FruitNinja: 3D Object Interior Texture Generation with Gaussian Splatting

Supplementary Material

1. Cross-Section Specifications for Training

In Section 3.2 (Fig. 3), we introduced the use of user-defined cross-sections for generating internal textures. Here, we present the detailed cross-sectional angles used for training the six objects in our experiments. We used two cross-section types for objects in Fig. 1 due to their more complex internal structures, whereas the objects in Fig. 2 used only a single type. Additional visual results for objects after arbitrary geometric transformations demonstrate texture coherence, even when cuts are misaligned with the trained angles.

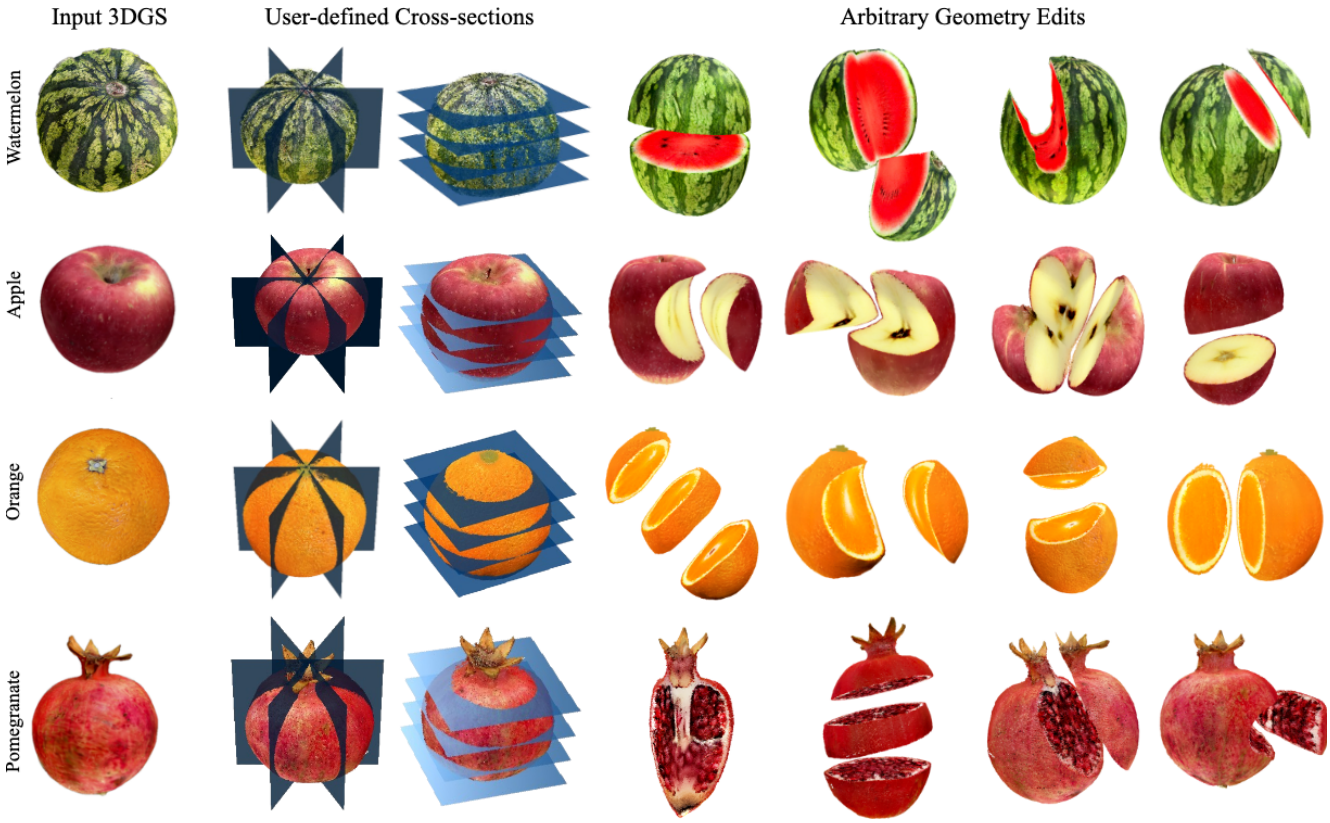

Figure 1. For the four objects shown above (watermelon, apple, orange, and pomegranate), we use two types of input cross-sections: (1) **Vertical Cross-sections**: 30 slices evenly spaced in angle, spanning a full rotation around the central axis of the object (as illustrated in the second column). (2) **Horizontal Cross-sections**: We use 40 horizontal slices (as shown in the third column), evenly spaced along the vertical axis to cover the entire object.

Figure 2. For bread and red velvet cake, we use only vertical cross-sections: (1) **Bread**: 60 evenly spaced vertical cross-sections (second column) covering the object. (2) **Red Velvet Cake**: 30 vertical cross-sections, evenly distributed and radially arranged around the central axis, spanning a full rotation of the object.

2. Additional Evaluations

2.1. Cross-sections Consistency

We provide additional qualitative evaluations in Figure 3, where the object is sequentially sliced along a fixed direction to assess cross-sectional texture consistency.

