3D Scanning & Spatial Learning
Project 2: Dynamic Geometry Reconstruction

Marco Busch
TUM
marco.busch@tum.de

Vlad Bratulescu
TUM
vlad.bratulescu@tum.de

Abstract

Since the introduction of Gaussian Splatting, it has been adapted separately for static surface reconstruction and dynamic reconstruction. However, the integration of these two tasks into a single training framework remains underexplored. We propose an approach that combines techniques from RaDe-GS and E-D3DGS to address this challenge. Using multi-view videos as input, we use a multilayer perceptron (MLP) to derive deformations that determine the Gaussian parameters for a specific time step. Simultaneously, we apply normal consistency regularization to ensure the Gaussians align with the underlying surface. We evaluate our method in the domain of multi-view dynamic head reconstruction using subjects from the NeRSemble dataset. In the context of dynamic head reconstruction, the mouth interior poses a significant challenge due to its highly deformable geometry, frequent occlusions, and topological variability. To mitigate these issues, we introduce targeted strategies that specifically improve the reconstruction of tongue dynamics. Our code is publicly available at https://github.com/buma13/E-D3DGS.

1. Introduction

Dynamic geometry reconstruction involves representing a moving object as a temporally consistent geometry over time. Recently, NeRF-based methods have been extended to dynamic scenes, but due to their inherent computational intensity, they haven't been explored further. Since Gaussian Splatting addresses this computational constraint while still maintaining differentiable rendering, its extension to dynamic settings is naturally being investigated. Moreover, research into extracting geometry from Gaussians is gaining more attention but hasn't been fully explored. One use case for dynamic geometry reconstruction is the generation of 3D human avatars that can model complex facial movements and expressions.

Extracting a 3D mesh of real-world objects has long been a core topic in computer vision. Classical methods rely on sensors like structured-light scanners or multi-view camera systems and structure form motion (SfM) to capture point clouds, which are then fused (e.g., via ICP) and converted into meshes using techniques such as Poisson surface reconstruction [5]. Implicit representations, such as signed distance functions integrated from RGB-D data (e.g., via TSDF) combined with TSDF fusion [9] or Marching Cubes [7] for mesh extraction, offer another alternative.

More recently, neural implicit methods like Neural Radiance Fields (NeRF) [8] have gained popularity for photorealistic novel view synthesis. Building on this trend, Gaussian Splatting has taken the computer vision world by storm. It uses 3D Gaussians as an explicit representation, where each Gaussian has its own position, orientation, scale, opacity, and color. By using splatting, these can be rendered into an image. Its main benefit is that the rendering process is fully differentiable, allowing for the optimization of Gaussian parameters using RGB supervision.

While Gaussian Splatting delivers state-of-the-art results in multi-view synthesis, this representation doesn't explicitly model geometry. However, works like 2D Gaussian Splatting [4] have extended 3D Gaussians to serve as a geometry representation by applying specific regularizations to ensure the Gaussians align with the underlying surface. Since Gaussian Splatting also enables the rendering of depth maps, TSDF fusion can be used to generate a 3D model. However, these models often result in noisy and incomplete meshes. To address this, Gaussian Opacity Fields (GOF) [13] further refined the method to achieve better results. One of its main contributions was the use of Marching Tetrahedra for mesh extraction instead. However, this approach comes with a high training time. Rasterizing Depth in Gaussian Splatting (RaDe-GS) [14] tackles the computational overhead while still maintaining strong performance in geometry extraction. This method combines the strengths of both approaches, offering faster training times and enabling the use of Marching Tetrahedra as the extraction technique.

