This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

GaussianUpdate: Continual 3D Gaussian Splatting Update for Changing Environments

Lin Zeng* Boming Zhao* Jiarui Hu Xujie Shen
Ziqiang Dang Hujun Bao Zhaopeng Cui†
State Key Lab of CAD & CG, Zhejiang University



Figure 1. **GaussianUpdate**. We present GaussianUpdate, a novel framework that can update the existing Gaussian fields by generatively replaying past data and currently captured data. The updated Gaussian fields can synthesize novel views across different times, capturing geometric layout changes between consecutive timestamps.

Abstract

Novel view synthesis with neural models has advanced rapidly in recent years, yet adapting these models to scene changes remains an open problem. Existing methods are either labor-intensive, requiring extensive model retraining, or fail to capture detailed types of changes over time. In this paper, we present GaussianUpdate, a novel approach that combines 3D Gaussian representation with continual learning to address these challenges. Our method effectively updates the Gaussian radiance fields with current data while preserving information from past scenes. Unlike existing methods, GaussianUpdate explicitly models different types of changes through a novel multi-stage update strategy. Additionally, we introduce a visibility-aware continual learning approach with generative replay, enabling self-aware updating without the need to store images. The experiments on the benchmark dataset demonstrate our method achieves superior and real-time rendering with the capability of visualizing changes over different times. Please refer to our project webpage for more informations: https://zju3dv.github.io/GaussianUpdate.

*Equal contribution.
†Corresponding author.

1. Introduction

High-quality scene reconstruction and realistic image rendering are crucial for many applications, such as AR/VR, 3D gaming, and robotics. In recent years, neural scene representations, represented by Neural Radiance Fields (NeRF) [22], have demonstrated outstanding performance for novel view synthesis. While these rendering results are impressive, the same scene often undergoes many changes over time. For example, the same room will have different objects, different furniture placements, and different lighting conditions at different times. Therefore, updating the original neural model is crucial to render images that align with the current appearance of the scene. Additionally, the knowledge from previous moments is valuable, and we hope to retain information from various times after updating the original 3D model. This would enable high-quality images of the same scene to be rendered at different moments when needed.

A simple approach is to retrain a neural model with the images from the current moment. However, preserving neural models from each moment would increasingly consume storage space. Another strategy is to directly update the neural model with data from the current time, but this may lead to catastrophic forgetting [21], causing the neural model to lose appearance and geometric information from

previous times and be unable to accurately render scenes from earlier times. Existing methods [24, 30, 31, 33, 34] for 4D modeling can represent continuous dynamics through dynamic fields, but they are not directly applicable to scenarios with abrupt scene changes, such as the sudden appearance or disappearance of objects.

Several attempts [2, 32] have been made to integrate continual learning strategies [5] into the learning of NeRF models. For instance, CLNeRF [2] combines continual learning with NeRF, updating the NeRF model with new data while preserving the geometric and appearance information from previous moments. However, due to the inherent drawbacks, NeRF-based methods fail to locate the changes over time and suffer from low rendering efficiency. Recently, 3D Gaussian Splatting [12] introduced a tile-based splatting rendering framework that uses 3D Gaussians to represent the scene. This explicit representation simplifies modeling and detection of fine-grained geometric and photometric changes, while enabling real-time rendering, offering a promising solution to the challenges mentioned earlier.

However, designing a system that combines 3D Gaussian representation with continual learning is far from straightforward and presents substantial challenges. Firstly, a temporal model is needed for the 3D Gaussian representation to address the differences in the appearance of the scene at different moments in time. Furthermore, when objects that originally existed in the scene are no longer present at the current moment, or new objects have appeared, we need a strategy to explicitly update the 3D Gaussian model. This process entails the precise removal of the 3D Gaussians associated with the objects that have disappeared and the addition of new 3D Gaussians at the necessary locations. The most relevant work is dynamic 3D Gaussian modeling [16, 31, 33]. These methods use an additional network to learn the deformation fields at different moments, warping the attributes of each 3D Gaussian from the canonical space to render temporally varying scene images. However, these methods assume that scene changes are caused by the continuous motion of objects and cannot model discrete changes in our tasks such as the appearance or disappearance of objects.

In this paper, to tackle these challenges, we introduce GaussianUpdate, a novel method that combines 3D Gaussian representation with continual learning for the first time. Our method effectively updates the previous neural model with current data while preserving as much information as possible from all past scenes. Compared to past moments, the changes in the current 3D scene are explicitly modeled as three types: lighting changes, disappearance of existing objects, and the addition of new objects. To model these discrete changes, we first propose a global appearance updating method for 3D Gaussians that utilizes hash encoding to store temporal information, thereby fitting the lighting variations at different moments. Additionally, we developed a 3D change detection method to identify and remove the 3D Gaussians corresponding to objects that have disappeared at the current moment. We also designed a simple but efficient COLMAP-based point addition strategy to represent newly added objects. With these meticulous designs, the proposed GaussianUpdate can ensure real-time photorealistic rendering of the scene's appearance at different times while also supporting the visualization of detailed changes over time.

