Deformable Tetrahedral Meshes with Gaussian Splats for Fast Rendering

Harshit Gupta
University of Toronto
harshit.gupta@mail.utoronto.ca

Abstract

This paper presents an innovative approach to 3D reconstruction and real-time rendering, focusing on the challenges posed by deformable objects. We introduce a novel integration of deformable tetrahedral meshes, leveraging DefTet, with 3D Gaussian splatting for efficient and high-fidelity radiance field rendering. Our method addresses the need for realistic, interactive environments in VR/AR, enhances surgical planning through accurate medical imaging simulations, and contributes to more immersive experiences in animation and gaming. By dynamically adjusting embedded Gaussians within the deformable mesh using barycentric coordinates, we achieve accurate modeling of deformations and efficient real-time rendering. This approach not only optimizes the rendering process for complex scenes but also scales effectively for large-scale applications, offering a significant advancement in various fields requiring detailed and dynamic 3D representations.

1. Introduction

3D reconstruction and real-time rendering are pivotal in advancing the capabilities of various fields, ranging from virtual reality (VR) and augmented reality (AR) to medical imaging and computer-aided design. The problem at hand involves enhancing the fidelity and efficiency of 3D reconstruction processes, particularly when dealing with deformable objects. Achieving high-quality, real-time rendering of such objects remains a significant challenge due to the dynamic nature of deformations and the computational demands of rendering complex scenes accurately.

This work is critical for several reasons. First, in VR and AR, the realism of interactive environments is heavily dependent on the quality and responsiveness of 3D rendered objects. Enhanced methods for rendering deformable objects in real-time can significantly improve user experience in these applications. In medical imaging, accurate and real-time rendering of deformable structures, like organs, is vital for surgical planning and simulation. Similarly, in animation and gaming, improved rendering techniques can lead to more realistic and immersive experiences.

Incorporating these considerations, our work presents a learned shape representation that is adept at handling intricate deformations and allows for the efficient rendering of high-fidelity novel views. This is achieved through the integration of a novel deformable tetrahedral mesh framework with advanced rendering techniques, paving the way for significant improvements across a wide range of applications.

Our approach to addressing this challenge is a novel integration of deformable tetrahedral meshes, specifically NVIDIA Labs' DefTet, with 3D Gaussian splatting for radiance field rendering. The key idea is to place Gaussians inside a deformable mesh. As the mesh undergoes deformation, the coordinates of the embedded Gaussians are dynamically adjusted using barycentric coordinates. This allows the mesh deformations to be accurately reflected in the 3D reconstruction. Subsequently, these Gaussians are rasterized to generate the final image, ensuring that the rendering process remains efficient and capable of operating in real-time. This method offers a promising solution by leveraging the strengths of both deformable meshes for accurate modeling of objects and Gaussian splatting for efficient and high-quality rendering.

2. Related work

Tetrahedral meshes are a common representation for 3D shape, particularly useful when modelling deformation, materials, or physical simulation. This includes everything from finite element simulations in the engineering context (e.g., [1]) to physics-based animation for modern media (e.g., [7]).

In computer vision, tetrahedral representations have been applied to a variety of tasks. DefTet [2] considered the use of tetmeshes for deformable object repre sentation and novel view synthesis (NVS). Howe Followup work considered extensions to generative modelling [6], but did not focus on high-fidelity NVS. Our approach will build on DefTet, in an attempt to retain deformability while enabling fast and high-fidelity rendering.

Separately, Neural Radiance Fields (NeRFs) have shown state-of-the-art capabilities in NVS from multiview images

(e.g., see [3]), particularly recent methods based on Gaussian splatting [4]. In the NeRF context, tetrahedral meshes have been used as an adaptive representation that aids in training speed and fidelity [5], as well as applied to editing (particularly deforming) 3D scenes (e.g., [8, 9]). However, these works have not been combined with Gaussian splatting approaches, to enable rapid rendering; further, our work also explicitly uses the Gaussian splatting point cloud in the process of fitting the tetmesh to the scene.

