**3GPP TSG-SA WG4 Meeting #131 S4-250066**

**Geneva, CH, 17 – 21 February 2025**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.2* | | | | | | | | |
| **PSEUDO CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **26.956** | **CR** | **pseudo** | **rev** | **-** | **Current version:** | **0.2.1** |  |
|  | | | | | | | | |
| *For* ***[HE](http://www.3gpp.org/3G_Specs/CRs.htm" \l "_blank)******[LP](http://www.3gpp.org/3G_Specs/CRs.htm" \l "_blank)*** *on using this form: comprehensive instructions can be found at  <http://www.3gpp.org/Change-Requests>.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME | **X** | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | [FS\_Beyond2D] Representation Format - 3D Gaussian Splatting (3DGS) | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | China Mobile Com. Corporation, Qualcomm (?), Tencent | | | | | | | | | |
| ***Source to TSG:*** | SA4 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | FS\_Beyond2D | | | | |  | ***Date:*** | | | 2025-02-07 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **B** |  | | | | | ***Release:*** | | | Rel-19 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | During SA4#129-e, a proposal（S4-241519）on 3D Gaussian Splatting was reviewed and received some constructive comments from Qualcomm experts.  We believe Gaussian splats will be a relevant representation formats, but it is too early to come up with a concrete definition and a set of work flows.   * What is provided in the description is an incomplete summary of: [1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Trans. Graph. 42, 4, Article 139 (August 2023), 14 pages. https://doi.org/10.1145/3592433 with omissions and incomplete statements. * We also point that this Inria paper (and especially the limited, incomplete summary) is fairly outdated (12+ months) material. * The basic representation of 3D Gaussians in a trained model has remained the same by and large, but the training & rendering pipelines have evolved quite a bit with several variants available even for static representations (3D-GS), and let alone for dynamic representations (4D-GS). Depending on a use case, people would certainly choose a most appropriate variant, rather than going back to the seminal paper. * Based on this it just shows there is lots of research and new aspects being developed every day and it is not expected that we can have a convergence to a stable representation format any time soon.   A few very specific comments on the document:  For Definition:   * You copied the word-for-word set of parameters for 3D Gaussian Splatting from the original Inria paper, except for:   Scale (3x3 matrix) : this is insufficient as shape & scale/size of 3D Gaussians are represented using an anisotropic 3D covariance matrix Σ. Alternatively, instead of recording the covariance matrix, one can decompose it as scale/scaling and rotation matrices. The following quote from the paper: The covariance matrix Σ of a 3D Gaussian is analogous to describing the configuration of an ellipsoid. Given a scaling matrix 𝑆 and rotation matrix 𝑅, we can find the corresponding Σ: Σ = 𝑅𝑆𝑆𝑇𝑅𝑇   * The original paper shows how view-dependent coloring can be described using (a) Spherical Harmonic (SH) coefficients. Instead, your description breaks it down into (i) RGB color and (ii) Material Properties.   Only on a very high level one could we say that (i)+(ii) = (a). During rendering stage SH neatly encodes all that rasterizer needs to know about the color appearance of a Gaussian given a viewing direction. In contrast, if starting from Material properties, rasterizer would need to deploy a specialized per-material shader just to reproduce this view-dependent color information. In summary, (a) = look-up-table, while (i)+(ii) = extra processing, potentially very expensive.  For pipeline:   * Before jumping into Structure from motion (SfM) outlined in Figure 4.3.x.1-2, one needs to determine estimates for 6DOF poses corresponding to each training image (view), and also camera intrinsic parameters. These 6DOF poses (extrinsics), camera intrinsics, as well as RGBs are then processed in SfM block to obtain: (i) 3D point cloud - i.e., 3D projection of salient points observed in multiple images, as well as (ii) refined camera intrinsics and poses, optimized in a large nonlinear solver system. The inputs to 3DGS training are: (1) Refined 6DoF poses (camera extrinsics), (2) Refined (or original) camera intrinsics - a.k.a., projection matrices, (3) Images/RGBs corresponding to these poses and intrinsics, and (4) 3D points - point cloud - obtained using SfM. COLMAP - a CUDA-accelerated perception estimator, which is only one of several available tools one could use - generates (1), (2), and (4) starting from (3). In contrast, as an example, our internal perception pipeline can estimate all the above also starting from (3) in a fraction of the time, also on Android. If we don't want to single out our perception stack, we could cite any from a number of alternative tools, like GLOMAP for instance... * You do not mention once in the text details about Adaptive Density Control (or Adaptive Control - see the block in Figure 4.3.X.1-3), while this is perhaps the most important contribution of the Inria paper, and certainly of 100s of papers since then. * Differentiable Gaussian rasterization – also quite a shaky definition, which does not give the original paper justice. * The supplied apps both for training and rendering are antiquated by now. Of course, this is bound to happen when new SW is popping up in this field 3x a month. * Representation size and note on compression - also very outdated. Between 50:1 and 200:1 compression ratios are quite standard these days. Note also that VQ is a tiny part of any compression scheme.   So what should we do?   * We believe Gaussian splats will be relevant eventually (much more relevant that Point clouds will ever be) * We are very interested in these formats, in the rendering and production of these formats. * All of this is a highly dynamic environment and agreed common representation formats and beyond this, common compression standards are a long way to go * The dynamic/video aspect around GS is even more challenging. * I like the discussion and conclusions in this paper: https://arxiv.org/html/2405.03417v1. They also include some challenges. * We should provide an introduction to GS in the study * Summarizing the high-level properties and idea * Clearly documenting that this is a highly active research topic * Clearly mention that the documentation is only reflecting the state of the art at the time of writing * Making clear that this has potential to address many of the issues known from other representation formats * Discussing the opportunities and challenges * Recommending that 3GPP follows the research work on GS and awaits stabilization in the industry to commonly agreed formats. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | This document focuses on 3D Gaussian Splatting. Updated based on the comments received from previous meeting. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | An emerging and highly promising representation format is not yet documented. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 2, 4.3 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