Our contributions can be summarized as follows:

- We present GaussianUpdate, a novel framework that innovatively integrates 3D Gaussian representation with continual learning, leveraging images of the current scene to update the previous model and render novel view images at different moments.
- We develop a novel multi-stage update strategy including hash encoding attribute updating, 3D change detection for object removal, and COLMAP-based point addition, which can adapt the previous 3D Gaussian model to the current changes in real-world scenes.
- We present the visibility-aware continual learning with generative replay which enables self-aware updating to various types of changes without the need to store additional images.
- Experimental results demonstrate our method achieves superior rendering quality compared to existing methods while ensuring real-time rendering and visualization of changes over different times.

2. Related Work

NeRF and 3D Gaussian Splatting. Neural Radiance Field(NeRF) possesses robust multi-view synthesis capabilities. The vanilla NeRF [22] uses multi-layer perceptrons (MLPs) to map the input 3D coordinates and view direction into different colors and opacities and then employs volume rendering [11] to produce images from novel views of the scene. However, NeRF can only reconstruct static scenes [9]. To capture the dynamic changes in a scene, dynamic NeRF [3, 7, 8, 24] trains a deformable field that deforms information from the canonical space to match different scene changes over time. NeRF-w [20] and its follow up works [1, 25] aiming to reconstruct from an internet collection dataset for outdoor benchmarks, introduces transient encoding and appearance encoding to capture variations in lighting and geometry at different times, thereby distilling the structural essence of the scene. However, the primary goal of NeRF-w's transient encoding is to filter out transient objects that occlude the main structure at different times. In contrast, our goal is to capture the changes at each moment, thereby rendering images for each specific instance.

Another major drawback of NeRF is its slow inference speed. Recently, 3D Gaussian Splatting [12] employs a significant number of explicit 3D Gaussians to rep-

resent a static 3D scene [10] and achieves real-time inference through tile-based splatting rendering. Many methods [17, 30, 31, 33, 34] have improved upon the dynamic NeRF based on 3D Gaussian, thus achieving higher rendering quality and inference speed. However, to the best of our knowledge, there are currently no methods based on 3D Gaussians that can solve our task.

Continual Learning. Continual learning refers to models continuing to learn from new data after completing initial training, in order to adapt to changes in the environment. However, naive training on new data can lead to catastrophic forgetting of the knowledge previously learned from historical data. One strategy involves incorporating regularization terms that are unrelated to historical data [14, 15], yet this method frequently exhibits subpar performance in practical applications. Alternatively, some methodologies mitigate forgetting by replaying historical data, such as memory replay [4, 18], which stores historical data in a dedicated memory pool, and generative replay [29], which utilizes a generative model to replay historical data. Moreover, certain techniques prevent forgetting by selectively freezing portions of the network's model while introducing new network modules [19, 28]. [2] and [32] integrate continual learning into NeRF. [2] achieves continual learning for NeRF in real-world scene changes by replaying historical data. In contrast, [32] attains higher training efficiency with a small number of images through a lightweight expert adaptor and a conflict-aware knowledge distillation strategy. We introduce continual learning to 3D Gaussian Splatting. Compared to NeRF-based continual learning methods, our approach achieves faster inference speeds and higher quality in novel view synthesis.

3. Method

Given a trained 3D Gaussian field, along with time-varying scene images and corresponding poses, our method can continuously update and record changes in this scene. In Sec. 3.1, we briefly introduce rasterization principles and the task formulation of continual learning in a 3D Gaussian field. We developed a three-stage scheme to progressively update the scene representation, avoiding interference from the tight coupling of appearance and geometry properties(Sec. 3.2). Moreover, in Sec. 3.3, we elaborate on our visibility-aware learning technique with generative replay for self-aware scene updating.

3.1. Preliminaries

Rasterization Principles. We defines a 3D scene as a set of 3D Gaussian primitives with 3D covariance $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$, locations $\mu \in \mathbb{R}^3$, opacity σ, and spherical harmonics coefficients $\mathbf{Y}$:

$$\mathbf{G} = \{G_i : (\mu_i, \Sigma_i, \sigma_i, Y_i) \mid i = 1, ..., N\}. \tag{1}$$

It is worth noting that, to ensure positive semi-definite in the gradient descent, $\mathbf{\Sigma}$ is formulated using a scaling matrix $\mathbf{S} = diag([s])$ and rotation Matrix $\mathbf{R}$:

$$\Sigma = RSS^TR^T. \tag{2}$$

Following EWA Splatting [35], given a world-to-camera rotation $\mathbf{W}$ and the Jacobian $\mathbf{J}$ of the affine approximation of the projective transformation, 3D Gaussians $\{\mathbf{\Sigma}, \mu\}$ can be projected to 2D distributions $\{\mathbf{\Sigma}', \mu'\}$ as:

$$\Sigma' = JW\Sigma W^TJ^T. \tag{3}$$

μ' indicates the pixel coordinates of Gaussian center points. For view synthesis, the rasterizer sorts all Gaussians in a front-to-back order and performs α-blending rendering as follows:

$$\alpha_i = \sigma_i e^{-\frac{1}{2}(x-\mu')^T\Sigma'^{-1}(x-\mu')}, \tag{4}$$

$$T_i = \prod_{j=1}^{i-1}(1-\alpha_j), \tag{5}$$

$$C = \sum_{i\in N} c_i\alpha_i T_i, \tag{6}$$

where N is the number of 3D Gaussian primitives, c_i is the color of Gaussians obtained from spherical harmonics coefficients $\mathbf{Y}$, α_i is the opacity contributed to a pixel, and T_i is the is the accumulated transmittance.

