

Close-up-GS: Enhancing Close-Up View Synthesis in 3D Gaussian Splatting with Progressive Self-Training

Jiatong Xia Lingqiao Liu*

Australian Institute for Machine Learning, The University of Adelaide

{jiatong.xia, lingqiao.liu}@adelaide.edu.au

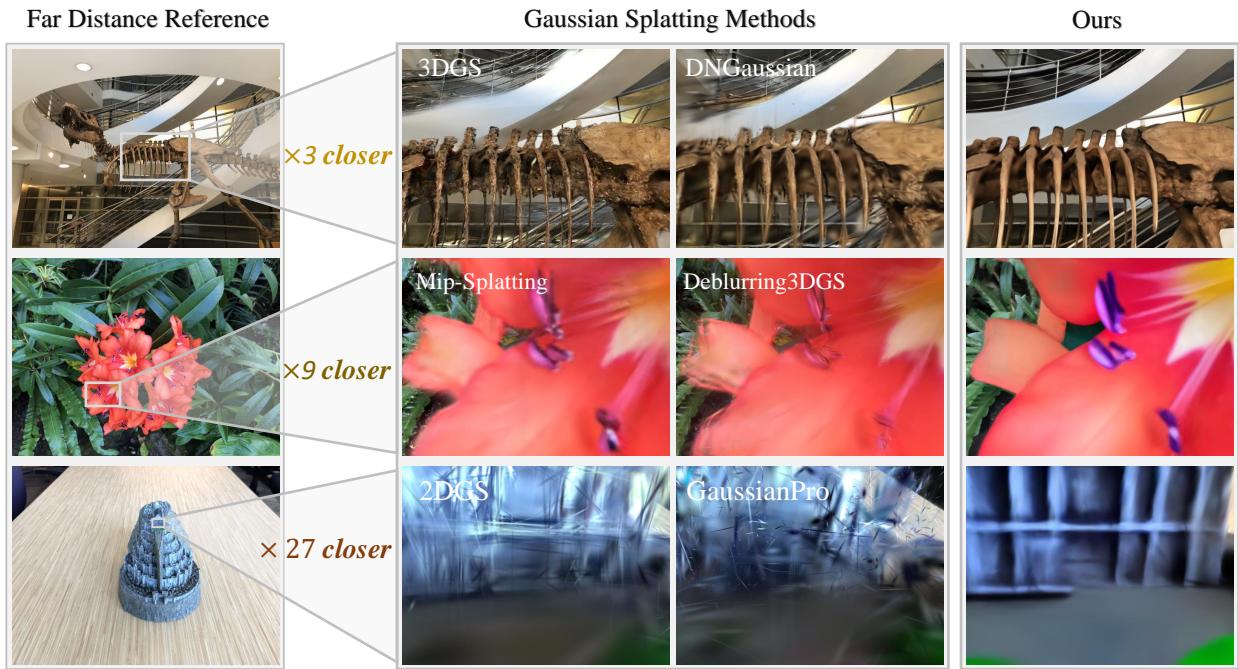


Figure 1. **Close-up view synthesis** remains a challenge in Gaussian Splatting, often lacking effective solutions. In this paper, we aim to address the close-up views issue by proposing a novel method Close-up-GS. Our method demonstrates impressive rendering quality across different close-up scales, even at a $27\times$ close-up distance, achieving extreme close-up with remarkable details.

Abstract

3D Gaussian Splatting (3DGS) has demonstrated impressive performance in synthesizing novel views after training on a given set of viewpoints. However, its rendering quality deteriorates when the synthesized view deviates significantly from the training views. This decline occurs due to (1) the model’s difficulty in generalizing to out-of-distribution scenarios and (2) challenges in interpolating fine details caused by substantial resolution changes and occlusions. A notable case of this limitation is close-up view generation—producing views that are significantly closer to the object than those in the training set. To tackle this issue, we propose a novel approach for close-up view generation based by progressively training the 3DGS model with self-

generated data. Our solution is based on three key ideas. First, we leverage the See3D model, a recently introduced 3D-aware generative model, to enhance the details of rendered views. Second, we propose a strategy to progressively expand the “trust regions” of the 3DGS model and update a set of reference views for See3D. Finally, we introduce a fine-tuning strategy to carefully update the 3DGS model with training data generated from the above schemes. We further define metrics for close-up views evaluation to facilitate better research on this problem. By conducting evaluations on specifically selected scenarios for close-up views, our proposed approach demonstrates a clear advantage over competitive solutions.

1. Introduction

Advances in radiance field methods have enabled remarkable progress in synthesizing novel views of a 3D scene from input training views. Among these approaches, recent work on 3D Gaussian Splatting (3DGS) [12] has drawn significant attention for its ability to produce high-quality renderings in real time. By modeling a scene with an explicit set of 3D Gaussians, and optimizing a radiance field over these primitives, 3DGS reduces both training and inference costs compared to implicit representations (e.g., NeRF [23]). Through an efficient rasterization rendering pipeline, 3DGS has demonstrated impressive fidelity and speed for novel view synthesis.

Despite its strengths, 3DGS struggles when required to predict appearances under out-of-distribution conditions. In particular, close-up view synthesis—where the camera moves significantly closer to an object than any of the training viewpoints—leads to notable degradations. Two issues predominantly underlie this limitation. First, a large fraction of the rays in these novel views diverge substantially from what the model has seen during training, leading to artifacts in the color rendering. Second, any fine details that were interpolated or occluded in the training views are difficult to restore from the 3DGS alone. Currently, there is still a gap in research to address all these issues well and provide an effective solution to the close-up views problem.