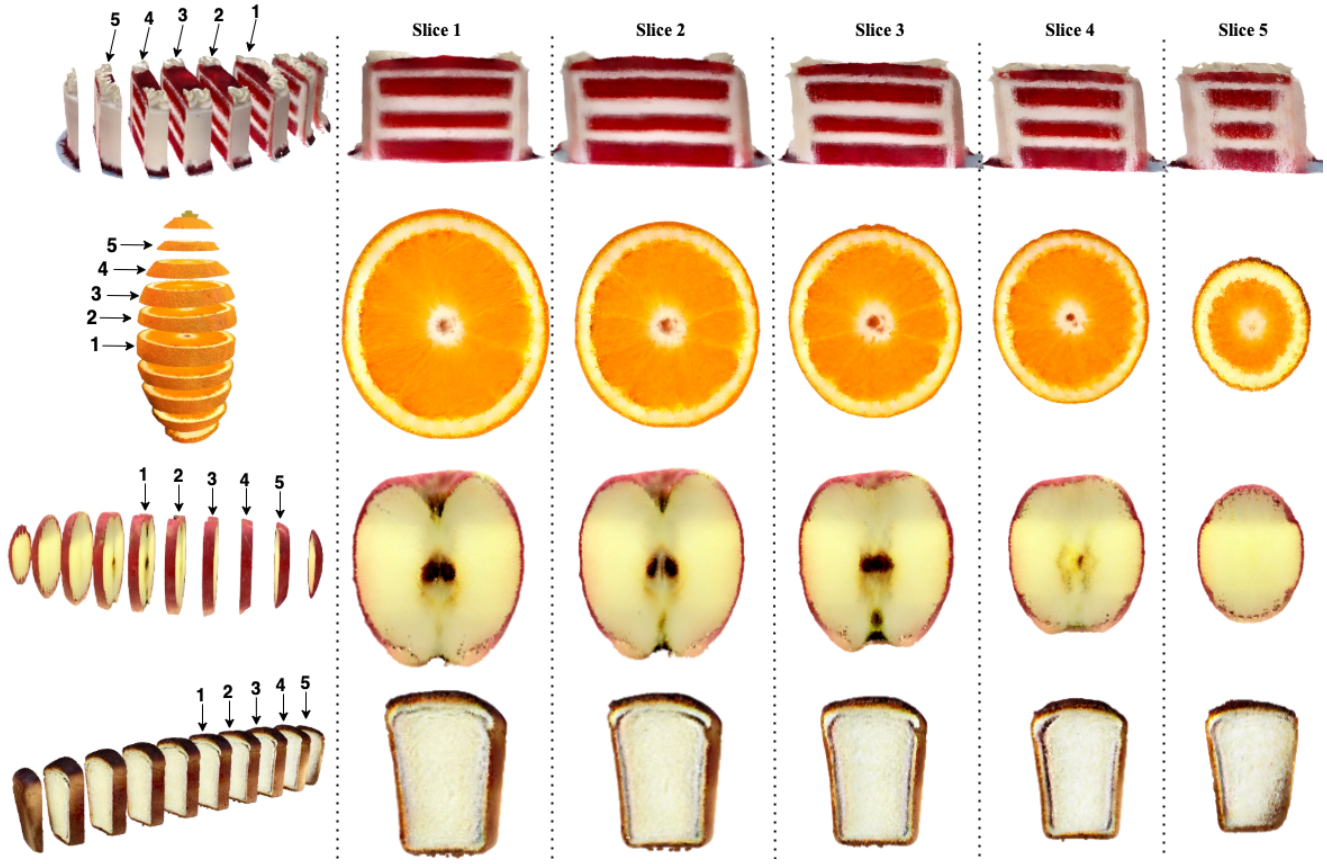

Figure 3. As shown in the first column, objects (cake, orange, apple, and bread) are sequentially sliced along a fixed direction.

2.2. Texture Quality

To evaluate the quality of generated internal textures in the absence of ground truth, we conducted additional evaluations by using the visual understanding capabilities of Multimodal Large Language Models (MLLMs). These models offer structured, human-aligned assessments across key perceptual dimensions, making them a convenient and scalable choice for evaluating generated 3D content. For each of the six trained objects, we sample 100 random slicing angles and render four nearby views of the sliced object, as shown in Figure 4.

Figure 4. Examples of rendered views used as input prompts for the MLLM-based evaluation. The top row shows slices from FruitNinja, and the bottom row shows slices from PhysGaussians. Each set includes multiple angles rendered from a sampled slice. These views are fed into GPT-4v to assess perceptual quality across key dimensions such as texture fidelity, geometry, visual quality etc.

We then prompt the MLLM with a Visual Question Answering (VQA) query to obtain numerical scores for realism, geometric accuracy and visual quality. Additionally, we extract a confidence score to verify whether the object is correctly identified during the evaluation. Aggregated results and evaluation details are summarized in Tables 1 and 2.