## ===== CHANGE ===== (add to References)

[GS-1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Trans. Graph. 42, 4, Article 139 (August 2023), 14 pages. https://doi.org/10.1145/3592433

[GS-2] Wei, Meng et al. “Normal-GS: 3D Gaussian Splatting with Normal-Involved Rendering.” ArXiv abs/2410.20593 (2024): n. pag.

[GS-3] Özyeşil, Onur, et al. "A survey of structure from motion\*." Acta Numerica 26 (2017): 305-364.

[GS-5] T. Wu, Y.-J. Yuan, L.-X. Zhang, J. Yang, Y.-P. Cao, L.-Q. Yan, and L. Gao, “Recent advances in 3d gaussian splatting,” Computational Visual Media, pp. 1–30, 2024.

[GS-6] X. Lei, M. Wang, W. Zhou, and H. Li, “Gaussnav: Gaussian splatting for visual navigation,” arXiv preprint arXiv:2403.11625, 2024.

[GS-7] 3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods, https://3dgs.zip/

[GS-8] Dalal, Anurag et al. “Gaussian Splatting: 3D Reconstruction and Novel View Synthesis: A Review.” IEEE Access 12 (2024): 96797-96820.

[GS-9] Dalal, Anurag & Hagen, Daniel & Robbersmyr, Kjell & Knausgård, Kristian. (2024). Gaussian Splatting: 3D Reconstruction and Novel View Synthesis, a Review. 10.48550/arXiv.2405.03417.

[GS-10] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp and Zan Gojcic; “3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes”, ACM Transactions on Graphics and SIGGRAPH Asia, 2024.

[GS-11] Zhu, Huixin et al. “Scene reconstruction techniques for autonomous driving: a review of 3D Gaussian splatting.” Artif. Intell. Rev. 58 (2024): 30.

## ===== CHANGE =====

### 4.3.X Future formats

Note: formats in that section will not be part of the evaluation framework of release 19, due to their maturity status, or complexity.