To address this challenge, we propose a progressive self-training framework for close-up view synthesis, guided by three core ideas that build upon the strengths of 3D Gaussian Splatting (3DGS) while overcoming its limitations in out-of-distribution scenarios. First, we leverage See3D, a newly developed 3D-aware generative model that can generate novel views with the guidance of existing views, to refine the missing details in newly rendered close-up views. See3D serves as a data-driven prior, injecting intricate details that would otherwise remain absent when the camera moves significantly closer than the training distribution allows¹. Second, we introduce a progressive update scheme to expand the 3DGS “trust region,” which systematically identify and select intermediate novel views between the far-distance training set and the final close-up vantage points. This approach mirrors the idea of pseudo-labeling in semi-supervised learning: some refined (and verified) output becomes a new training sample that helps 3DGS adapt to increasingly difficult viewpoints, effectively bridging the gap from far to near distances. Finally, we design a specialized fine-tuning procedure for 3DGS, ensuring consistent geometry and color mappings between real and self-generated images. By introducing new metrics focused on

¹Note using the generative model inevitably introduces content that might not always match the unobserved ground-truth. In this work, our aim is to create a close-up view that balances the fidelity and visual quality.

evaluating close-up render quality, we offer a thorough assessment of this method’s practical impact. Our experiments on challenging benchmarks confirm that this progressive self-training paradigm substantially outperforms existing solutions, extending the utility of Gaussian Splatting to scenarios that demand precise, high-quality results at extreme close-ups.

2. Related Work

Radiance Fields. The development of radiance fields has significantly advanced the learning of 3D representations from 2D images. This type of methods were first introduced by Neural Radiance Fields [23] (NeRF), which implicitly represent a scene using multi-layer perceptions (MLPs) and applying volume rendering for pixel rendering. Since then, NeRF-related techniques have been extensively adopted in various applications [13, 17, 39]. However, NeRF has several limitations, primarily due to its implicit representation, which relies on neural networks for ray rendering, leading to high computational costs and inefficiencies in both training and inference. Several advancements [3, 24, 36] have been made, enabling a much faster process. Compared to NeRF, 3D Gaussian Splatting (3DGS) [12] employs an explicit representation to model radiance fields, significantly reduces training time while enabling high-quality and real-time rendering. With the widespread application of 3DGS, further improvements [4, 6, 11, 15] have been proposed to enhance its ability in different scenarios. Yuan et al. [10] introduced 2D Gaussian Splatting (2DGS), which provides accurate and view-consistent geometry, and enhance the quality of the reconstructions. Yu et al. [38] addressed aliasing issues by designing sampling filters based on sampling principles. These Gaussian Splatting related method advancements significantly enhanced radiance fields, enabling them to achieve remarkable performance in faithfully reconstructing real-world scenes.

Unconventional Condition Radiance Fields. As Gaussian splatting and other radiance field methods are applied to real-world scenarios, various challenges have emerged under unconventional conditions, leading to targeted improvements. For example, when the number of training views is insufficient, the rendering results will suffer from low quality. Several methods [7, 14, 25, 28] first addressed this issue in NeRF, and more recently, methods [19, 34] like DNGaussian [16] tackle this problem in 3DGS. Another situation different from limited views is single-view reconstruction; many works [30, 35, 43] have been proposed to address this issue. As the study of radiance fields deepens, it has also been discovered that even with sufficient training samples, issues arise when rendering from viewpoints that deviate from the training views, this is known as an out-of-distribution (OOD) issue. SplatFormer [5] intro-

duced this issue in general 3DGS. They use low-elevation views as training views and enhance 3DGS in the high-elevation views rendering, yet it does not specifically address the close-up view problem. Xia et al. [33] first highlight the OOD problem when moving the camera closer and proposed a method based on geometric consistency to generate reliable sparse pixels for those views. However, their approach does not address issues such as occluded regions or interpolated details at close distances. In general, there is still a lack of effective solutions to address rendering issues in close-up views.

Diffusion Priors. Applying diffusion models [8, 9, 29, 40] to generate reliable images or enhance existing images [21, 37] has become a well-established technology. Due to the scalability of diffusion models, their effectiveness has been evolving at a fast speed. Using the priors of diffusion models to obtain 3D representations such as radiance fields, has become a key area of research. Many recent works [18, 27, 31] apply score distillation sampling (SDS) to distill 2D diffusion priors into 3D representations, however, they currently limited to object-level reconstruction. A recently proposed See3D [20] applies a reference-guided, warping-based generation framework. It utilizes known images as references and warped results of known images under novel views as input to restore the novel view results, demonstrating strong performance in generating novel views for complex scenes. Thus, using See3D’s priors to enhance radiance fields holds significant potential, while it has many challenges. 3D representations such as 3DGS impose strict constraints on geometric consistency, and subtle inconsistencies can impact the rendering quality. See3D can maintain some level of consistency, but it faces challenges in achieving a fully consistent reconstruction across all views, particularly in close-up scenarios. In this paper, we aim to propose a robust method that leverages See3D’s priors to address the challenges of close-up views in 3DGS.

3. Preliminary: Gaussian Splatting

Gaussian splatting [12] represents a 3D scene using anisotropic 3D Gaussians and enable image rendering through a differentiable tile rasterization process. Each 3D Gaussian primitive $G_k(p)$ at a center point $p_k = \{x_k, y_k, z_k\}$ is described together with a covariance matrix Σ_k , the function of the k -th primitive can be formulate as:

$$G_k(p) = \exp\left(-\frac{1}{2}(p - p_k)^T \Sigma_k^{-1} (p - p_k)\right), \quad (1)$$

To render an image, 3D Gaussians are transformed into 2D camera coordinates using the world-to-camera transform matrix W and the Jacobian J of the affine approximation of the projective transformation [44] as follows:

$$\Sigma' = JW\Sigma W^T J^T. \quad (2)$$

A point-based rendering is utilized to compute the color C of each pixel, by blending \mathcal{N} depth-ordered Gaussians overlapping the pixel, with each Gaussian has its opacity σ and color c which is determined using the SH coefficient with the viewing direction. Formulate as:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (3)$$

where $\alpha_i = \sigma_i G_i(p)$. The optimization process minimizes the difference between the rendered color and the ground truth images, using the following loss function:

$$\mathcal{L} = (1 - \lambda) \mathcal{L}_1 + \lambda \mathcal{L}_{SSIM} \quad (4)$$

where \mathcal{L}_1 and \mathcal{L}_{SSIM} represent the image \mathcal{L}_1 loss and the similarity loss, respectively.

4. Methods

4.1. Close-up Views Rendering Problems

Our work focuses on the problem of generating novel close-up views around an area-of-interest given a set of far-distance training views. Formally, we assume an initial 3DGS model has already been trained from a collection of known views $\mathcal{V}_s = \{V_i\}, i = 1 \dots n$. The center of object-of-interest is given as p_{target} ². The aim of our approach is to update the 3DGS model to render high-quality views around p_{target} .