Moreover, several works have explored dynamic Gaus-

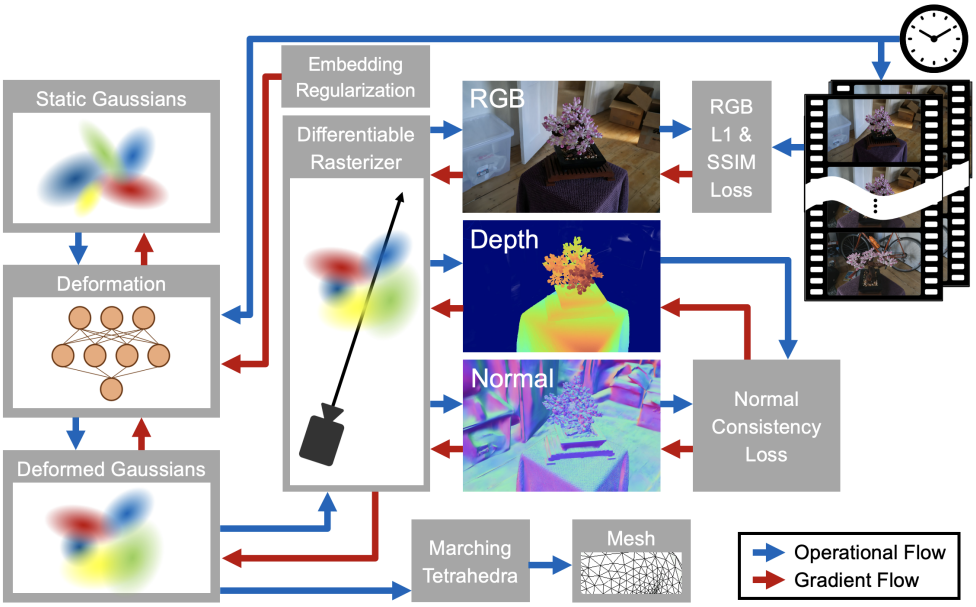


Figure 1. Based on a set of static 3D Gaussians, initialized by SfM Points, a deformation network predicts per-frame transformations to generate deformed Gaussians (see 3.1). These are rasterized into RGB, depth, and normal images using a differentiable renderer. The system is supervised with RGB (L1 and SSIM) and normal consistency regularization (see 3.2), using multi-view image sequences as input. Additionally, an embedding regularization term is applied to stabilize the deformation space. The deformed Gaussians can be converted into a mesh via Marching Tetrahedra. Blue arrows denote the operational flow, while red arrows indicate the gradient flow during optimization

sian Splatting, an area that is still being actively researched. One such method is Embedding-Based Deformable 3D Gaussian Splatting (E-D3DGS) [1], which uses a deformation field to model the movement of Gaussians. However, it doesn't introduce any modifications to the Gaussians to align them with the underlying surface.

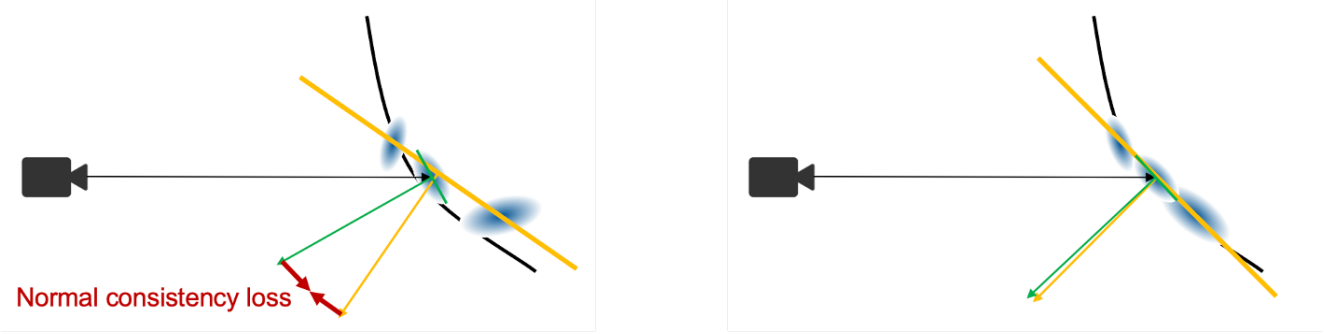
The simplest approach to achieving dynamic geometry reconstruction with Gaussian Splatting would be to apply a static method like RaDe-GS to every time step of the scene. However, this not only leads to long training times but also produces temporally inconsistent 3D models. Our main contributions involve combining a static surface reconstruction method with dynamic scene reconstruction. We are testing our framework on the NeRSemble [6] dataset, which consists of multi-view images of faces. Finally, we apply domain-specific adjustments to enable the modeling of more complex scenes, such as tongue movement.

2. Related Work

2D-GS [4] In the original 3DGS, inconsistent depths across viewpoints occur due to ray-Gaussian intersections, resulting in poor geometry reconstruction. In contrast, 2DGS employs 2D Gaussian primitives, providing precise geometry representation through explicit ray-splat intersections. This perspective-correct splatting enhances reconstruction quality. Furthermore, 2DGS introduces depth distortion and normal consistency regularizations to better align the Gaussian primitives with the underlying surfaces. For mesh extraction, it uses TSDF fusion on the rendered depth maps.