3. Background

3.1. Deformable Tetrahedral Meshes (DefTet)

DefTet's [2] approach to deformable tetrahedral meshes involves a specific representation of vertices, faces, and tetrahedrons. The vertices of a tetrahedron are denoted as $\mathbf{v}_i = [x_i, y_i, z_i]^T$, where $i \in \{1, \ldots, N\}$ and N is the number of all vertices. We use $\mathbf{v}$ to denote all vertices. Each triangular face of the tetrahedron is denoted with its three vertices as $f_s = [\mathbf{v}_{as}, \mathbf{v}_{bs}, \mathbf{v}_{cs}]$, with $s \in \{1, \ldots, F\}$ indexing the faces and F being the total number of faces. Each tetrahedron is represented with its four vertices: $T_k = [\mathbf{v}_{ak}, \mathbf{v}_{bk}, \mathbf{v}_{ck}, \mathbf{v}_{dk}]$, with $k \in \{1, \ldots, K\}$ and K the total number of tetrahedrons.

The (binary) occupancy of a tetrahedron T_k is denoted as O_k. The occupancy depends on the positions of the vertices, which can deform. Thus, we use the notation:

$$O_k = O_k(e(T_k)), \tag{1}$$

to emphasize that O_k depends on the four vertices of the tetrahedron. DefTet simplifies the notation to $O_k(\mathbf{v}_k)$ indicating that O_k depends on the (correctly indexed) tetrahedron vertices. The probability of whether a triangular face f_s defines a surface of an object is then equal to:

$$P_s(\mathbf{v}) = O_{s1}(\mathbf{v}_{s1})(1-O_{s2}(\mathbf{v}_{s2}))+(1-O_{s1}(\mathbf{v}_{s1}))O_{s2}(\mathbf{v}_{s2}), \tag{2}$$

where T_{s1} and T_{s2} are two neighboring tetrahedrons that share the face f_s.

Given input observations I, which can be an image or a point cloud, DefTet utilizes a neural network h to predict the relative offset for each vertex and the occupancy of each tetrahedron:

$$\{\Delta x_i, \Delta y_i, \Delta z_i\}_{i=1}^N, \{O_k\}_{k=1}^K = h(\mathbf{v}, I; \theta), \tag{3}$$

where θ parameterizes the neural network h. The deformed vertices are thus:

$$\mathbf{v}_i = [x_i + \Delta x_i, y_i + \Delta y_i, z_i + \Delta z_i]^T. \tag{4}$$

In terms of optimization for 3D reconstruction, DefTet directly optimizes the position offset, occupancy, and RGB color for each vertex. This approach is particularly effective in handling datasets with significant geometry and texture variations. For instance, in the NeRF dataset, which exhibits more substantial geometric and textural variations compared to the ShapeNet dataset, DefTet employs a strategy of mesh subdivision and resolution enhancement. Starting with a 40x40x40 resolution, the tetrahedral mesh is subdivided twice, resulting in a final resolution of 160x160x160. This method focuses on optimizing the complex objects in the NeRF dataset efficiently.

3.2. 3D Gaussian Splatting (3dgs)

The core idea of 3D-GS [4] is to model a static 3D scene explicitly with point primitives. Each point is parameterized as a scaled Gaussian with a 3D covariance matrix Σ and mean μ:

$$G(X) = e^{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)} \tag{5}$$

The covariance matrix Σ can be decomposed into a scaling matrix S and a rotation matrix R as:

$$\Sigma = RSS^TR^T \tag{6}$$

The projection of Gaussians from 3D space to a 2D image plane is implemented by a view transformation W and the Jacobian of the affine approximation of the projective transformation J. The covariance matrix Σ' in 2D space is computed as:

$$\Sigma' = JW\Sigma W^TJ^T \tag{7}$$

Point-based alpha-blend rendering is then applied, similar to the technique used in NeRF. The color and opacity at each point are determined by a 2D Gaussian with covariance Σ' and a learned per-point opacity. The color is typically defined by spherical harmonics coefficients.

4. Methods

Our methodology encompasses a two-stage process that involves the integration of 3D Gaussian Splatting (3G-DS) and Deformable Tetrahedral Meshes (DefTet) for robust 3D reconstruction from images [1].

4.1. 3D Gaussian Splatting (3G-DS)

Initially, images of an object are processed through 3GDS, which generates a Gaussian point cloud. This point cloud effectively represents the object's structure in a 3D Gaussian distribution format, capturing the essential geometric and spatial information.