This task is a non-trivial task due to two reasons: (1) The rays that can be accurately rendered by Gaussian Splatting are within the regions that can be interpolated from the rays in training views. Close-up views inevitably include rays that deviate from the known regions, which makes it difficult for 3DGS to render accurately. A similar issue occurs when rendering rays pass through areas that training rays are occluded, the missing information in the occluded regions can cause artifacts in 3DGS rendering. (2) In addition, the finer details in the close-up region are interpolated from far-distance views, the limited resolution of these distant images often results in lower-quality close-up details.

4.2. Enhancement with Generative Models

To address the above limitation, this paper leverages a generative model to infer missing details using its rich prior knowledge. Specifically, we adopt the recently proposed See3D [20], defined as \mathcal{M}_θ , which utilizes reference views to guide novel view generation. It generates a novel view based on several inputs: (1) multiple reference views, (2) an initial image input of the novel view, which can be obtained by wrapping from reference views, and (3) a mask

²For the simplicity, we assume a single center for the object-of-interest. Our approaches can be straightforwardly generalized to the case with multiple centers if the object of interest involves multiple objects.

highlighting which pixels are unknown. Unlike standard image generative models, See3D generates with the guidance of other views and thus can better preserve the consistency across view. Moreover, its mask input allows user to define the known pixel and unknown pixel, making it ideal for the close-up rendering problem as we wish the generation not deviating too much from the content can be rendered from existing 3DGS model. Formally, See3D creates the novel view image via:

$$I_n = \mathcal{M}_\theta(I'_n, M'_n, \{I_k^{ref}\})$$

In our work, we propose a scheme to seamlessly integrate See3D to mitigate information loss in out-of-distribution view rendering. Given a novel view, we first render it using a pretrained 3DGS model and then identify reliable regions using the method from Xia et al. [33]. This produces a mask highlighting pixels that are geometrically consistent with the training views, which serves as input for the See3D model. The model then refines the novel view, and its output can undergo further enhancement, such as super-resolution [37], for improved interpolated close-up details and quality. Once the refined image is generated, it can be used to update the Gaussian Splatting by treating it as a new training view.

Although conceptually simple, this approach faces some challenges when generating close-up views. To achieve high-quality generation, it is essential that the reference views provide sufficient coverage of the novel view, allowing the See3D model to fully utilize their guidance. Thus, the See3D model should be used to generate views that are not too far from the reference views. This implies that the 3DGS model update process should be performed incrementally in multiple steps, and each step requires a carefully designed scheme to select the reference views and the views to be updated, ensuring both high generation quality and efficiency. Additionally, the reference views should be continuously updated, and the update process of the 3DGS model must be specially designed to maintain consistency across views.

4.3. Progressive model update scheme

In this paper, we propose a scheme to progressively updating the 3DGS model by gradually expanding the “trust region” of the 3DGS, ie, the region in which the 3DGS model can generate high-quality views. This process involves multiple rounds of expansion, which we illustrate in Fig. 2. In each expansion, the scheme involves four steps: (1) select the anchor views \mathcal{R}_s of the current known views. (2) Select the to-be-updated views \mathcal{V}_u outside of the current trust region. (3) render and refine the views to be updated using anchor views as references. (4) update the 3DGS model by fine-tuning it on the to-be-updated views and then append

those views to the known views. The whole process is akin to the pseudo-labeling process in semi-supervised learning.

In our implementation, we separate the whole process into multiple expansion stages, based on the distance to the center of the object-of-interest p_{target} . The frontier of the trust region will get closer to p_{target} in each round. More implementation details can be found in section 5.2. In the following sections, we elaborate on the algorithm details for each step in the process.

4.3.1. Select anchor views

In each expansion round, our method selects k anchor views from n known views ($k < n$). Those anchor views will be used later in the other steps, such as reference views for See3D, using k anchors rather than all known views helps reduce computational cost. Intuitively, the selection of the anchor views should follow two principals: (1) the view should cover the object of interest as much as possible. (2) The redundancy of view should be minimized.

To evaluate how well a view covers the object-of-interest, our method identifies a set of frontier views through the following process. Initially, we select an anchor view—either discovered in the previous round or randomly chosen in the first round—and draw a line segment connecting it to p_{target} with a length of d . A virtual camera is then positioned along this line at a distance of $\alpha_t \cdot d$ from p_{target} , where $0 < \alpha_t < 1$ is a predefined constant for each round. The camera’s orientation is subsequently adjusted to ensure that p_{target} is projected at the center of the image. With k anchor views, this process will produce k frontier views $\{v_j^f\}, j = 1 \dots k$. These frontier views represent the closest views achievable within the distance constraints for camera placement in the current round.

Using these frontier views, we can quantify how well a known view covers the target region. This is determined by calculating the number of pixels visible in both the frontier view and the known view³. Averaging the results from k frontier views, we obtain a coverage score w^i for the i -th known view. In a similar way, we can quantify the redundancy between two views by counting how many pixels are visible in both views. This produces a pairwise view similarity score $e_{i,j}$. Representing $\{w^i\}$ in a vector form as $\mathbf{w} \in \mathbb{R}^n$ and $\{e_{i,j}\}$ in a matrix form as $\mathbf{E} \in \mathbb{R}^{n \times n}$, anchor view selection can be formulated as an optimization problem:

$$\begin{aligned} & \max_{\mathbf{s}} \mathbf{s}^\top \mathbf{w} - \mathbf{s}^\top \mathbf{E} \mathbf{s} \\ & \text{s.t. } \mathbf{s}^\top \mathbf{1} = k \end{aligned} \quad (5)$$

where $\mathbf{s} \in \mathbb{R}^n$ is a binary vector, with “1” element indicates select and “0” not select. This is a combinatorial optimization problem and exists greedy approximation solution. We

³This can be achieved by warping the frontier view to align with the known view. In practice, higher weights are assigned to pixels near the center of the frontier view, as they are more relevant to the object-of-interest.