GOF [13] TSDF fusion struggles to accurately model thin structures and reconstruct unbounded scenes. Resorting to high-resolution voxel grids for TSDF leads to the creation of large meshes. To address this, GOF proposes a Gaussian opacity field based on ray-Gaussian intersection to determine the opacity of a given 3D point. This enables the direct identification of a level set, allowing for surface reconstruction from Gaussians without relying on Poisson reconstruction or TSDF. Instead, it employs the Marching Tetrahedra algorithm. Additionally, GOF approximates the surface normals of 3D Gaussians as the normals of the ray-Gaussian intersection plane, enabling the incorporation of normal-based regularization, as done in 2DGS.

RaDe-GS [14] extends both previous methods. Due to the ray-tracing-based approach used to compute opacity in GOF, training suffers from computational overhead. To address this, RaDe-GS proposes an efficient rasterization method to compute depth and normal maps for Gaussian splats. Additionally, this method incorporates the 3D filter proposed in Mip-Splatting [12] to tackle aliasing issues. Originally, RaDe-GS relied on TSDF fusion as its extraction method. However, it has also adopted the extrac-



(a) before regularization

(b) after regularization

Figure 2. The intuition behind normal consistency regularization is as follows: The green and yellow arrows indicate the normal of the Gaussian and the gradient of the depth map, respectively. The black curve represents the underlying surface. After regularization, the two vectors align with each other.

tion technique used in GOF. To achieve this, it computes opacities in ray space and then applies the Marching Tetrahedra algorithm to the triangulated points surrounding the Gaussians. Overall, RaDe-GS manages to improve Gaussian Splatting's shape reconstruction capability while maintaining its training and rendering efficiency.

E-D3DGS [1] solely models the deformation of Gaussians across different time frames. For this purpose, it employs a decoder as a deformation field, which takes per-Gaussian embeddings and temporal embeddings as input. The temporal embeddings are decomposed into coarse and fine embeddings, allowing different decoders to separately learn slow and detailed movements. Additionally, this method applies local smoothness regularization to the per-Gaussian embeddings to ensure that the deformations of neighboring Gaussians are similar.

DynaSurfGS [2] similarly attempts to model the deformation of Gaussians over time. To achieve this, it uses a Hex-Plane to encode spatial and temporal variations instead. Subsequently, it applies an MLP to the features encoded by the Hex-Plane to obtain the deformations, which are then applied to the Gaussians. Through ARAP regularization, the Gaussian points across different time steps are constrained to adhere to the principle of local rigidity. Moreover, it uses TSDF fusion to achieve mesh extraction from the rendered depth maps.

GSTAR [15] approaches the dynamic surface reconstruction problem differently by using an adaptable mesh that is updated at every time frame, ensuring time-consistent geometry. A Gaussian is attached to each face of the mesh and moves with it. In regions experiencing topology changes, Gaussians are detached from the faces. These Gaussians are then used to generate new geometry and update the original mesh. Moreover, to better reconstruct large movements, the mesh is updated at the beginning of each time frame using scene flow warping. Similarly, regularizations are employed to ensure surface continuity and maintain local geometry.

3. Method

Our method takes multiview RGB videos as input. We seek to integrate, within a unified framework, the dynamic representation of E-D3DGS [1] with the normal consistency regularization initially introduced in 2DGS [4] to ensure the alignment of Gaussians with the underlying surface. Our research primarily concentrates on the dynamic surface reconstruction of human avatars. Consequently, we implement domain-specific measures to enhance the reconstruction of tongue movements, which have proven difficult. To address this, during training, we preferentially sample frames featuring the tongue. Additionally, we utilize tongue-specific Gaussians, to which we apply distinct loss functions. Finally, we employ Marching Tetrahedra, as introduced in GOF [13] and RaDe-GS [14], to extract the geometry at a specific time step. An overview of our method can be found in Figure 1. Please refer to the supplementary section for additional experiments.

3.1. Dynamic representation

As done in E-D3DGS [1], we represent the motion of a scene as a single set of Gaussians, on which we apply an MLP to derive the deformation for a given time step. The MLP takes as input a per-Gaussian embedding combined with a temporal embedding. The output consists of the per-Gaussian deformation in position, rotation, scaling, opacity, and color. Moreover, to ensure that neighboring Gaussians exhibit similar deformations, both E-D3DGS and our approach employ local smoothness regularization on the per-Gaussian embeddings. Specifically, neighboring Gaussians should intuitively have similar embeddings as well. To optimize the parameters of the Gaussians and the weights of the MLP, we use a color L1 loss combined with an SSIM

loss.

3.2. Normal consistency regularization

Furthermore, as geometry-specific regularizations have shown improved surface reconstruction, we also employ a normal consistency regularization, as introduced in 2DGS [4]. The normal consistency term minimizes discrepancies between the rendered normal map and the gradient of the rendered depth, ensuring alignment between the geometries defined by depth and normals. Figure 2 provides an intuition behind this method.