#### 4.3.X.1 3D Gaussian Splatting

##### 4.3.X.1.1 Introduction

3D Gaussian Splatting (3DGS) is an emerging method that's gaining attention for its ability to render highly realistic scenes with impressive efficiency and speed. It shows potential in addressing many of the limitations associated with other representation formats. As 3D Gaussian Splatting is a rapidly evolving field, new developments and insights are emerging regularly. This documentation reflects the state of the art at the time of writing and may not capture the most recent advancements. However, a comprehensive, searchable database of 3D Gaussian Splatting papers is available through the following link, which will help you stay updated with the anticipated surge of research in the coming months:

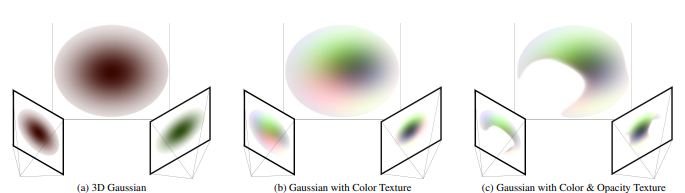
- **Awesome 3D Gaussian Splatting Paper List**: https://mrnerf.github.io/awesome-3D-gaussian-splatting/

##### 4.3.X.1.2 Overview

3D Gaussian Splatting (3DGS) [GS-1], also referred as Gaussian Splatting Radiance Field, is an explicit radiance field based 3D representation that represents 3D scene or objects using a large number of discrete 3D anisotropic balls or particles, each defined by its spatial mean *μ* and covariance matric *∑* [GS-2]:

The covariance matric *∑* is parameterized by using a scaling matrix *S* and a rotation matrix *R,* such that *∑=RSSTRT.* Each 3D Gaussian is associated with a color *c* and an opacity *α*. During rendering, these Gaussians are projected (rasterized) onto the image plane, forming 2D Gaussian splats *G′(x)*. The 2D Gaussian splats are sorted from front to back tile-wisely, and α-blending is performed for each pixel *x* to render its color as follows:

The color of each Gaussian, *c*, is represented by Spherical Harmonics (SH) as to provide view-dependent effects, where *(l,m)* is the degree and order of the SH basis ,  is the corresponding SH coefficient, and specifies the viewing direction.



**Figure 4.3.X.1-1 3D Gaussian Splatting (3DGS) representation**

The data need to perform this 3DGS rendering process are, for each point, known as a Gaussian:

- 3 position values

- 4 rotation values

- 3 color values

- 3 scale values

- 1 transparency value

- 45 spherical harmonics values

Currently, the supported formats for 3D Gaussian Splatting data can be:

- A .PLY file, in which a detailed splat of an outdoor scene might exceed 250 MB (1888 bits by points for float 32 values).

- .splat format, a Javascript types serialized version of .PLY datas.

- The .SPZ format, shrinks the file size from a standard PLY file by using more compressed representation by reduce the different values that are associated with each Gaussian splat. This format reduces the size of the 3DGS files by quantizing the data, reducing the number of spherical harmonics used and compressing the quantized data, using lossless data compression process. This reduction in data in term of quantization and reduction disturbs the 3D object and is not lossless.

The Table 4.3.X.1-1 summarized the parameter types for .PLY and .SPZ format:

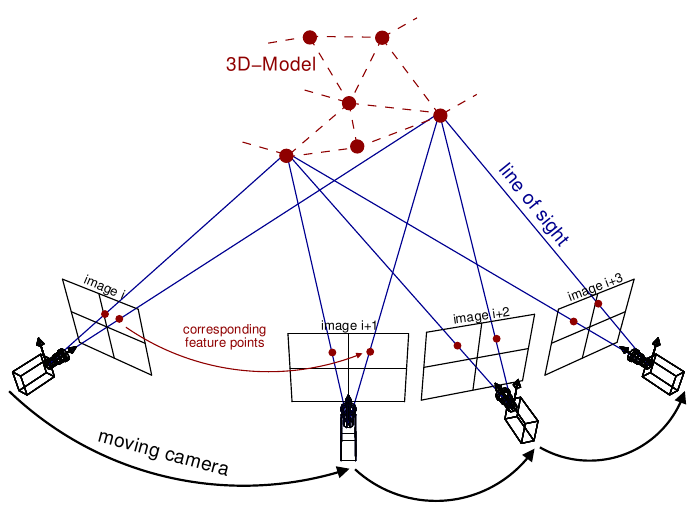
**Table 4.3.X.1-1 Existing 3D Gaussian Splatting (3DGS) representation Formats**