4.2. Integration with DefTet

Once the Gaussian point cloud is obtained, it is fed, along with the original images, into the DefTet framework. In this stage, DefTet utilizes both the visual data from the images and the spatial information from the Gaussian point

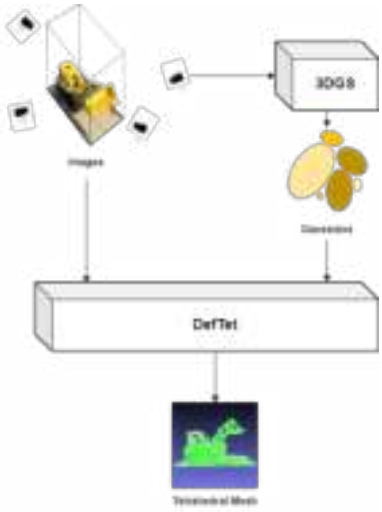

Figure 1. Our architecture

cloud to ensure mesh encapsulates the gaussian point cloud. The Gaussian point cloud primarily serves as an integral component in the training objective. It aids in guiding the reconstruction process, ensuring that the output meshes accurately reflect the underlying structure and details of the object as captured in the point cloud.

4.3. Training Objective:

Leveraging and extending DefTet's loss framework, we define our composite loss function, $\mathcal{L}_{all}(\theta)$, as follows:

$$\mathcal{L}_{all}(\theta) = \lambda_{recon}\mathcal{L}_{recon} + \lambda_{vol}\mathcal{L}_{vol} + \lambda_{lap}\mathcal{L}_{lap} + \lambda_{sm}\mathcal{L}_{sm} + \lambda_{del}\mathcal{L}_{del} + \lambda_{amips}\mathcal{L}_{amips}$$

We also introduce our unique loss terms, tailored to the specifics of our application, which will be discussed in subsequent sections.

4.4. Chamfer Loss with mahalanobis distance

To address the issue of non-positive semi-definite covariance matrices in our Gaussian representations, we first perturb the covariance matrix Σ with a small value ϵ, where ϵ is 10^{-6}, resulting in:

$$\hat{\Sigma} = \Sigma + \epsilon I \tag{8}$$

We then perform a Cholesky decomposition on $\hat{\Sigma}$ to obtain the matrix L. The Mahalanobis distance between a point y and a Gaussian with mean μ is typically defined as:

$$D_{\text{M}}(y, \mu, \Sigma) = \sqrt{(y-\mu)^T \Sigma^{-1} (y-\mu)} \tag{9}$$

Using the matrix L, this distance can be reformulated as:

$$D_{\text{M-L}}(y, \mu, L) = \|L^{-1}(\mu - y)\|_2 \tag{10}$$

We incorporate this distance measure into our Chamfer Loss formulation for sets X (representing Gaussians) and Y (a point cloud):

$$\mathcal{L}_{\text{chamfer-M}}(X, Y) = \frac{1}{|X|} \sum_{(\mu, L) \in X} \min_{y \in Y} \|L^{-1}(\mu - y)\|_2^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{(\mu, L) \in X} \|L^{-1}(\mu - y)\|_2^2 \tag{11}$$

Our final loss function, $\mathcal{L}$, is defined as follows:

$$\mathcal{L} = \mathcal{L}_{all} + \mathcal{L}_{\text{chamfer-M}} \tag{12}$$

5. Results

In the PSNR comparison[1], both DefTet and our method show closely matched results across various datasets. DefTet has a slightly higher average PSNR at 25.36 compared to our method's average of 25.07. While both methods show lower PSNR values than 3dgs, the focus here is on the comparison between DefTet and our model. Our approach demonstrates a competitive edge in datasets like "Hotdog" and "Materials," but sees a slight decrease in others such as "Drums" and "Mic." While our method slightly trails DefTet in average PSNR, there are noteworthy improvements in specific areas. In the "Materials" dataset, our method better captures lighting nuances that DefTet misses. For the "Drums" dataset, unlike DefTet, our model successfully represents the rubber at the end of the drum legs, a detail that DefTet overlooks. Additionally, in the "Ship" model, DefTet creates extraneous tetrahedrons outside the actual model, an issue our approach avoids.

These enhancements in our model can be attributed to the precision in placing Gaussians, which typically do not extend beyond the surface.