3.3. Tongue-specific Gaussians

We also employ Gaussians that are solely responsible for reconstructing the tongue. We do this to enhance its dynamic and geometric reconstruction. To achieve this, we add 5,000 points to our initial COLMAP [11] point cloud, from which the Gaussians are subsequently generated. These 5,000 points are classified as "tongue" and are located in the mouth region. Figure 3 illustrates this. In this case, the Gaussians not only inherit the positions of the initialization points but also their classifications. Additionally, we use mask images of the recorded person as input to our method. These provide segmentations for facial parts such as the eyes, mouth, tongue, and etc. We have also modified the rasterizer from [14] to return the alpha mask of only those Gaussians classified as "tongue." This enables us to compute a loss between the ground truth mask and the rendered one. Filtering Gaussians based on their class also allows us to render RGB images exclusively of the tongue Gaussians or the non-tongue Gaussians. We further utilize these two renderings to compute additional loss function. Given a ground truth binary tongue mask $M^{\text{tongue}} \in \{0,1\}^{H\times W}$ for each camera view, we compute a masked L1 RGB loss that only considers the tongue region. Let $\hat{\mathbf{C}}^{\text{tongue}}$ be the rendering only considering tongue Gaussians and $\mathbf{C}$ the ground truth image, both masked by M^{tongue}. The loss is defined as:

$$\mathcal{L}_{\text{tongue}} = \frac{1}{\sum_i M_i^{\text{tongue}}} \sum_i M_i^{\text{tongue}} \cdot \left\| \hat{\mathbf{C}}_i^{\text{tongue}} - \mathbf{C}_i \right\|_1 \quad (1)$$

This loss is only applied when the tongue is sufficiently visible, i.e. the tongue region makes up more then 0.5% of the image. We also define a second RGB loss over all pixels not belonging to the tongue. Using the complement mask $M^{\text{non}} = 1 - M^{\text{tongue}}$, and the rendering without tongue Gaussians $\hat{\mathbf{C}}^{\text{non-tongue}}$, the loss is:

$$\mathcal{L}_{\text{non-tongue}} = \frac{1}{\sum_i M_i^{\text{non}}} \sum_i M_i^{\text{non}} \cdot \left\| \hat{\mathbf{C}}_i^{\text{non-tongue}} - \mathbf{C}_i \right\|_1 \quad (2)$$

This decomposition encourages the model to prioritize accurate reconstruction of the tongue, while still maintaining overall image consistency elsewhere. Figure 4 showcases the two types of renderings. Additionally, we regularize the embeddings of the tongue Gaussians in a manner similar to that described in [1], but without the nearest-neighbor search. The intuition here is that all tongue Gaussians should exhibit similar deformations.

Figure 3. Illustration of the 5000 tongue-specific points (red color) in the mouth region of the initialization pointcloud we receive from COLMAP.

3.4. Sampling strategy

To enable the deformation network to better learn the dynamic representation of the tongue, we sample frames containing the tongue more frequently during training. Thus, 50% of the time, we sample only tongue frames, while the remainder are sampled randomly.

4. Results

4.1. Dataset

We tested our method on the NeRSemble dataset [6]. The dataset provides recorded sequences from 16 different views, covering a wide range of facial dynamics, including head motions, natural expressions, emotions, and spoken language. Moreover, it includes ground truth point clouds along with face segmentation masks for each time step. To validate our method, we primarily used two different individuals from this dataset, focusing on tongue-intensive scenes.

4.2. Metrics

To measure the performance of our method, we employ the standard PSNR and SSIM metrics to evaluate the RGB renderings. We compute these metrics based on a single test camera, averaged across all time steps. Figure 5 showcase

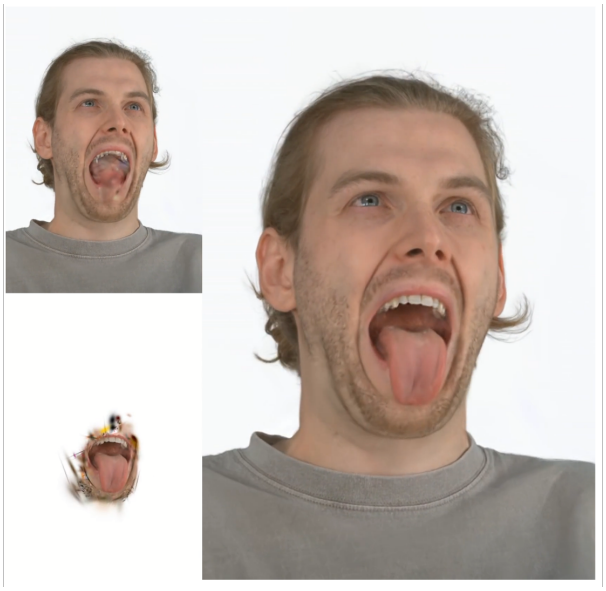


Figure 4. On the left, stacked vertically, we show the renderings of both Gaussian classes. On the right, we have the combined rendering.

the renderings of our method side by side with the ground truth, while the computed PSNR and SSIM values for both persons are presented in Table 1.

