Supplementary Document
EnliveningGS: Active Locomotion of 3DGS

In this document, we derive the relevant formulas mentioned in the main paper and provide additional details on the implementation and experimental results.

We also present the results of locomotion in the supplementary video.

1. Implementation Details

1.1. Splitting Gaussians at Center

Problem Definition Given a Gaussian kernel $(\alpha_0, \mu_0, \Sigma_0)$, define the problem of splitting it at the center such that it is divided into two new Gaussian kernels $(\alpha_l, \mu_l, \Sigma_l)$ and $(\alpha_r, \mu_r, \Sigma_r)$. The goal is to ensure that each of the resulting parts remains as consistent as possible with its own part at the original Gaussian kernel before splitting.

Assumptions Since a truncated Gaussian kernel cannot be fully represented by a new complete Gaussian kernel, and there are multiple ways to truncate it, we make the following assumptions to simplify the problem for accurate description and approximation:

- The split occurs at the center of the Gaussian, with the truncation plane passing through the mean of the Gaussian kernel and its normal vector aligned with the direction of the eigenvector with largest eigenvalue of the covariance matrix.
- The spherical harmonic (SH) coefficients of the two parts after the split remain consistent with those of the original Gaussian kernel.
- The splitting kernels are detected under deformation but the splitting process occurs in the rest shape of $\mathcal{G}_o$. After a Gaussian is split, the new Gaussian kernels need to establish their own bidirectional local embeddings and barycentric coordinates to support deformation.
- For approximation, we consider the statistical information of the Gaussian distribution i.e. zero, first, and second-order moments of the truncated Gaussian and new kernels:

$$\begin{aligned}
\int_{\mathbb{V}_k} \alpha_0 \mathsf{PDF}_0(\mathbf{v}) dV &= \int_{\mathbb{R}^3} \alpha_k \mathsf{PDF}_k(\mathbf{v}) dV, k \in \{l, r\}, \\
\int_{\mathbb{V}_k} \alpha_0 \mathbf{v} \mathsf{PDF}_0(\mathbf{v}) dV &= \int_{\mathbb{R}^3} \alpha_k \mathbf{v} \mathsf{PDF}_k(\mathbf{v}) dV, k \in \{l, r\}, \\
\int_{\mathbb{V}_k} \alpha_0 \mathbf{v}\mathbf{v}^T \mathsf{PDF}_0(\mathbf{v}) dV &= \int_{\mathbb{R}^3} \alpha_k \mathbf{v}\mathbf{v}^T \mathsf{PDF}_k(\mathbf{v}) dV, k \in \{l, r\},
\end{aligned} \tag{1}$$

where $\mathbb{V}_k$ represents the truncated half of the Gaussian kernel.

Zero-order moment Zero-order moment of a Gaussian kernel describes the total mass or the integral of the probability density function over the entire space.

Since the 3D Gaussian kernel exhibits spherical symmetry, any plane that passes through the center of the Gaussian kernel will divide the kernel into two regions of equal volume and mass. We have:

$$\int_{\mathbb{V}_k} \alpha_0 \mathsf{PDF}_0(\mathbf{v}) dV = 0.5\alpha_0 \tag{2}$$

and

$$\begin{aligned}
\int_{\mathbb{R}^3} \alpha_k \mathsf{PDF}_k(\mathbf{v}) dV &= \alpha_k \int_{\mathbb{R}^3} \mathsf{PDF}_k(\mathbf{v}) dV \\
&= \alpha_k = 0.5\alpha_0, \quad k \in \{l, r\}
\end{aligned} \tag{3}$$

First-order moment The first-order order moment is the mean vector of the distribution, which is the center of the expected value of the distribution in three-dimensional space.

The covariance matrix $\mathbf{\Sigma}_0$ can be decomposed using eigenvalue decomposition:

$$\mathbf{\Sigma}_0 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \tag{4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i, and $\mathbf{V}$ is the matrix of eigenvectors, with each column $\mathbf{v}_i$ being an eigenvector.