Task Formulation. In the context of a time-varying scene and Gaussian representation, we can define the continual learning task in a 3D Gaussian field as follows:

1. At newly arrived time step t, a set of multi-view images $\mathcal{I}_{gt} = \left\{I_f^t \mid f = 1 \dots N\right\}$ of the changed scene, along with camera poses, are collected.
2. Based on these latest observations, while preserving prior scene knowledge, an updated 3D Gaussian field can be obtained from the one-at-time step t-1, enabling continual updating and scene recall at different times.
3. A flexible and continual 3D Gaussian field is built up to achieve photo-realistic novel view synthesis and support the visualization of changes in a time-varying scene.

Unlike short-term continuous dynamics [24, 30, 31, 33, 34], our goal is to follow temporally arbitrary changes in a scene over a long term, which requires adaptive and self-aware Gaussian management in 3D Gaussian fields. As shown in Figure 2, a modularized approach has been proposed to effectively handle this task, and we will detail different modules below.

3.2. Model Update

In order to temporally model scene changes, we additionally introduce an efficient hash grid into our 3D Gaussian field as a global appearance model, generating a hybrid scene representation. In the real world, environment

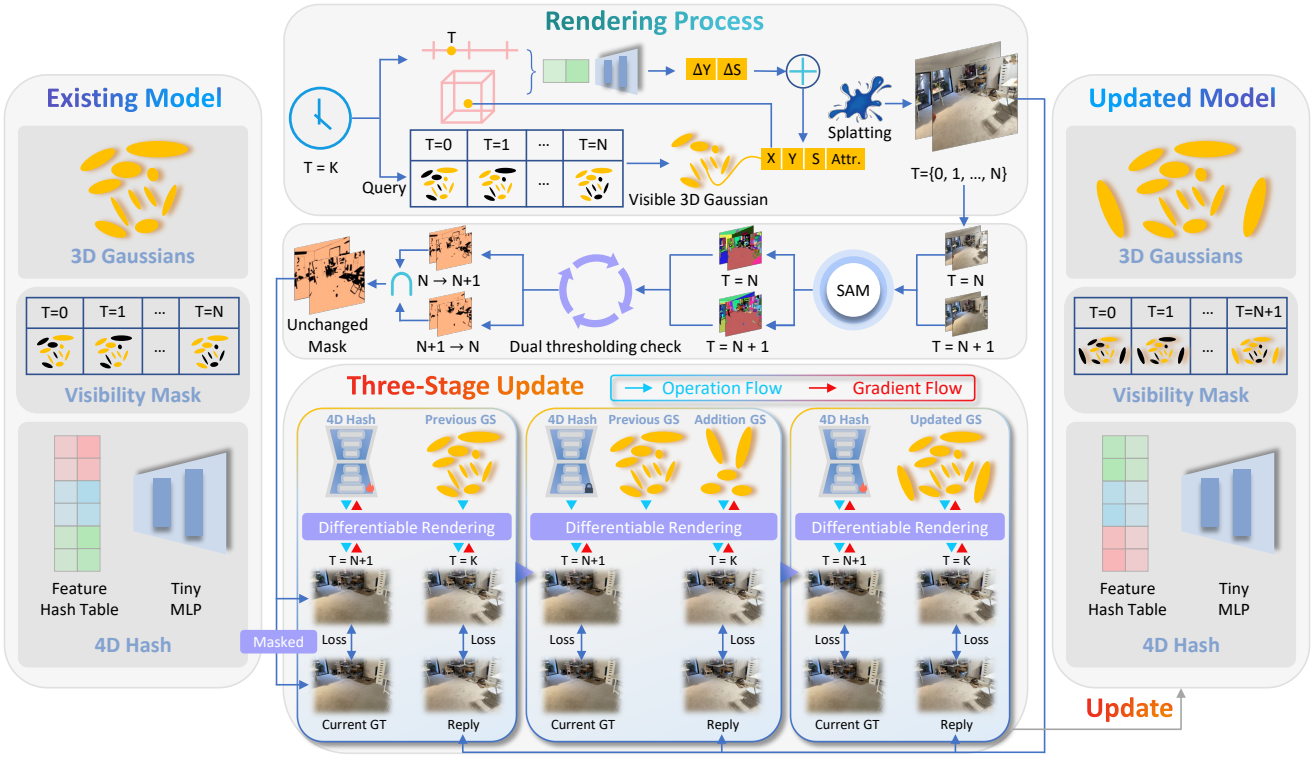


Figure 2. **System overview**. Given an existing 3D Gaussian Field, we use a visibility pool to record changes in scene layout and a 4D Hash as an appearance model to learn global illumination changes in the scene. The model update can be divided into three stages. In the first stage, we learn the global illumination changes in the layout-invariant regions. In the second stage, we learn the geometric layout changes of the scene while fixing the appearance model and growing sparse Gaussian primitives from COLMAP. Finally, we refine both the appearance model and parameters of added Gaussians.

changes mainly occur in two aspects: global illumination and scene layout. Straightforward optimization for both of them, especially in a α-blending method, will cause serious ambiguity and overfitting problems. Therefore, we design a progressive solution to decouple lighting and layout parts to achieve accurate and fine-grained scene updates.