To assess the quality of our extracted mesh, we use a unidirectional distance from each point in the ground truth point cloud to the nearest surface point on our mesh. Additionally, we compute the cosine similarity between the normal vectors associated with these closest points. We average both metrics across all time steps in the scene. These computed metrics are illustrated in Figure 7 and presented in Table 2.

We also present qualitative results of our extracted meshes in Figure 6.

4.3. Ablations

In Figure 8, we present the results of our ablation study. For each time step, we trained RaDe-GS independently and compared its performance to that of our method. Furthermore, we separately trained the scene without the tongue-specific Gaussians and without our sampling strategy to assess their individual contributions to the quality of the extracted meshes. The position and normal metrics are averaged across all time steps.

When RaDe-GS is applied to each individual time step, it achieves the highest performance according to our proposed metrics, and the resulting extracted mesh shows greater smoothness. However, these meshes lack temporal consistency, resulting in small mesh fragments that appear and disappear in front of and behind the subject. Moreover, training RaDe-GS for every time step significantly increases

Person	PSNR	SSIM
407	20.809	0.844
037	24.377	0.910

Table 1. Averaged PSNR and SSIM values across all time steps of our method.

Person	Unidirectional distance	Cosine similarity
407	1.14mm	0.909
037	0.73mm	0.920

Table 2. Averaged unidirectional distance and cosine similarity metrics across all time steps of our method.

Figure 5. GT images (right) with the renderings (left) of our method on subjects 407 and 037 from the NeRSemble dataset.

training duration. A comparison of training times is provided in Table 3.

Removing the tongue-specific Gaussians leads to a slight decline in performance, as measured by our positional metric, with this reduction most noticeable in the tongue region. The extracted mesh also struggles to accurately reconstruct the tongue. Overall, disabling our tongue-specific sampling strategy results in the most substantial performance drop.

Figure 6. Extracted meshes for subjects 407 and 037 using our method.

Method	Training time
RaDe-GS	21h
Ours	3h 40m

Table 3. Training times of RaDe-GS and our method trained on a whole scene of subject 407.

5. Conclusion

We introduce a method that integrates state-of-the art surface and dynamic reconstruction techniques based on Gaussian Splatting. Our experiments primarily focus on human avatars, with our approach ensuring temporal consistency across challenging scenes, such as those featuring subjects with moving tongues. Additionally, our method substantially reduces training time compared to applying static surface reconstruction to each time step individually. However, the extracted meshes show a lack of smoothness, and accurately reconstructing key features like the tongue remains a challenge. Looking ahead, our method could be extended not only to enhance performance in such scenarios but also to address more complex cases, including scenes with hair movement. Inspired by the recently published GSTAR method, we believe that updating a canonical mesh offers a promising approach to overcoming our current limitations.

References

[1] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024. 2, 3, 4

[2] Weiwei Cai, Weicai Ye, Peng Ye, Tong He, and Tao Chen. Dynasurfgs: Dynamic surface reconstruction with planar-based gaussian splatting. *arXiv preprint arXiv:2408.13972*, 2024. 3

[3] Vien Cheung, Stephen Westland, David Connah, and Caterina Ripamonti. A comparative study of the characterisation of colour cameras by means of neural networks and polynomial transforms. *Coloration Technology*, 120:19–25, 2004. 1

[4] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, page 1–11. ACM, 2024. 1, 2, 3, 4

[5] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, page 61–70, Goslar, DEU, 2006. Eurographics Association. 1

[6] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Trans. Graph.*, 42(4), 2023. 2, 4

[7] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 1

[8] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[9] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 1

