: The translation of the camera (local frame)
- K: Intrinsic Parameters (i.e., f_x, f_y, c_x, c_y)

Pipeline

The overall pipeline for rendering a 2D scene from a set of 3D Gaussians is as follows:

1. Calculate the 2D coordinates when projecting the mean of the 3D Gaussian onto the 2D scene.
2. Compute the 3D covariance of the 3D Gaussian.
3. Calculate the 2D covariance when projecting the 3D covariance onto the 2D scene.

4. Calculate the color of the 3D Gaussian based on spherical harmonics.
5. Render the 2D scene using the information calculated in steps 1-4.

1. Project the Mean of 3D Gaussian

We use the camera's rotation (R_{cw}) and translation (t_{cw}) to transform the mean (p_w) of the 3D Gaussian to a point in the camera coordinate system (p_c). This process is shown in equation (1.1).

$$\begin{aligned} p_c &= \mathrm{T}_{cw}(p_w) \\ &= R_{cw}p_w + t_{cw} \end{aligned} \tag{1.1}$$

Then, we calculate the pixel (u) coordinates when projecting the point (p_c) onto the 2D scene.

$$u(p_c) = \begin{bmatrix} xf_x/z + c_x \\ yf_y/z + c_y \end{bmatrix} \tag{1.2}$$

Here, each component of (p_c) is represented as x, y, and z, respectively.

2. Calculate the 3D Gaussian from Rotation and Color

The 3D covariance Σ of the 3D Gaussian is expressed not directly as a matrix but as a composition of rotation q and scaling s. The composition calculation is shown in equation (2).

$$\begin{aligned} \Sigma(q,s) &= RSS^TR^T \\ \sigma(q,s) &= \mathrm{upper\_triangular}(\Sigma) \end{aligned} \tag{2}$$

Here, R is the matrix representation of the quaternion q. S is a diagonal matrix formed from the vector s.

$$R = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_zq_w) & 2(q_xq_z+q_yq_w) \\ 2(q_xq_y+q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_xq_w) \\ 2(q_xq_z-q_yq_w) & 2(q_yq_z+q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix} \tag{2.1}$$

$$S = \begin{bmatrix} s_0 & 0 & 0 \\ 0 & s_1 & 0 \\ 0 & 0 & s_2 \end{bmatrix} \tag{2.2}$$

3. Project the 3D Covariance to 2D Image as a 2D Covariance

When a 3D Gaussian is projected onto a 2D scene, it can be represented as a 2D Gaussian distribution. Equation (3) shows the formula for calculating the covariance matrix of this 2D Gaussian distribution.

$$\begin{aligned} \Sigma'(\sigma, p_c) &= J R_{cw} \Sigma R_{cw}^T J^T \\ \sigma'(\sigma, p_c) &= \text{upper\_triangular}(\Sigma') \end{aligned} \tag{3}$$

Proof of Equation (3)

Let p_w be a set of 3D points in world coordinates, and let m_w be the mean of p_w. According to the definition of the covariance matrix, the covariance matrix of p_w is calculated as follows:

$$\Sigma = \mathrm{E}[(p_w - m_w)(p_w - m_w)^T] \tag{3.1}$$

The covariance matrix of points in camera coordinates can be calculated using equation (3.2):

$$\begin{aligned} \Sigma_c &= \mathrm{E}[(p_c - m_c)(p_c - m_c)^T] \\ &= \mathrm{E}[(\mathrm{T}_{cw}(p_w) - \mathrm{T}_{cw}(m_c))(\mathrm{T}_{cw}(p_w) - \mathrm{T}_{cw}(m_w))^T] \\ &= R_{cw} \mathrm{E}[(p_w - m_w)(p_w - m_w)^T] R_{cw}^T \\ &= R_{cw} \Sigma R_{cw}^T \end{aligned} \tag{3.2}$$

The covariance matrix of points in image coordinates can be calculated using equation (3.3):

$$\Sigma' = \mathrm{E}[(\mathrm{u}(p_c) - \mathrm{u}(m_c))(\mathrm{u}(p_c) - \mathrm{u}(m_c))^T] \tag{3.3}$$

Here, m_c is the mean of p_c, and $p_c - m_c$ is a small value, which is defined as δ. Although u (Equation 1.2) is a nonlinear function, there exists an approximate calculation using the Jacobian matrix J of u as follows:

$$\begin{aligned} \mathrm{u}(m_c + \delta) &= \mathrm{u}(m_c) + J\delta \\ \mathrm{u}(p_c) - \mathrm{u}(m_c) &= J(p_c - m_c) \end{aligned} \tag{3.4}$$

Substituting equations (3.4) and (3.2) into equation (3.3), the 2D covariance can be calculated as follows:

$$\begin{aligned}\Sigma' &= \mathrm{E}[J(p_c - m_c)(p_c - m_c)^T J^T] \\ &= J\Sigma_c J^T \\ &= JR_{cw}\Sigma R_{cw}^T J^T\end{aligned} \tag{3.5}$$

The Jacobian of u is given by:

$$J = \begin{bmatrix} \frac{f_x}{x} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{bmatrix} \tag{3.6}$$

4. Calculate the Color from Spherical Harmonics

In 3D Gaussian Splatting, spherical harmonics are used to approximate complex light absorption, refraction, or reflection, which exhibit color variations when viewed from different directions.

The color calculation using spherical harmonics is given by the following equation:

$$\mathrm{c}(r, h) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} h_{lm} \mathrm{Y}_l^m(r) \tag{4}$$

Here, r is the unit direction vector between the camera and the 3D Gaussian, and Y_l^m represents the l-th dimension m-th base function.

5. Calculate the Color for Each Pixel

Using the information obtained from steps 1 to 4, calculate the final values (RGB colors) for all pixels in the 2D scene.

$$\gamma_j = \sum_{i \in N} \alpha'_{ij}(x_j, \alpha_i, \Sigma') c_i \tau_{ij} \tag{5}$$

Where α'_{ij} represents the opacity of the i-th 3D Gaussian at pixel j, and is calculated using the following equation:

$$\alpha'_{ij} = \exp\left(-0.5(u_i - x_j)\Sigma'^{-1}(u_i - x_j)^T\right)\alpha_i \tag{5.1}$$

The opacity increases as the Mahalanobis distance between the mean of the 2D Gaussian (u_i) and the current pixel (x_j) gets closer.

As light from objects farther away from the camera travels through multiple objects and ends up in the camera, the light becomes weaker. As a result, the contribution to the final color decreases as the distance from the camera increases.

The τ_{ij} is a coefficient that accounts for the attenuation that occurs when rendering the i-th Gaussian at pixel j as the light passes through multiple 3D Gaussians in front of it. In other words, it represents the maximum amount of color that can be reflected on this pixel. If the value is 1, there is no attenuation and the color is shown as is. Conversely, if the value is 0, there is complete attenuation and it is totally not visible.

$$\begin{aligned} \tau_{ij} &= \prod_{k=1}^{i-1}(1-\alpha'_{kj}) \\ \tau_{1j} &= 1 \end{aligned} \tag{5.2}$$

Here, the inverse of the 2D covariance can be calculated as follows:

$$\Sigma'^{-1} = \frac{1}{ac-b^2}\begin{bmatrix} c & -b \\ -b & a \end{bmatrix} \tag{5.3}$$

where a, b, and c are the upper triangular elements of the 2D covariance.