The direction corresponding to the largest eigenvalue λ_{max} is given by the eigenvector $\mathbf{v}_{max}$.

After splitting, the left half of the Gaussian distribution can be considered a truncated Gaussian distribution. The mean of a truncated Gaussian distribution changes. The new mean for the left half can be calculated using the following formula:

$$\mu_l = \mu_0 - \mathbf{\Sigma}_0 \mathbf{v}_{max} \cdot \frac{\mathsf{PDF}_s(\beta)}{1 - \Phi(\beta)}. \tag{5}$$

where:

- $\mathsf{PDF}_s(\beta)$ is the PDF of the standard normal distribution.
- $\Phi_s(\beta)$ is the cumulative distribution function (CDF) of the standard normal distribution.
- $\beta = 0$, because we are splitting at the mean.

Given $\beta = 0$, we have $\phi(0) = \frac{1}{\sqrt{2\pi}}$ and $\Phi(0) = 0.5$. Thus:

$$\begin{aligned} \mu_l &= \mu_0 - \mathbf{\Sigma}_0 \mathbf{v}_{max} \cdot \frac{\frac{1}{\sqrt{2\pi}}}{1 - 0.5} \\ &= \mu_0 - \mathbf{\Sigma}_0 \mathbf{v}_{max} \cdot \frac{2}{\sqrt{2\pi}} \end{aligned} \tag{6}$$

Since $\mathbf{\Sigma}_0 \mathbf{v}_{max} = \lambda_{max} \mathbf{v}_{max}$, we get:

$$\mu_l = \mu_0 - \lambda_{max} \mathbf{v}_{max} \cdot \frac{2}{\sqrt{2\pi}} \tag{7}$$

In 3DGS, the covariance matrix is decomposed as:

$$\mathbf{\Sigma}_0 = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T, \tag{8}$$

where $\mathbf{S}\mathbf{S}^T$ forms the diagonal matrix $\mathbf{\Lambda}$, and each column vector of $\mathbf{R} = \mathbf{V}$ (orders may differ) represents a principal component's direction vector. Since the end points on the major axis $\mathbf{v}_k$ in the left-direction local embedding is defined as $\mathbf{v}_k = \mu_0 - \gamma s_k \mathbf{r}_k$, we have

$$\lambda_{max} \mathbf{v}_{max} = \frac{\mathbf{v}_k - \mu_0}{\gamma}. \tag{9}$$

To simplify, we can denote $\frac{2\gamma}{\sqrt{2\pi}}$ as a constant κ, so:

$$\mu_l = \mu_0 - \frac{\mathbf{v}_k - \mu_0}{\kappa} \tag{10}$$

By analogy, it can be deduced that for the right part:

$$\mu_r = \mu_0 + \frac{\mathbf{v}_k - \mu_0}{\kappa} \tag{11}$$

Second-order moment The second-order moment indicates the spread and correlation of the distribution through the covariance matrix. It is given by:

$$\mathbb{E}[(X - \mu)(X - \mu)^T] = \mathbf{\Sigma} \tag{12}$$

Here, X is a random vector representing the 3D Gaussian distribution. This equation can be expressed in another form as follows:

$$\int_{\mathbb{R}^3} \mathbf{v}\mathbf{v}^T \mathsf{PDF}(\mathbf{v}) dV = \mathbf{\Sigma} + \mu\mu^T \tag{13}$$

For the original Gaussian kernel:

$$\begin{aligned} &\int_{\mathbb{R}^3} \alpha_0 \mathbf{v}\mathbf{v}^T \mathsf{PDF}_0(\mathbf{v}) dV \\ &= \int_{\mathbb{V}_l} \alpha_0 \mathbf{v}\mathbf{v}^T \mathsf{PDF}_0(\mathbf{v}) dV + \int_{\mathbb{V}_r} \alpha_0 \mathbf{v}\mathbf{v}^T \mathsf{PDF}_0(\mathbf{v}) dV \\ &= \int_{\mathbb{R}^3} \alpha_l \mathbf{v}\mathbf{v}^T \mathsf{PDF}_l(\mathbf{v}) dV + \int_{\mathbb{R}^3} \alpha_r \mathbf{v}\mathbf{v}^T \mathsf{PDF}_r(\mathbf{v}) dV \end{aligned} \tag{14}$$