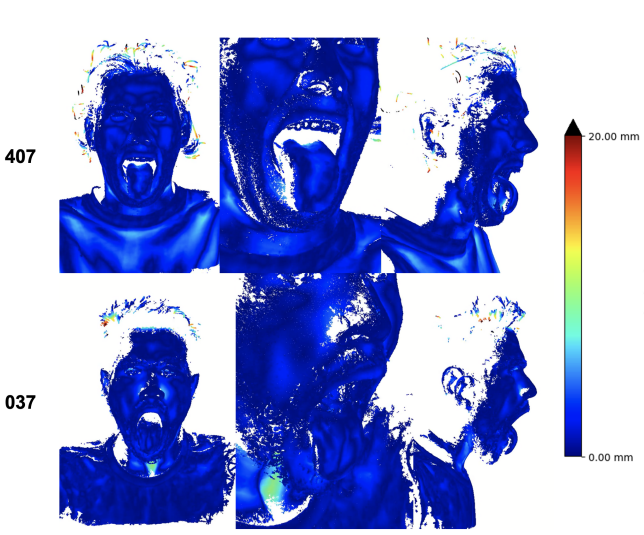
[10] Bui Tuong Phong. Illumination for computer generated pictures. *Seminal graphics: pioneering efforts that shaped the field*, 1975. 1

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[12] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2024. 2

[13] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 1, 2, 3

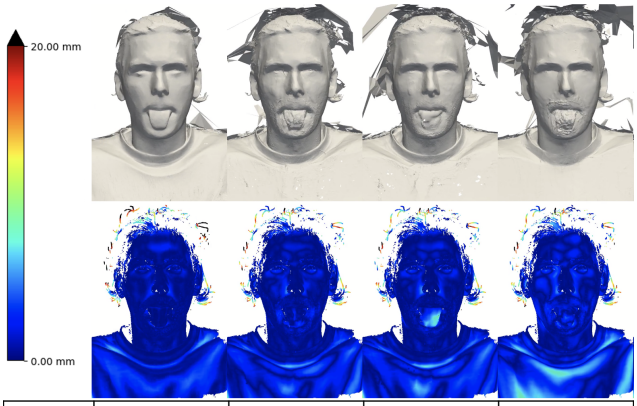
[14] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting, 2024. 1, 2, 3, 4



(a) Unidirectional distance metric

(b) Cosine similarity metric

Figure 7. Side-by-side view of the color-encoded unidirectional distance (left) and cosine similarity (right) metrics for a given time step. The unidirectional distance is measured in millimeters, while the cosine similarity value indicates whether the two normal vectors are orthogonal to each other (0) or aligned in the same direction (1).



	RaDe-GS Sequential	Ours	w/o Tongue gaussians	w/o Tongue sampling
Position	1.04 mm	1.14 mm	1.22 mm	1.65mm
Normal	0.921	0.909	0.909	0.902

Figure 8. Ablation study for subject 407.

[15] Chengwei Zheng, Lixin Xue, Juan Zarate, and Jie Song. Gstar: Gaussian surface tracking and reconstruction, 2025. 3

3D Scanning & Spatial Learning
Project 2: Dynamic Geometry Reconstruction

Supplementary Material

This supplementary material includes further details on experiments, visualization methods and dataset.

6. Other visualization methods

6.1. Phong shading

To evaluate geometry reconstruction, one would typically extract the mesh from our trained Gaussians. However, since this process is computationally intensive, we explored faster alternatives for performance analysis. Instead, we utilized rendered normal maps to compute the Phong reflection model [10] for each frame. This approach enables a visual assessment of our trained scene's performance. Figure 9 illustrates our visualization.

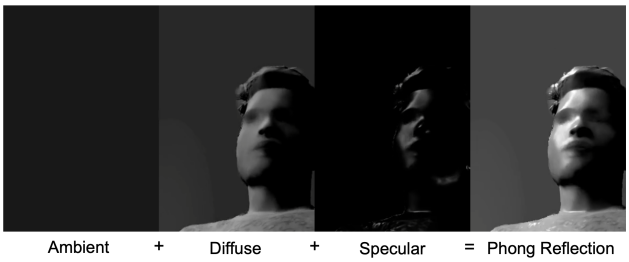


Figure 9. The Phong reflection model and all its components.

6.2. Gaussians to Ellipsoid Meshes

Additionally, the project instructors provided a script that converts trained Gaussians from our method directly into ellipsoid meshes. As shown in Figure 10, this visualization confirms that the Gaussians align with the underlying surface. However, it lacks fine detail, particularly in the mouth region. For this reason, we relied more on Phong shading visualization to analyze the quality of the mesh.