In the mesh comparison, both DefTet and our model exhibit the problem of flipping tetrahedrons, resulting in visible inverted tetrahedra in both meshes. This issue is particularly apparent in complex structures. Our model, designed to encompass all Gaussians within the tetrahedral mesh, occasionally captures a few Gaussians lying outside the surface, leading to small spikes extending outward, as seen in the ship dataset.

For the ficus dataset, our approach does not discard stretched tetrahedra, unlike DefTet, as we aim to enclose all Gaussians. This leads to a more faithful representation of the stretched areas. In the mic dataset, the initial breaking of tetrahedra in DefTet is mirrored in our model, affecting the continuity of structures like the wire in the mic, where

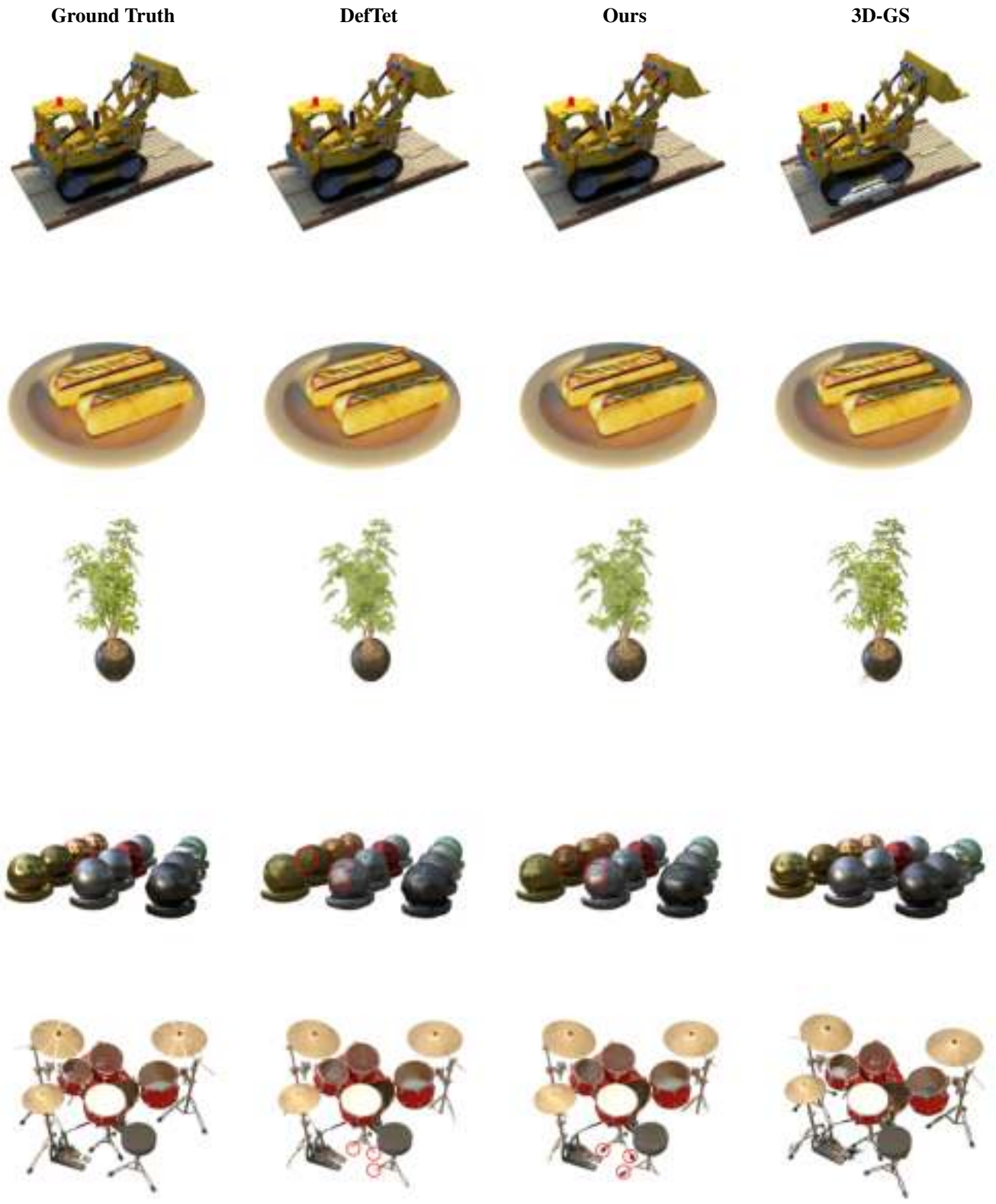

Figure 2. Comparative results across datasets (synthesized novel views).