3.2.1. First Stage: Global Appearance Update

Global Appearance Model. As shown in Figure 2, to model the global appearance changes caused by lighting, a 4D hash grid $\mathbf{H}$ and a tiny MLP $\mathbf{F}$ are used to serve as a global appearance model to infer incremental scaling and spherical harmonics properties $\{\Delta s_i^t, \Delta Y_i^t\}$ on Gaussian primitives at different times:

$$\{\Delta s_i^t, \Delta Y_i^t\} = F(H\{\mu_i, t\}). \tag{7}$$

Appearance Updates in Layout-invariant Areas. In general, global illumination dominates appearance changes in layout-invariant regions (i.e., the static regions without any geometric changes). In the first stage, our approach focuses on updating such changes by separately optimizing the 4D hash grid. In order to avoid the influence caused by the possible new or removing objects in the scene, we design a robust method to determine the layout-invariant areas. Specifically, suppose I_f^t is the newly captured image, we can render I_f^{t-1} with the camera pose of I_f^t and the previous neural model. Then for this pair of corresponding frames $\{I_f^t, I_f^{t-1}\}$, we can acquire their instance segmentation masks $\{S_f^t, S_f^{t-1}\}$ from a foundation model SAM [13]. Furthermore, we can calculate the Intersection over Union (IoU) score between corresponding instances from $\{S_f^t, S_f^{t-1}\}$. Layout-invariant areas with high IoU scores at both time steps t and t-1 will be masked into $\{M_f^t, M_f^{t-1}\}$. At the current time step t, a final layout-invariant mask M_f comes from the intersection of M_f^t and M_f^{t-1}, and it will be used to guide subsequent optimization of the appearance model. Combined with this mask M_f and photometric loss (Eq. 8-10), the hash-encoding appearance model can be accurately optimized in layout-invariant areas.

$$\mathcal{L}_{st} = (1-\lambda)\mathcal{L}_1 \cdot \mathcal{M}_f + \lambda \mathcal{L}_{D-SSIM} \cdot \mathcal{M}_f, \tag{8}$$

$$\mathcal{L}_1 = \frac{1}{HW} \sum_{n=1}^{HW} |C_n - \hat{C}_n|, \tag{9}$$

$$\mathcal{L}_{D-SSIM} = SSIM(C, \hat{C}). \tag{10}$$

By excluding all possible dynamic objects or parts, we can learn the global appearance model better, which further benefits the following geometric layout update.

3.2.2. Second Stage: Geometric Layout Update

In areas where the geometric layout changes, it is necessary to prioritize growing and pruning Gaussian primitives to fit the correct geometry shape and avoid unexpected overfitting. For emerging objects, new Gaussian primitives G_{add} are spawned from the COLMAP and added to the existing Gaussian field. For missing objects, we set learnable removal factors m which are activated by a function ψ as in Eq. 12. Notably, $\psi(m)$ is only multiplied by the opacity of previous Gaussians G_p.

At this stage, the removal factors m and properties of G_{add} are optimized as in Eq. 11 while keeping the global appearance model learned in the previous stage fixed. The regularization term is defined in Eq. 13 to drive $\psi(m)$ towards 0 or 1.

$$\mathcal{L}_{nd} = (1-\lambda_1)\mathcal{L}_1 + \lambda_1\mathcal{L}_{D-SSIM} + \mathcal{L}_{reg}, \tag{11}$$

$$\psi(m) = \frac{1}{1+e^{-1000m}}, \tag{12}$$

$$\mathcal{L}_{reg} = \lambda_2(1-\psi(m))\psi(m) + \lambda_3 BCE(\psi(m), 1). \tag{13}$$

Ultimately, we set a threshold $\tau = 0.01$ to identify candidate Gaussian primitives G_c ($\psi(m) < \tau$) to be removed. Moreover, we further employ the DBSCAN [6] algorithm to filter out sparse outliers in G_c which are far from surfaces of missing objects, and the remaining Gaussians will be clustered into a bounding box where any Gaussians are pruned from G_p. Different from the implicit neural field, our explicit continual learning method allows pruned Gaussians to be stored in a visibility pool (Sec. 3.2.3) for visualizing changes and recalling previous scene models.