Evaluation Aspect	**Prompt Used**
Visual Fidelity	"To what extent does this internal slice resemble that of a real *object*? Rate on a scale from 1 to 5."
Geometry	"How accurately does this slice's geometry match that of a *object*? Rate on a scale from 1 to 5"
Rendered Quality	"Is this internal slice a clear and high-quality view of a *object*? Rate on a scale from 1 to 5"
Recognition	"Based on this internal slice, what object do you believe it represents, and how confident are you in this identification (0–100%)?"

Table 1. Standardized GPT-4V prompts used to evaluate internal sliced textures across objects.

Evaluation Aspect	**Method**	**Apple**	**Bread**	**Watermelon**	**Orange**	**Pomegranate**	**Cake**
Visual Fidelity (1–5)	*PhysGaussian*	3.6	3.5	4.0	3.8	3.7	3.9
	FruitNinja	**4.8**	**4.7**	**4.9**	**4.7**	**4.6**	**4.8**
Geometry (1–5)	*PhysGaussian*	3.7	3.5	3.8	3.6	3.5	3.7
	FruitNinja	**4.6**	**4.4**	**4.8**	**4.5**	**4.4**	**4.6**
Rendered Quality (1–5)	*PhysGaussian*	3.8	3.9	4.0	3.8	3.9	4.0
	FruitNinja	**4.5**	**4.6**	**4.7**	**4.5**	**4.5**	**4.7**
Recognition (0–100%)	*PhysGaussian*	85%	82%	88%	84%	80%	86%
	FruitNinja	**98%**	**96%**	**99%**	**97%**	**95%**	**98%**

Table 2. Quantitative evaluation results from MLLM for six objects using PhysGaussian and FruitNinja. Higher scores indicate better performance across four criteria.

3. Method Differences with PhysGaussian

PhysGaussian addresses internal filling of Gaussian particles by discretizing particle opacities onto a uniform voxel grid and subsequently identifying internal voxels using a ray-casting procedure. Specifically, the density at voxel index $\mathbf{i} = (i, j, k)$ is computed as follows:

$$\rho(\mathbf{i}) = \sum_p \alpha_p \cdot \frac{1}{8} \sum_{u,v,w \in \{0,1\}} \exp\left(-\frac{1}{2} (\mathbf{x}_p - \mathbf{x}_{\mathbf{i}+[u,v,w]})^\top \mathbf{\Sigma}_p^{-1} (\mathbf{x}_p - \mathbf{x}_{\mathbf{i}+[u,v,w]}) \right),$$

where $\mathbf{x}_p$ denotes particle positions, α_p opacity, $\mathbf{\Sigma}_p$ covariance matrices, and $\mathbf{x}_{\mathbf{i}+[u,v,w]} = (\mathbf{i} + [u, v, w]) \cdot d_x$ represent voxel corner coordinates with voxel size d_x. A voxel is identified as internal if it is enclosed by surface regions in all directions, confirmed by casting rays along coordinate axes and checking intersection conditions:

$$\mathbf{i} \in \mathcal{V}_{\text{internal}} \quad \text{if} \quad \prod_{\mathbf{d} \in \{\pm \mathbf{e}_x, \pm \mathbf{e}_y, \pm \mathbf{e}_z\}} f(\mathbf{i}, \mathbf{d}) = 1, \quad \text{and} \quad \left(\sum_s [\rho(\mathbf{i} + s\mathbf{d}_{\text{ray}}) \text{ crosses } \rho_{\text{ray}}] \right) \mod 2 = 1,$$

where $f(\mathbf{i}, \mathbf{d})$ denotes a binary intersection test in direction $\mathbf{d}$, and ρ_{ray} is the threshold density for intersection checking. Internal voxels are then populated with new Gaussian particles whose attributes (opacity α_{new}, covariance $\mathbf{\Sigma}_{\text{new}}$, and color coefficients $\mathbf{C}_{\text{new}}$) directly inherit from the nearest surface particle:

$$p_{\text{closest}} = \arg\min_{p \in \mathcal{P}_{\text{surf}}} \|\mathbf{x}_{\text{new}} - \mathbf{x}_p\|_2, \quad \alpha_{\text{new}} = \alpha_{p_{\text{closest}}}, \quad \mathbf{C}_{\text{new}} = \mathbf{C}_{p_{\text{closest}}}, \quad \mathbf{\Sigma}_{\text{new}} = \mathbf{\Sigma}_{p_{\text{closest}}}.$$

However, this nearest-neighbor inheritance approach inherently assumes internal textures closely resemble surface textures, often causing unrealistic and blurred internal details (see evaluation results in Tables 1 and 2). In contrast, our proposed FruitNinja method explicitly synthesizes distinct and realistic internal textures using diffusion-based SDS, guided by selected reference cross-sectional views (see Section 3 for detailed methodology). FruitNinja introduces a progressive refinement pipeline and voxel-grid-based smoothing strategies to maintain global consistency of textures and geometry during arbitrary slicing interactions. Furthermore, OpaqueAtomGS achieves stable training and detailed texture representation by enforcing atomic clipping and uniform opacification. These measures prevent oversized Gaussians from overlapping multiple cross-sectional regions and eliminate unintended blending from background contributions, resulting in consistent internal geometry and enhanced visual realism compared to PhysGaussian (see Figure 4 for visual examples).