Figure 3. Comparative results across datasets (continued).

PSNR ↑	Lego	Hotdog	Ficus	Materials	Drums	Mic	Ship	Average PSNR
3dgs	40.658	38.044	35.497	30.479	26.282	36.813	31.688	34.21
DefTet	27.717	32.013	24.333	22.806	21.036	26.230	23.407	25.36
Ours	27.756	32.028	24.327	22.927	20.906	23.720	23.830	25.07

Table 1. PSNR comparison across datasets

it appears broken into separate pieces. This fidelity to the DefTet structure, while ensuring Gaussian containment, underlines the nuances in our mesh reconstruction approach.

The results from the mean volume [2] and 0.05 quantile volume comparisons [3] reveal subtle differences between DefTet and our model across various datasets.

In the mean volume comparison, both methods exhibit similar values, with our model showing slightly lower mean volumes in "Materials" and slightly higher in "Hotdog" and "Ship." This indicates minor variations in how each method estimates the overall volume of the tetrahedral meshes.

The 0.05 quantile volume comparison, reflecting the number of tetrahedrons in this quantile, also displays close values between the two methods. Notably, in "Materials" and "Ship," our model shows a higher count of tetrahedrons, suggesting a difference in the density or distribution of tetrahedrons at this quantile level.

These results suggest that while both methods are comparable in terms of mean volume, there are slight differences in the distribution of tetrahedrons, as indicated by the 0.05 quantile volume data.

With and Without Mahalanobis Correction. $\mathcal{L}_{\text{chamfer}}$ measures the Euclidean distance between the centroids of vertices and Gaussian means, while $\mathcal{L}_{\text{chamfer-M}}$ incorporates the Mahalanobis distance, considering the covariance of the Gaussians.

5.1. Standard Chamfer Loss vs. Chamfer Loss with Mahalanobis Distance

We applied both loss functions to a dense swarm of Gaussians confined within a unit square. The $\mathcal{L}_{\text{chamfer}}$ is given by:

$$\mathcal{L}_{\text{chamfer}}(X,Y) = \frac{1}{|X|}\sum_{(\mu,L)\in X}\min_{y\in Y}\|(\mu-y)\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\min_{(\mu,L)\in X}\|(\mu-y)\|_2^2 \quad (13)$$

The experimental outcomes revealed that for a dense population of Gaussians within a unit square, $\mathcal{L}_{\text{chamfer}}$ is approximately equal to $\mathcal{L}_{\text{chamfer-M}}$. This is because the dense arrangement of the Gaussians compensates for the variance captured by the Mahalanobis distance, making the simpler $\mathcal{L}_{\text{chamfer}}$ approximately equal to $\mathcal{L}_{\text{chamfer-M}}$ in this context. Therefore, no significant enhancement in reconstruction fi-