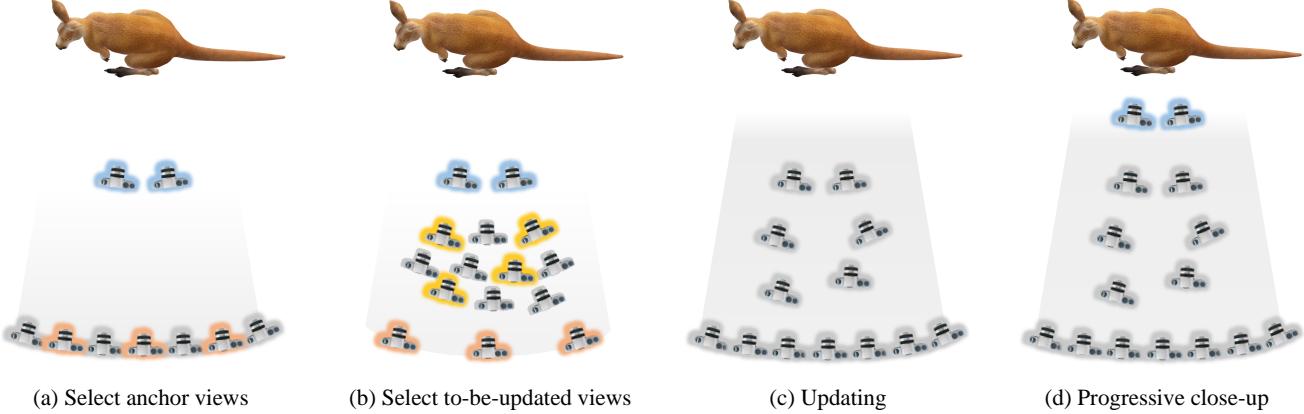


Figure 2. Progressive update. (a) Setting the frontier views (blue camera) and select the anchor views (orange camera) from known views (gray camera) based on our views selection algorithm. The anchor views will also serve as the reference views of See3D. (b) Select the to-be-updated views (yellow camera) from a variety of random views between frontier views and anchor views, also using the selection algorithm. (c) Render and refine the to-be-updated views and frontier views, then apply our fine-tuning procedure to update 3DGS therefore extend its “trust region” (gray region). The to-be-update views and frontier views are then added to the known views. (d) Begin the next round by setting new frontier views.

leverage the solution which is detailed in the supplementary to calculate the optimal selection as k anchor views.

4.3.2. Select to-be-updated views

In a given expansion round, anchor views serve as a summary of the known views, while the frontier views represent the targets we aim to reach. Our progressive model update scheme selects a set of to-be-updated views located between the known views and the frontier views for rendering, refinement, and updating of the 3DGS model. The selection of to-be-updated views begins by randomly sampling M views whose distance to p_{target} is smaller than that of the known views but larger than that of the frontier views. We then apply a similar procedure as used for selecting anchor views to obtain the final set of to-be-updated views. This process is illustrated in Fig. 2 (b).

Specifically, we still solve a similar optimization problem as in Eq. 5 to select views and use the same way to calculate the similarity score. The only difference is when we calculate w^j , we also consider both the view overlap with the frontier view and with the anchor view. This is achieved by taking the average of the pixel overlap count calculated from both frontier views and with the anchor views. This approach ensures that the to-be-updated view serves as a “bridge” between the known view and the frontier view, preventing the See3D-generated view from deviating too far from the reference view—i.e., the anchor views in our framework. Additionally, we apply a discount factor, as detailed in the supplementary materials, to mitigate the influence of distance in the calculation.

4.3.3. Render and refinement

Once the to-be-updated views are selected, we could run 3DGS model and See3D model to render and refine the

views, where the k anchor views will be used as references for See3D here. We also employ an image enhancement model [37] to further enhance the details and quality of the refined to-be-updated view images. Those view images are then used for updating the 3DGS model. These identified updated views are also be appended to the known views for the next round calculation.

4.3.4. Gaussian Splatting Fine-tuning

The most straightforward approach to updating the 3DGS model is fine-tune it using the newly refined images. However, this approach may not yield the best results, as the generated content might still exhibit color and geometry inconsistencies compared to the training views. Therefore, a constraint derived from the real data is necessary. As a result, we also incorporate the reliable pixels obtained during the refinement process into the fine-tuning procedure.

Another crucial aspect of 3DGS fine-tuning is the densification of Gaussian primitives. Increasing the density of primitives enhances the model’s capability in modeling complex texture regions. In practice, we apply Gaussian primitive densification only when the to-be-updated views are within $\frac{1}{3} \times$ the original distance. This scheme enables the model to gradually increase the number of Gaussian primitives as more views are rendered and incorporated as new training data.

5. Experiments

In Sec. 5.1, we provide a detailed description of the scenes used, with a particular emphasis on close-up views and the evaluation metrics for this issue. In Sec. 5.2, we present our implementation details. In Sec. 5.3, we show the results of our method and other widely used methods. In Sec. 5.4, we

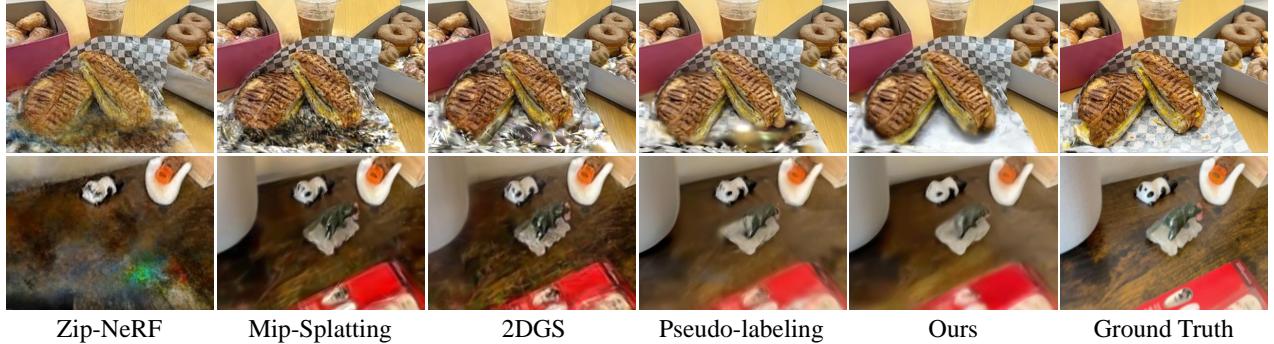


Figure 3. Qualitative comparisons with representative methods on extracted LERF dataset.