7. Further experiments and details

7.1. Color correction

Cameras don't always capture colors accurately due to for example sensor limitations. In the case of NeRSemble, the subject is recorded from 16 different cameras. As a result, the images across all cameras suffer from inconsistent colors, white balance, contrast, etc. Since Gaussian Splatting uses the RGB information to optimize the parameters of the Gaussians, it results in the Gaussians trying to compensate for this effect . This results in transparent "stripes" through the scene when viewing from different

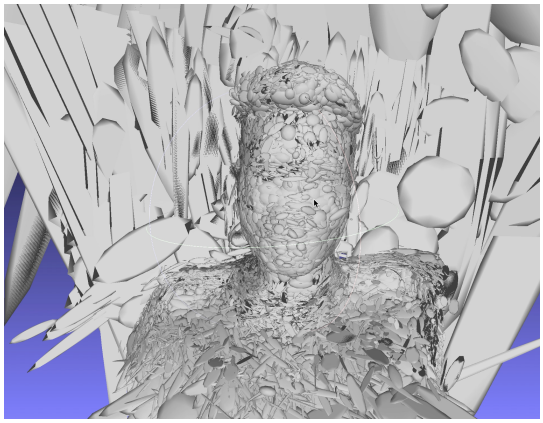


Figure 10. Conversion of Gaussians directly to ellipsoid meshes.

views. To account for this, we applied color correction using the method from [3] by using the color calibration matrices provided by NeRSemble. Figure 11 and Table 4 show some comparisons of our method with and without color correction.

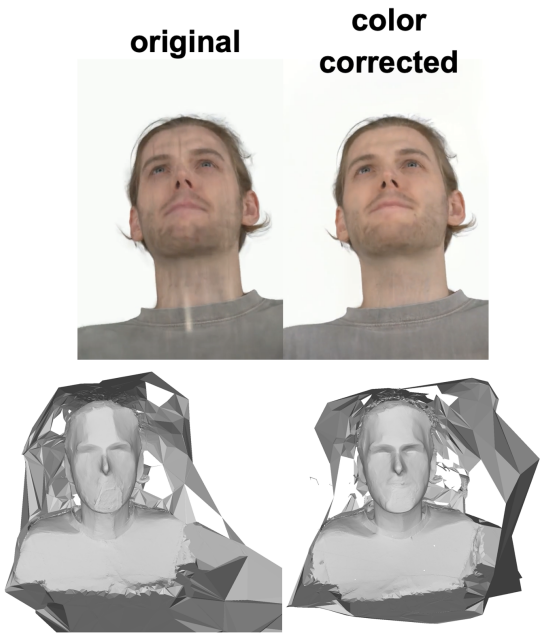


Figure 11. Renderings and mesh comparisons of our method with and without color correction. In the uncorrected case, the transparent stripes are visible in the rendered images. Color correction leads to slightly more detailed meshes.

Method	PSNR	SSIM
Original	19.711	0.817
With color correction	20.367	0.835

Table 4. PSNR and SSIM comparisons for with and without color correction.

7.2. COLMAP supervision

To investigate whether the MLP requires additional supervision to better reconstruct the scene's dynamics, we incorporated COLMAP point clouds from the NeRSemble data set as supervision. Specifically, for each timestep during training, we sampled 4,000 points from the corresponding point cloud and computed the L2 loss with the nearest Gaussians. The choice of 4,000 points was made to balance memory constraints. As shown in Figure 12, this approach resulted in worse rendering metrics without significantly improving the reconstruction of details such as the tongue. Additionally, since this step is performed at every sampled time step, it considerably increases training time.

Figure 12. The effect of point cloud supervision on rendering metrics.

7.3. Sampling strategies

7.3.1. Tongue sampling probabilities

We experimented with different sampling probabilities for the tongue, as shown in Figure 13. The best performance was achieved by sampling at least 50% of the frames from those containing the tongue.

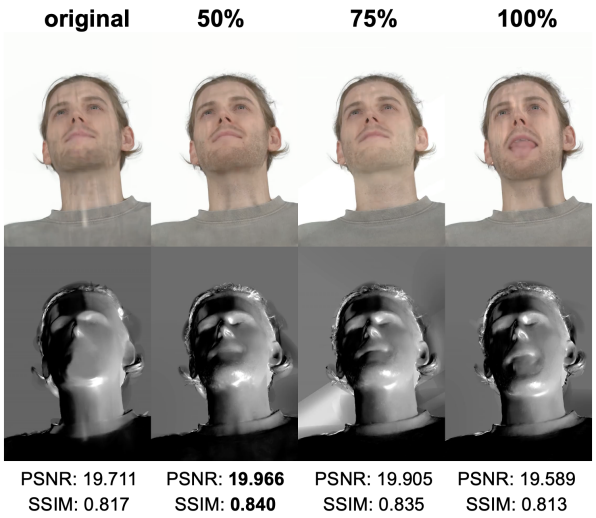


Figure 13. Rendering metrics for different tongue sampling probabilities.