3.2.3. Third Stage: Joint Refinement

After pruning Gaussian primitives, rendering artifacts inevitably appear around removed objects. Thus, in the third stage, we perform a joint refinement, following the photometric loss in Eq. 14, to optimize the appearance model and properties of G_{add} together.

$$\mathcal{L}_{rd} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}. \tag{14}$$

Importance Purning. Inspired by [23], we expect to extract valuable Gaussians to reduce memory consumption. By traversing all training views at the current time step, we can calculate an importance score v_i of the i-th Gaussian as follows:

$$v_i = \max_{n \in N_1}(\alpha_i^n T_i^n), \tag{15}$$

where N_1 represents the set of pixels related to the i-th Gaussian in all our training views. The Gaussians with importance scores less than 0.05 will also be discarded.

3.3. Visibility-aware Continual Learning

A continual 3D Gaussian field can not only solve the catastrophic forgetting problem over the long term but also recall the scene representation at any previous time step. To this end, we maintain a visibility pool P_v for each Gaussian in the entire time period. Additionally, we store the camera extrinsic of historical training views for generative play [29].

Visibility Pool. The explicit scene representation clearly indicates Gaussian primitives on which scene changes occur, which is a significant advantage over existing implicit works, especially in continual scene updates. In our method, we set a visibility pool to store all involved Gaussians over time. Pruned Gaussians will actually be set to inactive, and vice versa. This novel visibility-aware strategy allows our method to perform primitive-level control over the scene map, and it also serves as the foundation for visualizing scene changes.

Generative Replay. Benefiting from efficient and photorealistic view synthesis, Gaussian-based scene representation is naturally suitable for generative replay [29] in continual learning, which only requires recording a small number of camera extrinsic. Specifically, at each new time step, based on previous camera poses and trained 3D Gaussian fields, we can re-render past views as historical cues in our training dataset to mitigate the forgetting problem.

4. Experiments

4.1. Dataset and Implementation details

Dataset. We evaluate GaussianUpdate on the World Across Time (WAT) dataset [2], which comprises 10 real scenes encompassing both indoor and outdoor environments. Additionally, we conducted experiments on the Synthetic NeRF dataset [22], which comprises 8 distinct scenes.

We adopted the Synthetic NeRF dataset partitioning strategy from CLNeRF [2], dividing the training images of each scene into 10 sequential, equally-sized subsets. During training, these subsets were used sequentially as ground truth images. Although the scenes are static, the varying viewpoints change across subsets. To prevent forgetting, we employed a generative replay approach to generate historical images for supervision, utilizing saved camera poses. Since each scene in the Synthetic NeRF dataset theoretically remains static, we followed CLNeRF [2] by omitting appearance embeddings, as well as the appearance model and visibility pool for this dataset.

Implementation details. Our approach updates a given 3D Gaussian field using a three-stage training strategy. Initially, we initialize the 3D Gaussian field by leveraging the dense points from MVS [27] to achieve a more refined Gaussian field initialization. In the first stage of updating the Gaussian field, we conduct 7k iterations of training, focusing solely on the global appearance model, while the densify

Method	Breville		Kitchen		Living room		Community		Spa	
	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)
Baseline	20.66	0.745	16.44	0.610	17.28	0.720	11.13	0.448	15.38	0.580
CLNeRF [2]	28.02	0.826	28.40	0.877	24.58	0.829	22.88	0.629	26.28	0.811
4DGS [31]	28.92	0.908	27.03	0.900	24.41	0.873	22.99	0.711	26.04	0.859
Ours	30.11	0.927	28.02	0.912	26.10	0.881	23.88	0.764	27.84	0.891
UB	30.36	0.929	27.99	0.911	26.22	0.882	23.88	0.755	28.16	0.894

Method	Street		Car		Grill		Mac		Ninja	
	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)
Baseline	13.12	0.486	19.53	0.610	19.01	0.575	17.80	0.778	18.97	0.777
CLNeRF [2]	22.53	0.612	22.73	0.541	24.48	0.653	29.33	0.906	27.19	0.878
4DGS [31]	22.26	0.680	23.13	0.662	24.94	0.733	29.65	0.930	27.64	0.912
Ours	22.80	0.703	23.81	0.742	25.31	0.770	29.92	0.934	27.75	0.921
UB	23.23	0.715	23.77	0.741	25.46	0.774	30.17	0.935	27.98	0.921

Table 1. **Quantitative comparison on WAT dataset**. Best results are highlighted as **first** , **second** .



(a) w/o Layout-invariant Mask. (b) w/o Addition . (c) w/o Removal Factors. (d) Full Model.

Figure 3. **Effectiveness analysis of each component**.

operation [12] is closed. After the first phase of training, we add the sparse points obtained by SFM [26] as the initialization Gaussians of the new object and open the densify operation. A total of 8k iterations are trained in the second stage. We will use DBSCAN to filter the points to be deleted when 5k iterations and 8k iterations are trained in the second stage. In the third stage, we conduct a total of 15k training iterations and employ importance pruning to eliminate excess Gaussians in G_e after completing 4k iterations of training during this stage. Furthermore, we normalize the time steps t to the $[0, 1]$ for each scene. All experiments are conducted and evaluated on an NVIDIA GeForce RTX 3090 GPU with 24 GB of RAM.