|  |  |  |
| --- | --- | --- |
| **Element** | **SPZ Format** | **PLY Format** |
| Positions | 24-bit fixed point integer with adjustable fractional bits | 32-bit or 64-bit floating-point |
| Rotation | 3 components of a quaternion stored as 8-bit signed integers | 4 components of quaternion as 32-bit floats |
| Color (RGB) | 8-bit unsigned integers per channel | Typically 8-bit or 32-bit floats per channel |
| Scales | 8-bit log-encoded integer | Typically 32-bit or 64-bit floating-point |
| Alphas | 8-bit unsigned integer | Typically 32-bit float |
| Spherical Harmonics | 8-bit signed integers for coefficients, with 4-5 bits of precision | Varies, but usually stored with higher precision (e.g., 32-bit floats) |
| Number of Spherical Harmonics | 0, 9, 24 or 45 values according to compression parameter | 45 values |

The implementation of 3D Gaussian Splatting (3DGS) involves numerous elaborate steps, which may include:

- **Pre-process:**  Estimate 6DOF poses corresponding to each view, and also obtain camera intrinsic parameters. These 6DOF poses (extrinsics), camera intrinsics, as well as RGBs are then processed in SfM block to obtain: (i) 3D point cloud - i.e., 3D projection of salient points observed in multiple images, as well as (ii) refined camera intrinsics and poses, optimized in a large nonlinear solver system.

**- Structure from motion**: This process starts by creating a point cloud from images e.g., using the SFM method [GS-3] with the COLMAP library



**Figure 4.3.x.1-2 Structure from Motion (SfM) photogrammetric principle. Source: Theia-sfm.org (2016).**

- **Convert to gaussian splats**: Each point is then converted to a Gaussian splat, which is described by parameters such as position, covariance, color, and transparency.

- **Differentiable Gaussian rasterization:** Every 2D Gaussian requires differentiable Gaussian rasterization to be projected from the viewpoint of the camera, sorted according to depth, then repeated both backwards and forwards combined for every pixel.

- **Adaptive Density Control:** The method for dynamically adjusting the number, density, and parameters of Gaussians to accurately and efficiently represent the 3D scene. This involves two steps:

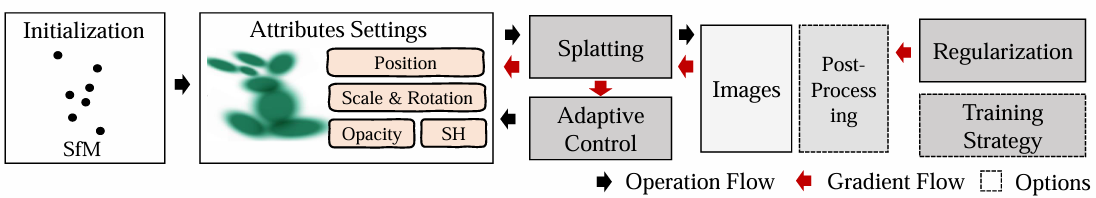
- **Pruning:** If the opacity of is too small or the gaussian is too large, then it’s being removed.

- **Densification:** this step handles two issues:

- Over-reconstruction: regions of a 3D scene are represented by excessively large or overlapping gaussians, leading to redundant and inefficient coverage of the geometry. To solve this the large gaussian is split in two parts (bottom row).