We have

$$\alpha_0(\mathbf{\Sigma}_0 + \mu_0\mu_0^T) = \alpha_l(\mathbf{\Sigma}_l + \mu_l\mu_l^T) + \alpha_r(\mathbf{\Sigma}_r + \mu_r\mu_r^T) \tag{15}$$

Because of the spherical symmetry of the Gaussian kernel, we also have

$$\begin{aligned} \int_{\mathbb{R}^3} \alpha_l \mathbf{v}\mathbf{v}^T \mathsf{PDF}_k(\mathbf{v}) dV &= \int_{\mathbb{R}^3} \alpha_l \mathbf{v}\mathbf{v}^T \mathsf{PDF}_k(\mathbf{v}) dV \\ \alpha_l(\mathbf{\Sigma}_l + \mu_l\mu_l^T) &= \alpha_r(\mathbf{\Sigma}_r + \mu_r\mu_r^T) \end{aligned} \tag{16}$$

Combining equations (3), (10), (11), (15) and (16), we can derive:

$$\mathbf{\Sigma}_l = \mathbf{\Sigma}_r = \mathbf{\Sigma}_0 - \frac{(\mathbf{v}_k - \mu_0)(\mathbf{v}_k - \mu_0)^T}{\kappa^2} \tag{17}$$

Conclusion For a Gaussian kernel $(\alpha_0, \mu_0, \Sigma_0)$, splitting it at the center into two new Gaussian kernels $(\alpha_l, \mu_l, \Sigma_l)$ and $(\alpha_r, \mu_r, \Sigma_r)$ can be expressed as follows:

$$\begin{aligned} &\alpha_l = \alpha_r = \frac{\alpha_0}{2}, \\ &\mu_l = \mu_0 - \frac{\mathbf{v}_k - \mu_0}{\kappa}, \quad \mu_r = \mu_0 + \frac{\mathbf{v}_k - \mu_0}{\kappa}, \\ &\mathbf{\Sigma}_l = \mathbf{\Sigma}_r = \mathbf{\Sigma}_0 - \frac{(\mathbf{v}_k - \mu_0)(\mathbf{v}_k - \mu_0)^T}{\kappa^2}. \end{aligned} \tag{18}$$

Here $\kappa = \frac{2\gamma}{\sqrt{2\pi}}$ is a constant.

1.2. Muscle Force

In this section, we discuss how to derive the muscle force acting on the object from muscle activation.

We model muscle fibers as polygonal curves with M segments, embedded in a tetrahedral mesh containing N points. Each muscle segment is represented as an individual spring that can either contract or extend along its current direction without bending. The segment applies an activation force $a \in \mathbb{R}$ along its direction, facilitating either contraction or extension. An activation affects multiple nearby elements. For the i-th element and j-th muscle segment, the

	Cap.	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{G}_o\rvert$	$\lvert\mathcal{G}_c\rvert$	Ele.	N	M	Solve (ms)	Deform (ms)
Jumping chess	112	1424043	3350	1729	2376	693	208	921.7	8.12
Peashooter vs. Zombie	192	547833	8912	1358	4280	1375	62	745.6	10.81
Walking stool	114	361487	16003	492	2582	888	224	492.3	13.28

Table 1. **Summary of key metrics for our locomotion experiments.** The columns represent the number of training views (Cap.), the total number of Gaussians in the scene ($|\mathcal{G}|$), the number of Gaussians associated with objects ($|\mathcal{G}_o|$), the number of Gaussians involved in contact ($|\mathcal{G}_c|$), the number of elements in the tetrahedral mesh (Ele.), the number of nodes in the tetrahedral mesh (N), the number of muscle segments (M), the average time to solve muscle activation per frame (Solve (ms)), and the average time for deformation per frame using bidirectional local embeddings (Deform (ms)).