4.2. Evaluation

To comprehensive evaluation, we compare GaussianUpdate to various methods for changing environments. (1) Baseline: the baseline model refers to 3D Gaussian Splatting [12] combined with our proposed appearance model and visibility pool, without using generative replay data. The baseline represents the lower bound performance of our model on continuous updates. (2) CLNeRF [2]: the SOTA method for continual learning with NeRF; * (3) 4DGS [31]: 4DGS extends 3DGS to handle dynamic scenes by incorporating temporal dynamics into the differentiable Gaussian rendering process. This approach predicts a 4D joint distribution that captures the time-variant behavior of Gaussian primitives; Additionally, 4DGS necessitates inputting data from all time points simultaneously to learn the 4D Gaussian distribution effectively. (4) Upper bound model (UB): we update the 3D Gaussian field by replacing generative replay with memory reply [4, 18] (data memory pool stores all past GT images) to represent the upper bound our method can achieve.

As shown in Table 1 and Table 2, our method generally outperforms CLNeRF across most scenes, except for the kitchen scene in the WAT dataset where PSNR is slightly lower. Notably, our method consistently achieves significantly higher SSIM scores across all scenes, suggesting that our rendered images more closely resemble the ground truth images in the test set. Moreover, leveraging the inference speed of 3DGS, our method achieves real-time rendering speeds of 34 FPS on the WAT dataset, while CLNeRF lags significantly behind at 0.75 FPS. Furthermore, our method consistently outperforms 4DGS across all scenes. Notably, since the Synthetic NeRF dataset represents a static scene, the time input for all images is set to 1 when training 4DGS on this dataset. In comparison to UB, which utilizes memory replay, our method achieves comparable rendering quality with lower storage consumption, as it does not necessitate storing historical images.

The visualization results in Figure 4 demonstrate that our method outperforms CLNeRF in handling details. Baseline overfits the current data, resulting in catastrophic forgetting of previous images. Since each scene in the WAT dataset

*CL-NeRF [32] is also a NeRF-based continual learning method, while the code is not publicly available at the time of our submission.

Method	Lego		Chair		Drums		Ficus		Hotdog		Materials		Mic		Ship	
	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)	PSNR(↑)	SSIM(↑)
Baseline	17.98	0.738	23.35	0.889	18.75	0.784	26.44	0.928	28.54	0.926	17.44	0.738	27.17	0.935	18.56	0.583
CLNeRF [2]	34.80	0.976	34.39	0.980	25.46	0.931	33.10	0.980	36.42	0.979	29.12	0.943	34.68	0.987	29.30	0.880
4DGS [31]	33.23	0.968	34.35	0.976	25.86	0.947	34.77	0.985	36.16	0.977	29.16	0.952	34.44	0.988	29.48	0.886
Ours	35.70	0.980	35.21	0.986	26.07	0.951	34.98	0.985	37.59	0.983	30.205	0.958	36.33	0.991	31.47	0.903
UB	36.27	0.982	35.74	0.987	26.32	0.954	35.52	0.987	38.12	0.985	30.52	0.960	36.68	0.992	31.79	0.906

Table 2. **Quantitative comparison on Synthetic NeRF dataset**. Best results are highlighted as **first**, **second**.