Methods	PSNR↑	SSIM↑	LPIPS↓	DINO↑	MetaIQA↑
NeRF [23]	14.93	0.569	0.623	0.767	0.275
Mip-NeRF [1]	16.49	0.577	0.623	0.799	0.269
Zip-NeRF [2]	16.54	0.583	0.565	0.803	0.309
3DGS [12]	16.22	0.558	0.560	0.814	0.353
2DGS [10]	16.75	0.600	0.538	0.847	0.376
GaussianPro [6]	16.09	0.548	0.563	0.813	0.308
Deblurring3DGS[15]	16.11	0.545	0.574	0.793	0.325
DNGaussian [16]	16.28	0.599	0.605	0.769	0.310
Mip-Splatting [38]	16.69	0.591	0.542	0.826	0.346
Pseudo-labeling [33]	17.48	0.628	0.513	0.881	0.403
Ours	17.94	0.640	0.502	0.886	0.431

Table 1. Quantitative comparisons on extracted LERF dataset.

conduct ablation studies on the proposed methods.

5.1. Close-up Views Data and Metrics

Extracted LERF dataset. Existing widely used datasets do not specifically focus on the issue of close-up views. In order to have numerical results compared to ground-truth images, we chose to extract scenes specifically tailored to close-up views from the LERF dataset [13]. LERF dataset is proposed for open-vocabulary tasks, and its scenes include both distant views and close-up views of objects, and we choose views from the long-distance as training views, and views from the close-up distance as testing views. We conducted a statistic analysis of the distances between the object and the views. The testing views are mainly positioned at a distance $3\times$ closer than that between the object and the training views. This fits well as one round in the progressive update process. Therefore, we utilize the extracted LERF dataset for single-round update experiments, accompanied by ground-truth comparisons.

Close-up on LLFF dataset. We present the close-up experiments on the most widely used novel view synthesis dataset, LLFF [22]. We selected an object-of-interest in different scenes, and conduct comparisons on three close-up scales: $3\times$, $9\times$, and $27\times$. This involves using the metrics we proposed for numerical comparison.

Close-up views evaluation metrics. When ground truth images are available, we use standard metrics from novel view synthesis, including PSNR, SSIM [32], LPIPS [41]. However, obtaining ground-truth images for most close-up view scenarios is challenging, especially for extreme close-ups that are difficult to capture in practice. Therefore, we introduce two new metrics to evaluate close-up views. One metric involves using the training image with the most coverage of the close-up objects as a reference to evaluate the DINO [26] score of the close-up views. It reflects the correlation between the rendered close-up images and the original images. We also select a no-reference image quality assessment metric MateIQA [42] to obtain a numerical score for the quality of the rendered close-up view images. These two metrics deliver an objective evaluation of close-up views without requiring ground-truth images.

5.2. Implementations

We divided our progressive update into 3 rounds, progressively moving the frontier views $3\times$, $9\times$, and $27\times$ closer to the original distance to P_{target} (each round moves $3\times$ closer compared to the previous round), achieving an extreme close-up at $27\times$ closer distances. For the view selection at each round, we use our selection method to choose 5 reference views and 8 to-be-updated views. For single-round update on extracted LERF dataset, we apply the testing views as the frontier views to run our method for one round. We choose the 30000 iterations initial optimized 2DGS [10] using training views as our baseline method. In each update round, the refined samples, along with the reliable pixel samples obtained during the refine process, are added to the existing training samples, to fine-tune the Gaussian Splatting for 5000 iterations. And those update views with a distance $3\times$ closer are used to densify the Gaussian primitives. We conduct our experiments on NVIDIA 4090 GPUs.

5.3. Main Results

Single-round update on extracted LERF dataset. Tab 1 shows the comparison under close-up views with widely