influencing weight w_{ij} is based on their geodesic distance g_{ij} on the mesh as a Gaussian kernel:

$$w_{ij} = \mathsf{exp}\left(-\frac{g_{ij}^2}{c^2}\right), \tag{19}$$

where c is the variance of the Gaussian function. w_{ij} only depends on the rest shape of the body and muscle, and it can be pre-computed. The accumulated muscle stress in the deformed coordinates of the element is:

$$\sigma_i = \sum_j w_{ij}\mathbf{R}_j\mathbf{E}_j\mathbf{R}_j^\top, \tag{20}$$

where

$$\sigma_j = \mathbf{U}\begin{bmatrix} f_j & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{U}^\top = \mathbf{d}_j \otimes \mathbf{d}_j a_j. \tag{21}$$

In this context, $\mathbf{R}_j$ transforms the stress tensor σ_j from the reference coordinates to the deformed coordinates, while $\mathbf{U}$ rotates the vector $[1,0,0]^\top$ in the reference coordinates to align with $\mathbf{d}_j$, which represents the direction of the muscle segments in the reference coordinates. We project σ_i onto the area-weighted face normal of the element in the deformed coordinates. This surface force is then evenly distributed to the vertices to determine $\mathbf{f}_m$ at each node. We employ a pose-dependent activation matrix $\mathbf{A} \in \mathbb{R}^{3N\times M}$ to encode the muscle force computation, such that:

$$\mathbf{f}_m = \mathbf{A}(\mathbf{p})\mathbf{a}\,, \tag{22}$$

where $\mathbf{a} \in \mathbb{R}^M$ is the vector of activations of all the muscle segments.

1.3. Inpainting

Our inpainting pipeline builds upon the InFusion framework while incorporating planar contact environment as a known prior. To ensure comprehensive coverage of our methodology, we provide a brief description of the inpainting pipeline in this section.

Given environmental incomplete Gaussians $\mathcal{G}_{\text{scene}}$ with holes in the occluded area and contact plane $P(\mathbf{N}, d)$, the inpainting algorithm will produce the completed environment $\mathcal{G}^\star_{\text{scene}}$.

First, based on a single reference view image I_r and camera pose Π_r, the user draws a mask m_r for the missing region. A 2D inpainting tool is applied to inpaint the color image:

$$I_r^\star \leftarrow F_{\text{Inpaint2D}}(I_r, m_r).$$

Using the plane parameters $P(\mathbf{N}, d)$, the inpainted image patch $I^\star_{\text{patch}} = I_r^\star \odot m_r$, and camera parameters Π_r, we unproject $I^\star_{\text{patch}}$ from image space to 3D coordinates to form a colored point cloud $\mathcal{P}^\star$. The merged environment $\mathcal{G}'_{\text{scene}} \leftarrow \mathcal{G}_{\text{scene}} \cup \mathcal{P}^\star$ is fine-tuned for approximately 100 iterations to produce the final Gaussian model $\mathcal{G}^\star_{\text{scene}}$, guided by a view-specific loss:

$$\mathcal{L}_{\text{fine-tune}} = (1-\lambda)\|I' - I_r^\star\|_1 + \lambda \cdot \text{D-SSIM}(I', I_r^\star),$$

where I' denotes the image rendered from the selected viewpoint and $\lambda = 0.2$ is held constant for all experiments.

2. Detailed Results

In the main paper, we demonstrate the actions of jumping, twisting, and walking through examples from three different scenarios. All examples were initially captured in real world using a digital camera with a 28mm focal length, reconstructed into sparse point clouds using COLMAP to estimate camera poses, and subsequently reconstructed using 3D Gaussian Splatting. In Table 1, we report the key metrics and time performance.