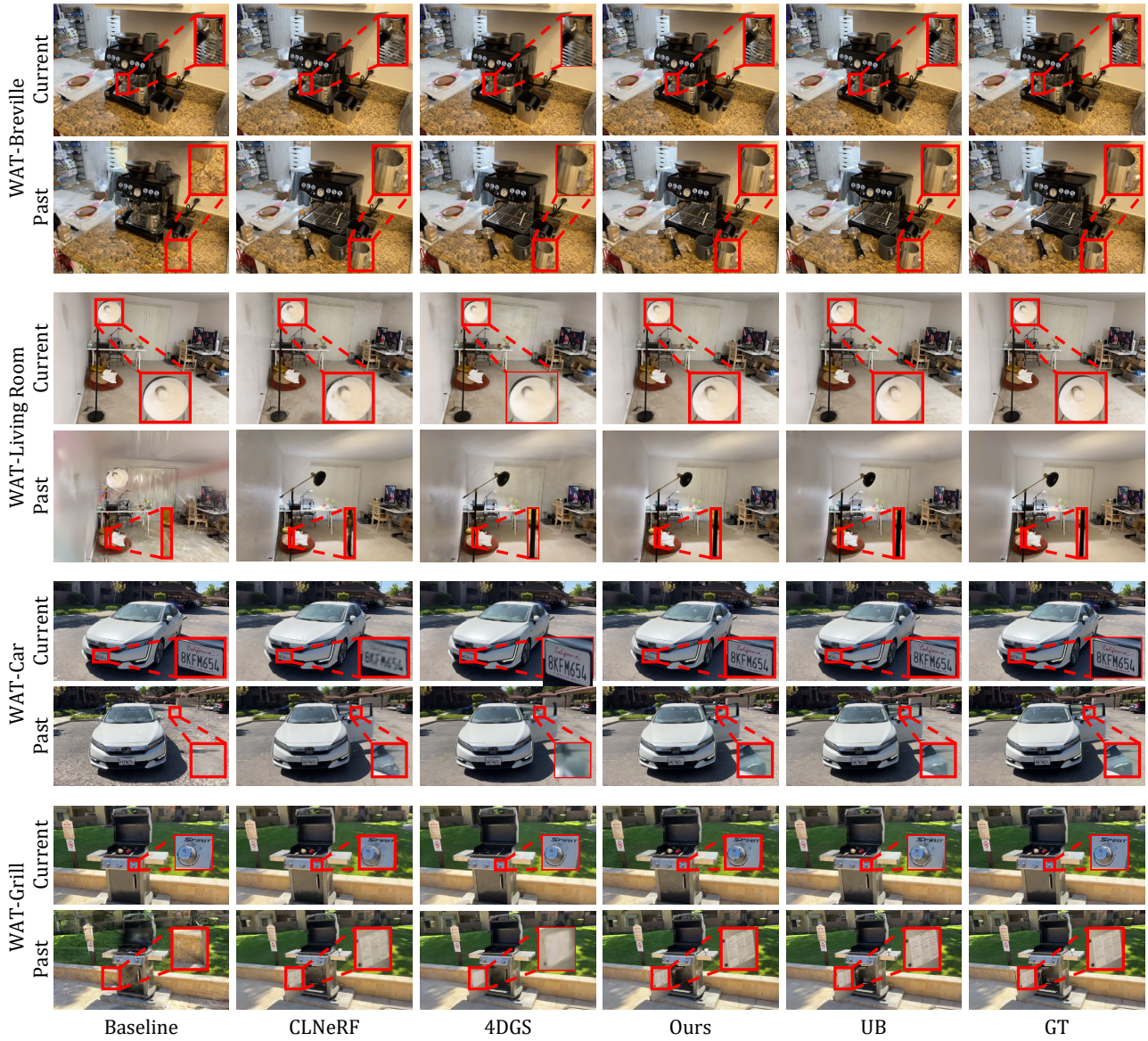
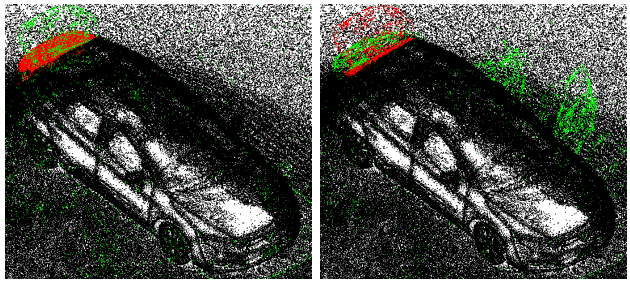
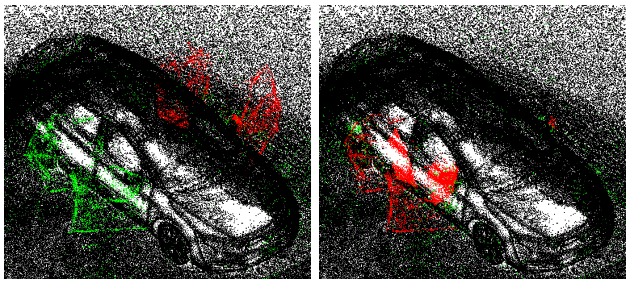


Figure 4. **Visualizaton results on WAT dataset**. Each pair of lines shows the current and past test images rendered by different methods. NT overfitted the current data, leading to forgetting past knowledge. Our method demonstrates better rendering results compared to CLNeRF and 4DGS.

undergoes significant changes over time, 4DGS is unable to accurately reconstruct the scene at each moment, even when data from all time points is provided during training. Table 4 illustrates the average PSNR of our method across sequential updates from Time1 to Time4 on the WAT dataset (detailed data for each scene is available in the supplementary



(a) $\mathbf{T}_1$ (b) $\mathbf{T}_2$



(c) $\mathbf{T}_3$ (d) $\mathbf{T}_4$

Figure 5. **The visualization results of the scene geometric changes learned by ours in scene car at each time**. The green points represent the objects that have been added compared to the previous time, while the red points indicate the objects that disappeared compared to the previous time.

Config.	WAT	
	PSNR ↑	SSIM ↑
w/o Removal Factors	25.87	0.834
w/o Addition of 3D Gaussians	25.96	0.829
w/o Layout-Invariant Mask	26.18	0.839
Full Model	26.55	0.844

Table 3. **Ablation study of each step discussed in Sec. 3.2 on WAT dataset.** Best results are highlighted as **first**, **second**.

file). As scenes are updated, the knowledge acquired from past scenes is largely retained. Compared to the baseline, our method effectively mitigates catastrophic forgetting.