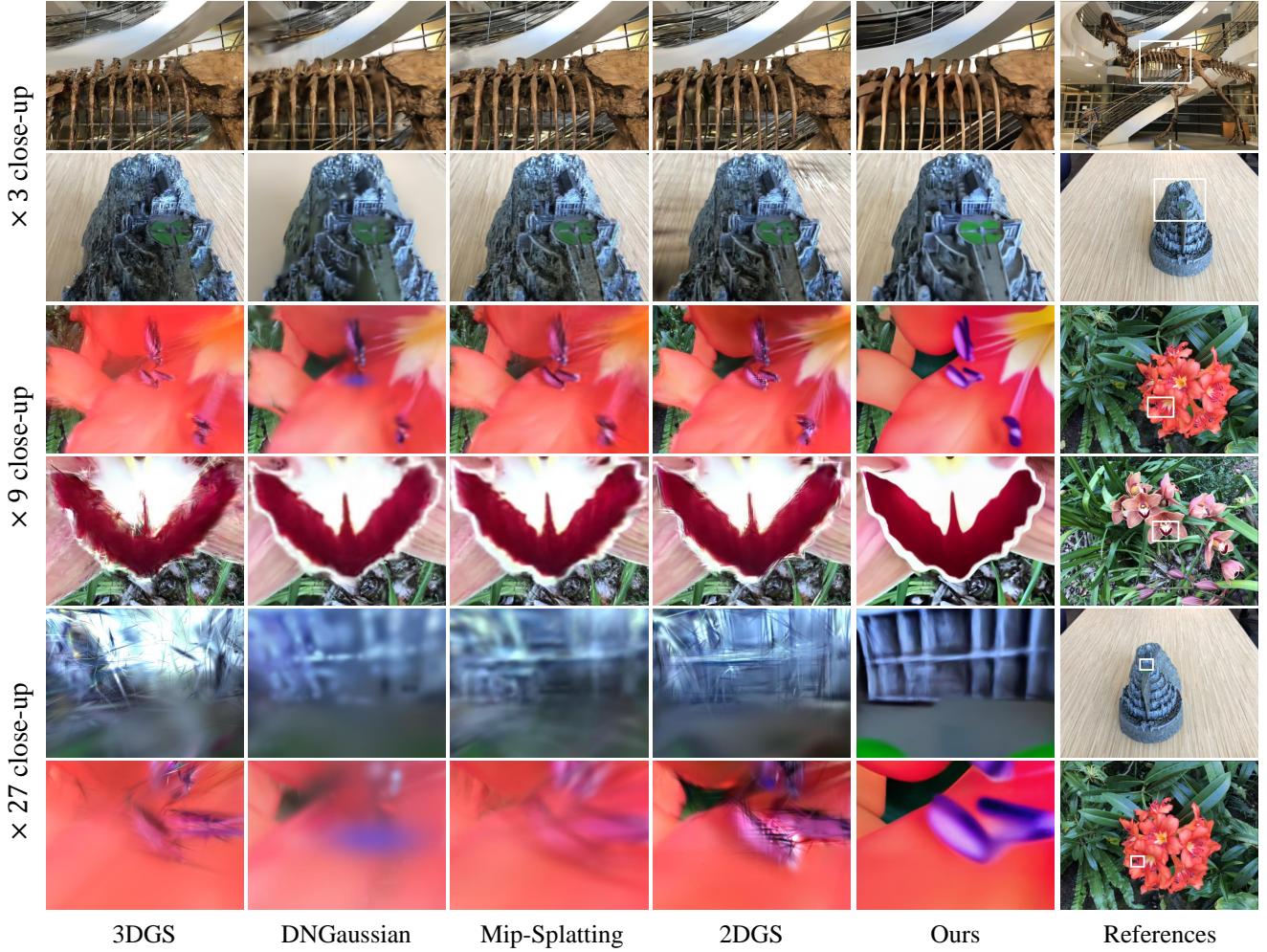


Figure 4. Comparisons of progressive update on LLFF dataset with 3 close-up scales, each scale showing two rows of results. The far distance reference images are displayed on the right, with the white box outlines the approximate close-up region.

Methods	Last Round (27× closer)		Overall	
	DINO↑	MetalQA↑	DINO↑	MetalQA↑
3DGS [12]	0.580	0.287	0.695	0.344
2DGS [10]	0.578	0.279	0.703	0.341
GaussianPro [6]	0.573	0.272	0.694	0.321
Delurring3DGS [15]	0.598	0.291	0.706	0.338
DNGaussian [16]	0.605	0.260	0.703	0.301
Mip-Splatting [38]	0.601	0.254	0.724	0.313
Pseudo-labeling [33]	0.586	0.265	0.718	0.343
Ours w/o progressive	0.587	0.322	0.728	0.387
Ours w/o densify	0.602	0.324	0.721	0.391
Ours	0.616	0.360	0.730	0.411

Table 2. Comparisons of progressive update on LLFF dataset. The last round is the 27× close-up update round.

used methods, our method shows significant improvements in all metrics compared to other methods. Specifically, our method outperforms others in traditional metrics including PSNR, SSIM, and LPIPS, highlighting its ability to pro-

duce high-quality close-up images with strong consistency. The substantial improvement on MetalQA further demonstrate the clear enhancement in image quality achieved by our method. We show visualization comparisons with some representative methods in Fig. 3, especially in row 1, where other methods including Pseudo-labeling [33] exhibited issues with artifacts. Our method effectively addresses this issue, demonstrate a notable improvement in close-up views rendering quality, presenting more complete and smoother results.

Progressive close-up update on LLFF dataset. Fig. 4 shows visualization results of views in three different close-up scales. As close-up scale increases, rendering issues from other methods become more evident, whereas our approach maintains excellent rendering quality at all scales. Even at a 27× close-up distance, it still produces impressive results, as seen in the reference images, the views refer

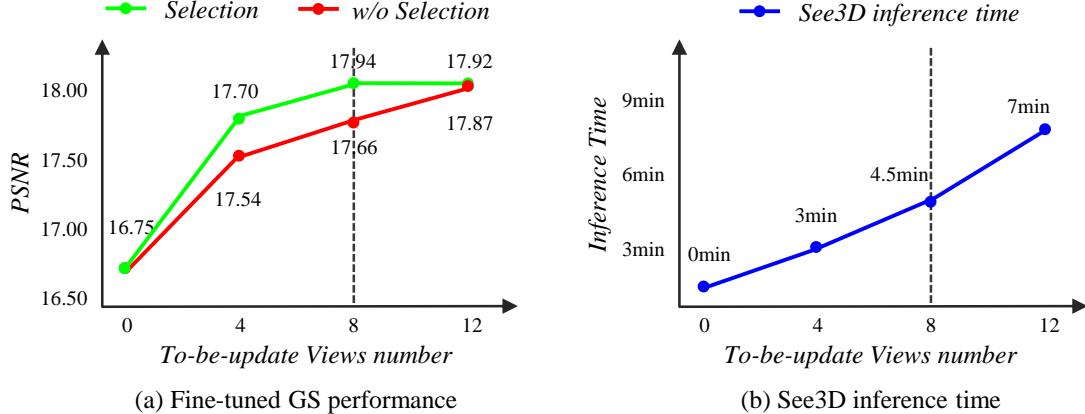


Figure 5. Ablation studies of views selection. (a) shows a single-round updated Gaussian Splatting’s PSNR performance on LERF dataset using same number of anchor views and different number of to-be-update views. (b) shows the corresponding See3D’s inference time.

to a very small area in the original training image, yet our method effectively handles it. This can also be observed in Tab. 2, for the $27 \times$ close-up scale, our method achieves a significant lead in image quality MetaIQA. In the overall evaluation, our method also demonstrates a clear and significant lead.

5.4. Ablation Studies

Ablation studies of refine methods. We evaluate the improvement of different refine methods in Tab. 3. It is important to note that the See3D results show sub-par performance in PSNR and SSIM, indicating the issue with consistency between the See3D results and the real data. Directly fine-tuning with the See3D results reduces some of the inconsistencies, but there is still a significant gap from the optimal result. Reliable pixels can lead to good fine-tuning results, since they are sparse, they cannot achieve the optimal outcome. The combination of reliable pixels and See3D results achieve excellent fine-tuning, with the subsequent enhancement leading to the best performance. It should be noted that directly enhancing the reliable pixels does not improve the results; the optimal solution is to enhance the See3D results and use reliable pixels to correct the inconsistencies.

Ablation studies of views selection. Fig 5 shows the effectiveness of our views selection strategy. Compared to updates without view selection, our method enables the update process to fully utilize the limited number of views, achieving better results in much less See3D inference time without views selection. This significantly improves update efficiency, ensuring the best outcomes with minimal cost.