7.3.2. Sequential sampling

We also experimented with a sequential sampling strategy, aiming to improve dynamic reconstruction by feeding frames sequentially into our MLP. The idea was that this approach would make learning the dynamics more intuitive. However, it led to earlier timesteps being forgotten.

To mitigate this, we tested two strategies: (1) extending the training duration on the initial frame to establish a robust foundation for the dynamics and (2) periodically sampling past frames to prevent the MLP from forgetting prior movements. The latter yielded better dynamic reconstruction but still underperformed compared to our baseline. Figure 14 illustrates our experiments.

7.4. Kernel size and 3D filter findings

A main part of our work was integrating the strategies from RaDe-GS into the E-D3DGS method. To do so, we needed to replace the E-D3DGS renderer with one capable of rendering both depth and normal maps, like the one already implemented for RaDe-GS. However, after making this switch, we observed a drop in rendering performance.

Upon investigation, we found that the original Gaussian Splatting renderer includes a 2D dilation filter applied in screen space for low-pass filtering. While this improves rendering quality, it introduces artifacts when camera views deviate from those seen during training, such as during zoom-in or zoom-out movements. This issue has been documented by the authors of [12], who replaced the 2D filter with a low-pass 3D filter applied directly to Gaussian primitives in 3D space.

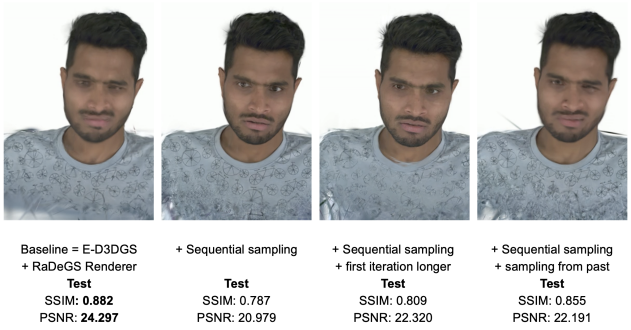


Figure 14. Comparison of our sequential sampling variants with the Baseline (E-D3DGS + RaDeGS Renderer). While the Baseline achieves higher average SSIM and PSNR scores across all test frames, it appears more blurred in individual frames. This is due to better motion modeling, which helps in overall metrics but reduces per-frame sharpness. In contrast, sequential sampling yields sharper reconstructions matching the last trained frames, but captures less dynamic variation which leads to lower average scores

The RaDe-GS renderer, based on this work, disables the 2D filter by default. As a result, using it in E-D3DGS initially led to worse performance, as shown in Figure 15. However, incorporating the 3D filter improved performance, as illustrated in Figure 16.

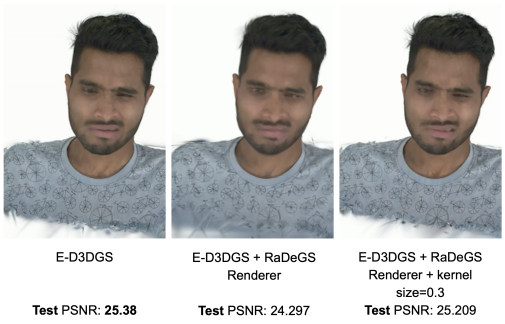


Figure 15. The effect of switching renderer on performance (middle). On the far right, we added back the 2D dillation filter.

8. Hair scene

We also tested our method on a more challenging scene featuring significant hair movement. Our initial results (Figure 17) indicate that further refinements are necessary to improve the joint reconstruction of both renderings and geometry in such complex scenarios.

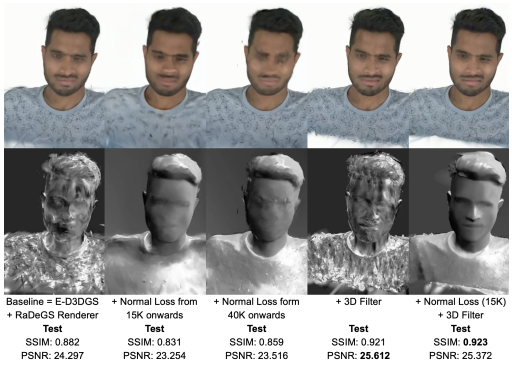


Figure 16. Visualization of performance analysis using the Phong shading. When applying the normal consistency loss, we a get a smooth shading. The fourth column illustrates the addition of the 3D filter and its effect on performance, compared to the RaDe-GS renderer with the 2D dillation filter being disabled.

Figure 17. Our results on a scene containing hair movement.