Our approach benefits from the advantages offered by 3DGS [12] for explicit representation, enabling it to detect alterations in scene geometric layout. As depicted in Figure 5, our method effectively discerns variations in scene geometry resulting from the act of opening and closing the car door. While CLNeRF uses an implicit radiance field to represent the scene, it lacks the ability to capture geometric changes in the scene. Please refer to our supplementary material for more experimental results.

4.3. Ablation Study

Removal Factors. We begin by analyzing the consequences of not learning the removal factors on our results. As illustrated in Figure 3, without learning removal factors result in the persistence of objects that should have disappeared, thereby producing artifacts. Additionally, the quantification in Table 3 demonstrates the effectiveness of learning the removal factors in deleting disappearing objects.

Testing on		WAT			
		$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	$\mathbf{T}_4$
Training on	$\mathbf{T}_1$	26.92	-	-	-
	$\mathbf{T}_2$	26.90	26.05	-	-
	$\mathbf{T}_3$	26.86	25.99	26.68	-
	$\mathbf{T}_4$	26.84	25.92	26.61	26.42

Table 4. **Average rendering results from $\mathbf{T}_1$ to $\mathbf{T}_4$ for sequential updates on WAT Dataset**.

Self-aware Addition of 3D Gaussians. In the second stage of model updating, we learn the newly added objects by incorporating sparse point initialization and densifying only the newly added points. To analyze the effectiveness of adding sparse points, we conducted experiments where we solely relied on the densify operation to learn new objects. As shown in Figure 3 and Table 3, using only the densified operation of 3DGS to add points does not sufficiently learn the newly appearing objects. However, leveraging the rough geometric cues from the sparse point cloud leads to improved results.

Layout-invariant Mask. We also analyzed whether to use the layout-invariant mask. As shown in Figure 3 and Table 3, neglecting the layout-invariant mask can cause the model to adapt to the visual alterations in the geometric change area during the initial training stage, potentially leading to the failure of removal for disappeared objects.

5. Conclusion

This paper presents GaussianUpdate, a novel approach integrating 3D Gaussian Splatting with continual learning. Our method updates the neural model with current data while preserving past information, explicitly modeling various changes through a multi-stage strategy. Additionally, our visibility-aware continual learning with generative replay supports self-aware updates without storing images. Experimental results demonstrate superior rendering quality and effective visualization of changes over time.

Limitation. Similar to most existing 3DGS-based methods, GaussianUpdate also fails to accurately model scenes with strong reflections (e.g., mirrors), and updates in these areas result in incorrect geometry or appearance, which may be solved by integrating more advanced 3D representations in the future.

Acknowledgments. This work was supported by the National Key R&D Program of China (Grant No. 2024YFB4505500 & 2024YFB4505501). We also express our gratitude to all the anonymous reviewers for their professional and insightful comments.

References

[1] Boming Zhao and Bangbang Yang, Zhenyang Li, Zuoyue Li, Guofeng Zhang, Jiashu Zhao, Dawei Yin, Zhaopeng Cui, and Hujun Bao. Factorized and controllable neural re-rendering of outdoor scene for photo extrapolation. In *Proceedings of the 30th ACM International Conference on Multimedia*, 2022. 2

[2] Zhipeng Cai and Matthias Müller. Clnerf: Continual learning meets nerf. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23185–23194, 2023. 2, 3, 5, 6, 7

[3] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2

[4] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. 3, 6

[5] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 2

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, pages 226–231, 1996. 5

[7] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2

[8] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2

[9] Jiarui Hu, Mao Mao, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cp-slam: Collaborative neural point-based slam system. *Advances in Neural Information Processing Systems*, 36:39429–39442, 2023. 2

[10] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. In *European Conference on Computer Vision*, pages 93–112. Springer, 2024. 3

[11] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 6, 8

[13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 4

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 3

[15] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 3

[16] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. *arXiv preprint arXiv:2312.11458*, 2023. 2

[17] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. *arXiv:2312.03431*, 2023. 3

[18] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 3, 6

[19] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 3

[20] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[21] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, pages 109–165. Elsevier, 1989. 1

[22] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 5

[23] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024. 5

[24] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2, 3

[25] Weining Ren, Zihan Zhu, Boyang Sun, Jiaqi Chen, Marc Pollefeys, and Songyou Peng. Nerf on-the-go: Exploiting uncertainty for distractor-free nerfs in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[26] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 6

[27] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016. 5

[28] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018. 3

[29] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017. 3, 5

[30] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. *arXiv preprint arXiv:2403.01444*, 2024. 2, 3

[31] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 2, 3, 6, 7

[32] Xiuzhe Wu, Peng Dai, Weipeng Deng, Handi Chen, Yang Wu, Yan-Pei Cao, Ying Shan, and Xiaojuan Qi. Cl-nerf: Continual learning of neural radiance fields for evolving scene representation. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3, 6

[33] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2, 3

[34] Boming Zhao, Yuan Li, Ziyu Sun, Lin Zeng, Yujun Shen, Rui Ma, Yinda Zhang, Hujun Bao, and Zhaopeng Cui. Gaussianprediction: Dynamic 3d gaussian prediction for motion extrapolation and free view synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024. 2, 3

[35] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 3