Ablation studies of Gaussian Splatting update. As shown in Tab. 2, compared to the fine-tuning without progressive updates and without Gaussian primitives densify,

Methods	LERF dataset		
	PSNR↑	SSIM↑	LPIPS↓
Baseline [10]	16.75	0.600	0.538
See3D render	16.35	0.566	0.492
See3D update	17.22	0.614	0.517
Reliable Pixels update	17.65	0.635	0.514
Reliable Pixels + Enhance update	17.48	0.621	0.519
Reliable Pixels + See3D update	17.86	0.636	0.506
Reliable Pixels + See3D + Enhance update	17.94	0.640	0.502

Table 3. Ablation studies of refine methods. “See3D render” are the results from See3D, while others use rendered view to update the model. “Reliable Pixels” refers to the reliable pixels obtained during the refine process, and “Enhance” is the image enhancement operation.

our method achieves higher overall correlation and image quality. Especially in the extreme close-ups (last round), both factors play a crucial role in delivering a noticeable improvement for our approach.

6. Conclusion

This work effectively addresses the long-standing challenge in existing Gaussian Splatting methods of generating reliable images from close-up views. We propose an approach that leverages existing generative models to refine close-up novel views and apply them to progressively expand the Gaussian Splatting’s “trust region”. A proposed views selection strategy ensures this process in a cost-effective manner. Additionally, we introduce a Gaussian Splatting fine-tuning procedure to ensure high-quality optimization in the close-up regions. We also define new metrics and specific scenes to better evaluate close-up views. Overall, our approach demonstrates a robust and effective solution for handling close-up views problem.

References

- [1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021. 6
- [2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 6
- [3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, pages 333–350, 2022. 2
- [4] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, pages 370–386, 2024. 2
- [5] Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. Splatformer: Point transformer for robust 3d gaussian splatting. In *ICLR*, 2025. 2
- [6] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *ICML*, 2024. 2, 6, 7
- [7] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, pages 12882–12891, 2022. 2
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3
- [9] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *NeurIPS*, pages 8633–8646, 2022. 3
- [10] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*, 2024. 2, 6, 7, 8
- [11] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *CVPR*, pages 5322–5332, 2024. 2
- [12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 2, 3, 6, 7
- [13] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. LERF: Language embedded radiance fields. In *ICCV*, pages 19729–19739, 2023. 2, 6
- [14] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*, 2022. 2
- [15] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. In *ECCV*, pages 127–143, 2024. 2, 6, 7
- [16] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024. 2, 6, 7
- [17] Quwei Li, Feichao Li, Jie Guo, and Yanwen Guo. UHD-NeRF: Ultra-high-definition neural radiance fields. In *ICCV*, pages 23097–23108, 2023. 2
- [18] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In *CVPR*, pages 6517–6526, 2024. 3
- [19] Xi Liu, Chaoyi Zhou, and Siyu Huang. 3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors. In *NeurIPS*, 2024. 2
- [20] Baorui Ma, Huachen Gao, Haoge Deng, Zhengxiong Luo, Tiejun Huang, Lulu Tang, and Xinlong Wang. You see it, you got it: Learning 3d creation on pose-free videos at scale. In *CVPR*, 2025. 3
- [21] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 3
- [22] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019. 6
- [23] 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. 2, 6
- [24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):102:1–102:15, 2022. 2
- [25] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Reg-NeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, pages 5480–5490, 2022. 2
- [26] Maxime Oquab, Timothée Darct, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 6
- [27] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3
- [28] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, pages 12892–12901, 2022. 2
- [29] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021. 3
- [30] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. *CVPR*, 2024. 2
- [31] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *ICLR*, 2024. 3
- [32] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6

- [33] Jiatong Xia, Libo Sun, and Lingqiao Liu. Enhancing close-up novel view synthesis via pseudo-labeling. In *AAAI*, 2025. 3, 4, 6, 7
- [34] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsesgs: Real-time 360° sparse view synthesis using gaussian splatting, 2023. 2
- [35] Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint 2401.04099*, 2024. 2
- [36] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, pages 5752–5761, 2021. 2
- [37] Fanghua Yu, Jinjin Gu, Zheyuan Li, Jinfan Hu, Xiangtao Kong, Xintao Wang, Jingwen He, Yu Qiao, and Chao Dong. Scaling up to excellence: Practicing model scaling for photo-realistic image restoration in the wild. In *CVPR*, pages 25669–25680, 2024. 3, 4, 5
- [38] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *CVPR*, pages 19447–19456, 2024. 2, 6, 7
- [39] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. NERS: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *NeurIPS*, pages 29835–29847, 2021. 2
- [40] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 3
- [41] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 6
- [42] Hancheng Zhu, Leida Li, Jinjian Wu, Weisheng Dong, and Guangming Shi. MetaIQA: deep meta-learning for no-reference image quality assessment. In *CVPR*, pages 14143–14152, 2020. 6
- [43] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147*, 2023. 2
- [44] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3

A. Views Selection Algorithm

We detailed the proposed views selection algorithm in Algorithm 1, which is a greedy initialization local searching algorithm, as we select the set of nodes that have highest weight as the initial solution and iteratively swap the node in this solution to find the maximum $s^T \cdot \mathbf{w} - s^T \cdot \mathbf{E} \cdot s$. ultimately identifying K optimal elements as the selected views. We designed it as an optimization algorithm because the number of views making brute-force searching for views has a very high time complexity ($O(N^K)$). Therefore, a highly efficient optimization algorithm with low time complexity ($O(1)$) is needed to make this process feasible. We set the optimization to run for 1,000 iterations in our paper's case, ensuring that the entire algorithm can perform real-time computation.

Algorithm 1 Views selection optimization

Require: Weights \mathbf{w} ; Similarity \mathbf{E} ; Selected Number k
Ensure: Maximizing $s^T \cdot \mathbf{w} - s^T \cdot \mathbf{E} \cdot s$.