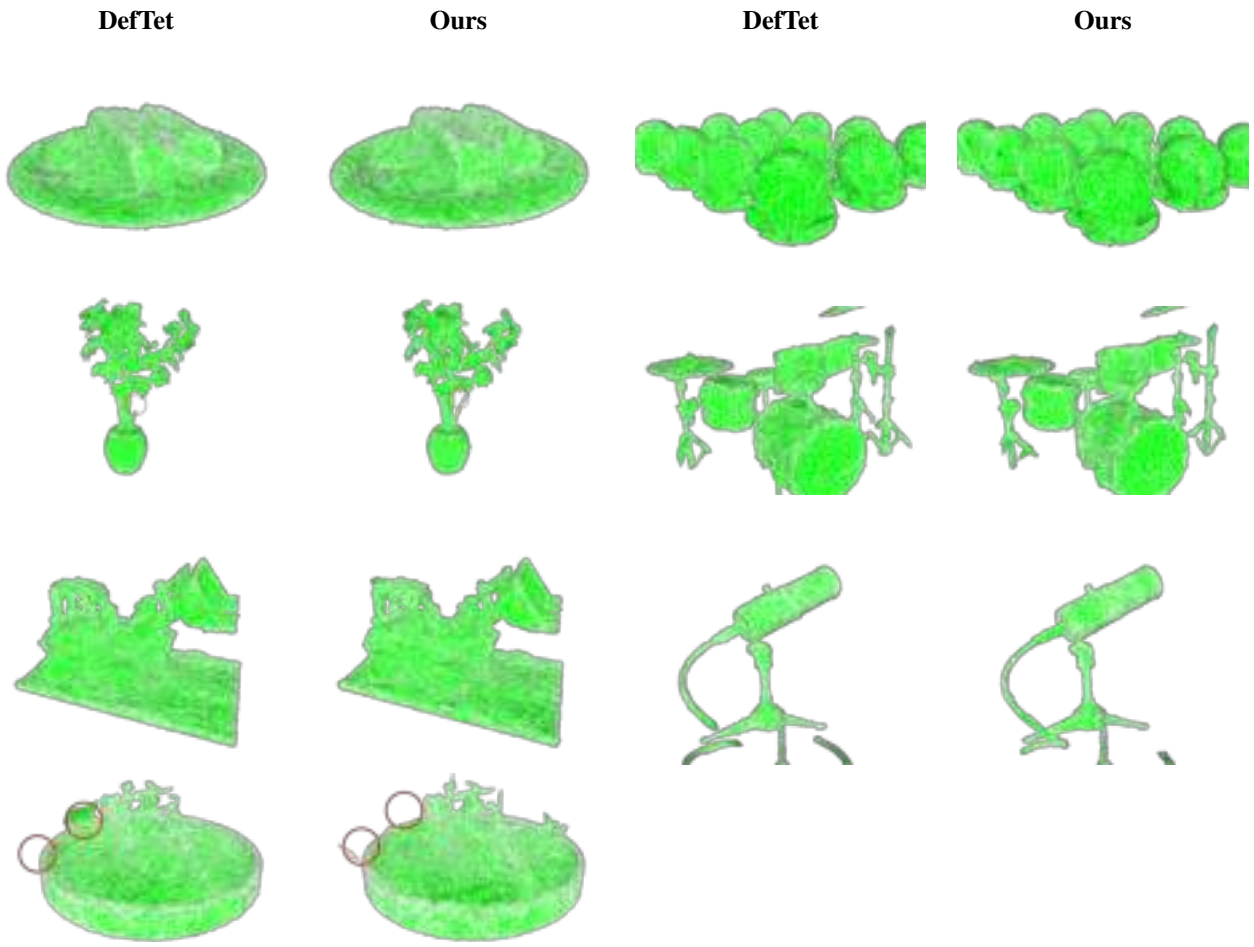

Figure 4. Mesh comparison between DefTet and our Model across datasets. Notice our model is better at not losing thin structures, but still has some outlier geometry.

Mean Volume	Lego	Hotdog	Ficus	Materials	Drums	Mic	Ship
DefTet	1.9354e-06	1.6752e-06	3.0106e-06	3.4488e-06	2.2100e-06	1.8343e-06	2.5061e-06
Ours	1.9311e-06	1.7045e-06	2.9700e-06	2.2452e-06	2.2452e-06	1.9151e-06	2.5300e-06

Table 2. Mean volume comparison across the datasets

delity was observed with the addition of Mahalanobis distance in the Chamfer Loss function.

6. Conclusion

In conclusion, we present a novel method for creating deformable meshes integrated with Gaussian splats, aimed at facilitating fast and efficient rendering. This approach offers a significant advantage as it encapsulates a Gaussian point cloud within the mesh. Consequently, this enables the deformation of the mesh while maintaining the ability to rapidly render these changes. This synergy between deformable meshes and Gaussian splats not only enhances the rendering speed but also preserves the detailed structure and dynamics of the mesh, making it particularly beneficial for applications requiring high fidelity and real-time interactivity.

The methodologies and findings from this study present several avenues for future exploration and application. One promising direction is the adaptation of our approach to large-scale scenes, which could significantly benefit urban planning, architectural design, and large-scale simulations for VR training environments. Moreover, the principles of our learned shape representation and efficient rendering techniques could be applied to dynamic scene reconstruction in autonomous vehicle navigation, where real-time processing of deformable objects is crucial.