- Under-reconstruction: regions of a 3D scene lack sufficient Gaussian coverage, resulting in missing or poorly represented geometric details. This is solved by merging/cloning two or more gaussians associated with the area (top row).

- **Training**: iterative calculations on the content itself are used to determine additional details about how the point could be stretched/scaled (covariance) and its opacity using Stochastic Gradient Descent (SGD). Through this process, a model is created with millions of points containing data such as position, color, covariance, and opacity. The inputs to 3DGS training are: (1) Refined 6DoF poses (camera extrinsics), (2) Refined (or original) camera intrinsics - a.k.a., projection matrices, (3) Images/RGBs corresponding to these poses and intrinsics, and (4) 3D points cloud - obtained using SfM.



**Figure 4.3.X.1-3 Pipeline and Related Technologies of 3D Gaussian Splatting**

##### 4.3.X.1.3 Production and Capturing Systems

The formats as defined in clause 4.3.X.1 may be captured by mobile devices with several mobile apps (both Android and iOS devices) or online services utilize 3DGS technology. However, smartphones often apply automatic enhancement to each captured images, aiming for the best result. These optimizations, such as sharpening edges, adjusting exposure, and optimizing colors, can introduce noise or errors that are not suitable for Gaussian Splatting training. To mitigate these challenges, a photogrammetry capture method with professional-grade cameras (e.g., Interchangeable Lens Camera), standardized settings, and a streamlined capture process can be utilized to meet the requirements.

The New York Times (NYT) development team explored the practical applications of 3D Gaussian Splatting (3DGS) for spatial journalism. They tested a range of capture and processing techniques using various hardware and software, evaluating solutions for both desktop and mobile devices, and give an overview of the practical takeaways they learned exploring exploring gaussian splatting for spatial journalism [GS-4].

- Quality: High resolution RAW stills when photographing a stationary object or space that requires lots of detail. More resolution generally equates to a more detailed splat — up to a point. They tested images up to 45 megapixels and video up to 8K and found little increase in quality above 20 megapixels or 6K video. If you opt for extreme resolutions, you may need to rescale your images or video before processing.

- Speed: Using a high frame rate during video capture (e.g., 120fps vs 24fps) can ensure less subject movement between frames, more overlap between adjacent frames, decrease motion blur, and help guarantee sufficient coverage. Using a burst mode or automated continuous capture mode when capturing still images similarly speeds up the process.

##### 4.3.X.1.4 Rendering and Display Systems

During the rendering process, each Gaussian is rasterized onto the screen according to its parameters. Alpha blending technics are used to smoothly blend the transparenced splats to create a continuous surface appearance.

In the first step of the rendering process, the 3D points are sorted according to the viewer's position to be projected in the correct order on the screen and allow for proper color mixing. Rasterization of Gaussian splats works for each point by projecting the ellypsoid made based on the covariance matrix on the screen and for each pixel covered by the current splat, the color is calculated using the user's position and the values of the spherical harmonics. Alpha blending adds the colors of all the splats covering the current pixels to obtain the final color and achieve high-quality rendering.

Various implementations of the rendering process have been proposed in the literature. The standard implementation of the rasterization process can be made on CPU or on GPU using rendering shaders or compute shaders with various 3D graphics API: OpenGL, Vulkan, DirectX, CUDA, etc…

To facilitate high-frame-rate and high-resolution differentiable rendering, a tile-based rasterization process has been proposed in [GS-5]. For example, the rasterizer divides the image into a set number of tiles, assigning an index to each tile. For each Gaussian primitive, the rasterizer determines which tiles the primitive’s projection intersects and generates a key-value pair for each intersecting tile. By constructing these key-value pairs, the rasterizer only needs to perform a global sort on all pairs, eliminating the need for additional sorting of primitives for each pixel.

After sorting, the key-value pairs derived from each tile are stored in contiguous memory intervals. The rendering process for each tile is then managed by a CUDA thread block, with the number of threads within each block matching the number of pixels in the tile. Each thread is responsible for the alpha blending process for its corresponding pixel, completing the final rendering.