0.05 Quantile Volume	Lego	Hotdog	Ficus	Materials	Drums	Mic	Ship
DefTet	18307	20547	4064	5701	7178	4063	33748
Ours	18501	20356	4098	6024	7077	3792	34335

Table 3. Number of Tetrahedrons in 0.05 Quantile comparison across the datasets

Further, the integration of our methods with deep learning architectures offers the potential for automated recognition and reconstruction of complex structures from sparse data, a development that could transform fields such as robotics and remote sensing. Lastly, the extension of our approach to incorporate temporal coherence could enable the creation of more sophisticated animations and simulations in the entertainment industry.

Limitations and Future Work

This study's limitations include the presence of flipped tetrahedrons within complex structures and spikes due to some Gaussians extending beyond the mesh surface. Additionally, the method sometimes retains stretched tetrahedra, affecting mesh fidelity. Future improvements can address these limitations by refining the mesh deformation process and improving Gaussian encapsulation within the mesh. Instead of encompassing all Gaussians, a more selective approach, targeting only those on or within the surface and discarding extraneous ones, would likely yield more accurate results. These enhancements can significantly improve the technology, extending its utility across a wider range of applications.

References

[1] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[2] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3D reconstruction. *Advances In Neural Information Processing Systems*, 2020.

[3] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[5] Jonas Kulhanek and Torsten Sattler. Tetra-NeRF: Representing neural radiance fields using tetrahedra. *arXiv preprint arXiv:2304.09987*, 2023.

[6] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 2021.

[7] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *SIGGRAPH 2012 courses*. ACM, 2012.

[8] Zackary PT Sin, Peter HF Ng, and Hong Va Leong. NeRFahedron: A primitive for animatable neural rendering with interactive speed. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2023.

[9] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. NeRF-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

A. Detailed Formulation of DefTet Loss Terms

This supplement provides an in-depth look at the individual loss terms used in the DefTet framework.

Occupancy Loss ($\mathcal{L}_{\text{occ}}$): Supervises occupancy prediction using Binary Cross Entropy.

$$\mathcal{L}_{\text{occ}}(\theta) = \sum_{k=1}^{K} \hat{O}_k \log(O_k) + (1 - \hat{O}_k) \log(1 - O_k), \quad (14)$$

where $\hat{O}_k$ is the ground truth occupancy for tetrahedron T_k.

Surface Deformation Loss ($\mathcal{L}_{\text{surf}}$): Aligns deformations with the target object's surface.

$$\mathcal{L}_{\text{surf}}(\theta) = \sum_{p \in S} \min_{f \in F} \text{dist}_f(p, f) + \sum_{q \in S_F} \min_{p \in S} \text{dist}_p(p, q), \quad (15)$$

where S and S_F are sets of points on the target surface and the surface of F, respectively. $\mathcal{L}_{\text{recon}}$ is defined as

$$\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{surf}} \quad (16)$$

Laplacian Loss ($\mathcal{L}_{\text{lap}}$): Penalizes relative positional changes of neighboring vertices.

$$\mathcal{L}_{\text{lap}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \Delta v_i - \frac{1}{|N(v_i)|} \sum_{v \in N(v_i)} \Delta v \right\|^2, \quad (17)$$

where $N(v_i)$ is the set of neighbors of vertex v_i, and Δv_i is the deformation of vertex v_i.

Delta Loss ($\mathcal{L}_{\text{del}}$): Encourages local deformation by penalizing large vertex deformations.

$$\mathcal{L}_{\text{del}} = \frac{1}{N} \sum_{i=1}^{N} |\Delta v_i|^2. \quad (18)$$

EquiVolume Loss ($\mathcal{L}_{\text{vol}}$): Ensures similar volumes for each deformed tetrahedron.

$$\mathcal{L}_{\text{vol}} = \frac{1}{K} \sum_{k=1}^{K} |V_k - V|^4, \tag{19}$$

where V_k is the volume of tetrahedron T_k, and V is the average volume.

AMIPS Loss ($\mathcal{L}_{\text{amips}}$): Penalizes tetrahedral distortion.

$$\mathcal{L}_{\text{amips}} = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{tr}(J_k^T J_k)}{(\det(J_k))^{\frac{2}{3}}}. \tag{20}$$

J_k is the Jacobian matrix of the deformation for tetrahedron T_k.