To improve the quality of the rendered sceen, [GS-10] proposes to render a 3D Gaussian splat scene with the ray tracing process. This method improves the quality of rendering by allowing for precise soft shadows, reflections, and transparency, surpassing the previously mentioned rasterization processes in terms of realism, but greatly increasing complexity.

##### 4.3.X.1.5 Supporting Information

- Typical quality criteria for evaluating the format

- In the image domain: The Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM),Learned Perceptual Image Patch Similarity (LPIPS), the resultant size in megabytes (MB), the training time, and required storage.

- In the 3D point domain: point to point/ point to plane, / point to surface for position. Attribute PSNR.

- Conversion from other formats (lossless, lossy)

- Point Clouds: There are tools, e.g., 3D Gaussian Splatting Converter (https://github.com/francescofugazzi/3dgsconverter), can seamlessly convert 3DGS .ply files to a Cloud Compare-friendly format and vice-versa. Converting point cloud representations to 3DGS can effectively fill in the point clouds’ holes, which is typically done after high-precision reconstruction of the point clouds. Conversely, 3DGS can also be converted into point clouds, followed by voxelizing the point cloud into 3D voxels and then projecting them onto 2D BEV grids [GS-6].

NOTE: filling holes in point cloud, can also be achieved by different techniques.

- Mesh: A considerable amount of work has discussed how to convert 3DGS to Mesh. Once converted, the quality of mesh can be further optimized to achieve better geometric and appearance,

- Uncompressed data size

A Gaussian splat of a scene is a representation of 3D points. On average, a splat contains between 0.5 and 5 million of these 3D points. Each 3D point has unique parameters that represent the scene as accurately as possible.

- Known compression technologies:

Early research on 3DGS compression is ongoing. Vector quantization (VQ) [GS-7] had been applied in some research for 3DGS compression and encoding.Between 50:1 and 200:1 compression ratios are quite standard these days. MPEG has started an exploration that is looking at the most appropriate representation formats for 3DGS and various coding strategies.

- Extensibility of the format

The format is undergoing massive academic and industrial research, can be further expanded for more capabilities and is hence not stable. Example of use cases include: 3DGS can be further expanded for more capabilities, including dynamic 3DGS [GS-5, GS-8], surface representation from 3DGS [GS-5, GS-8], editable 3DGS [GS-5, GS-8], 3DGS with semantic understanding [GS-5, GS-8], and 3DGS-based physics simulation [GS-5, GS-8].

##### 4.3.X.1.6 Benefits and Limitations

###### 4.3.X.1.6.1 Benefits

- Real-time Rendering with GPU acceleration.

- Accurate Reconstruction, it can capture the geometry accurately

- Explicit representation

- Ability to render complex scenes in real-time

- Interpretability of the representation, an explanation of the mathematical mechanism, i.e., the working principle, of 3DGS can help researchers analyze the complex relationships in 3D scene reconstruction technology and reveal the performance characteristics of 3DGS in depth. [GS-11]

- Gaussian Splatting can deliver high-quality, real-time visualizations.[GS-9].

- Gaussian Splatting has evolved to handle dynamic and deformable objects [GS-9].

- Gaussian Splatting can be applicable to various application space, such as digital avatars and SLAM [GS-9].

###### 4.3.X.1.6.2 Limitations

- There is a lack of industry agreement on the 3DGS format(s), due to no stable representation and compression format exists for static and dynamic 3DGS.

- Static and Dynamic 3DGS formats is evolving, multiple options are considered in current academic and industrial research. For dynamic, such research include modeling in 4 dimensions (i.e. temporal), time evolving 3DGS, and MLP predicted motion for 3DGS among others.

- High memory usage

- Not yet fully compatible with existing rendering pipelines

- Computation complexity [GS-9], the computational demands of handling large numbers of splats, especially for high-resolution rendering and complex scenes; it requires to process large datasets for training, which can also be time-consuming and resource-intensive.

- Edge artifacts [GS-9].