- 1: **Initialize:** Sort \mathbf{w} and select top k elements as *selected_w*
- 2: Set a binary vector s with same size of \mathbf{w}
- 3: Set the elements of s with *selected_w*'s position as 1, other as 0.
- 4: Compute initial *best_value* = $s^T \cdot \mathbf{w} - s^T \cdot \mathbf{E} \cdot s$
- 5: Set *max_iter* = 1000
- 6: **for** *iter* = 1 to *max_iter* **do**
- 7: Randomly pick $w \in \text{selected_}w$
- 8: Randomly pick $new_w \in (\mathbf{w} - \text{selected_}w)$
- 9: Swap w with new_w in *selected_w*
- 10: update s 's elements with *selected_w*
- 11: **Compute new value:**
- 12: $new_value = s^T \cdot \mathbf{w} - s^T \cdot \mathbf{E} \cdot s$
- 13: **Acceptance criteria:**
- 14: **if** $new_value > best_value$ **then**
- 15: Update *best_value* = *new_value*
- 16: **else**
- 17: Revert swap
- 18: **end if**
- 19: **end for**
- 20: **return** selection vector s

B. Camera Operations

Many of the operations in our paper such as determining the target view, are based on the changes of camera poses $P = [R|T]$ with its focal length f . The two main camera operations involved are changing the orientation and move closer.

Orientation. When the image coordinates (i, j) in a $h \times w$ of a point-of-interest is known, and we want to adjust the

camera's orientation so that this point is positioned at the center of the image, we can first change the rotation matrix R of the camera into the Euler angles $(\theta_x, \theta_y, \theta_z)$:

$$(\theta_x, \theta_y, \theta_z) = F_{R \rightarrow E}(R) \quad (6)$$

Then compute the angle at which this point appears with the camera orientation:

$$\begin{aligned} \Delta\theta_x &= \arctan\left(\frac{i - \frac{h}{2}}{f}\right), \\ \Delta\theta_y &= \arctan\left(\frac{j - \frac{w}{2}}{f}\right). \end{aligned} \quad (7)$$

And we can change the Euler angles of the camera orientation as:

$$\begin{cases} \theta'_x = \theta_x + \Delta\theta_x \\ \theta'_y = \theta_y + \Delta\theta_y \\ \theta'_z = \theta_z \end{cases} \quad (8)$$

Then change the Euler angles into the new rotation matrix:

$$R' = F_{E \rightarrow R}(\theta_x, \theta_y, \theta_z) \quad (9)$$

Thus, we can obtain the new camera pose with the point-of-interest as the image's center:

$$P' = [R'|T] \quad (10)$$

Move closer. Using a close-up factor $0 < \alpha_t < 1$ we can move the camera P closer to a target position P_{target} :

$$P'' = (1 - \alpha_t) \cdot P + \alpha_t \cdot P_{target} \quad (11)$$

C. Discount Factor in Observability

When we want to comprehensively evaluate the contents a view v_0 observe is visible to v_1 (which is the observability of v_0 under v_1) together with the content v_1 observe is visible to another view v_2 , the distance between v_1 and both v_0 , v_2 , will naturally impact the statistical results. As shown in Fig. 6, as the distance d_c increases, the sum result of area A_1 and A_2 becomes larger compared to views with a smaller d_c . We define a as the observability of v_0 under v_1 when v_1 is at the midpoint between v_0 and v_2 . The area sum A of A_1 and A_2 can then be defined as:

$$A = \left(\left(\frac{1}{1+d_c}\right)^2 + \left(\frac{1}{1-d_c}\right)^2\right) \cdot a \quad (12)$$

which can be simplify as:

$$\begin{aligned} A &= \frac{1}{\beta} \cdot a, \\ \text{where } \beta &= \frac{(1-d_c^2)^2}{2 \cdot (1+d_c^2)} \end{aligned} \quad (13)$$

Thus, if we want to eliminate the influence of distance on the statistical results in this situation, we need to multiply the observability results by a distance discount factor β .

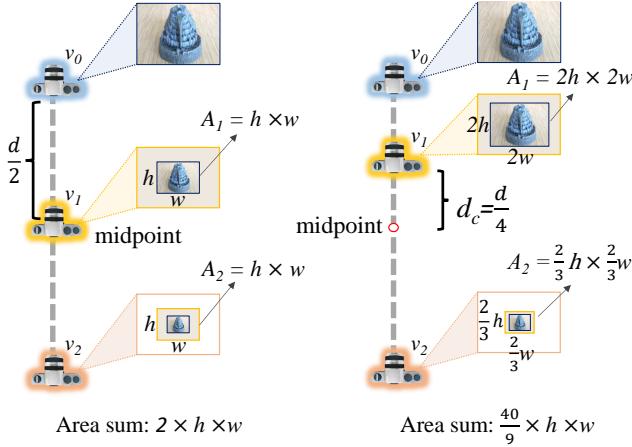


Figure 6. Distance factor in observability. Here we show two different positions of view v_1 , and the sum of the area A_1 which is the observability of view v_0 under view v_1 and A_2 which is the observability of view v_1 under view v_2 reaches its minimum when v_1 is positioned in the midpoint of v_1 and v_2 , and varies with the change of d_c which is the distance between v_1 and the midpoint.

D. Datasets Details

Extracted LERF dataset. For our experiments on the extracted LERF dataset, we select three scenes from LERF dataset following the setting outlined in the paper. The three scenes are: “donuts”, “shoe_rack”, “teatime”. For each scenes, we run COLMAP on all the views that selected to calculate their camera parameters, and separate training and testing views when training the Gaussian Splatting.

LLFF dataset. For our progressive close-up update experiments on the LLFF dataset, we select four scenes which are: “flower”, “fortress”, “orchids”, “t-rex”. We apply all the views as training views to train the initial Gaussian Splatting. For the progressive close-up, we select object-of-interest in these scenes, specifically, the stamen in “flower”, the building in “fortress”, the petal in “orchids”, the bone in “t-rex”. We then use these object-of-interest to move the training views closer at three close-up scales $3 \times$, $9 \times$ and $27 \times$, to obtain 5 target views at each close-up scale. These target views are used as testing views for each corresponding close-up scale.

E. Visualization

Refine process. We show two example outcomes from the refine process in Fig. 8. The combination of reliable pixels and enhanced images of to-be-updated views is used to update the Gaussian Splatting.

Ablation studies of Gaussian Splatting update. We show the visualization result of Gaussian Splatting update

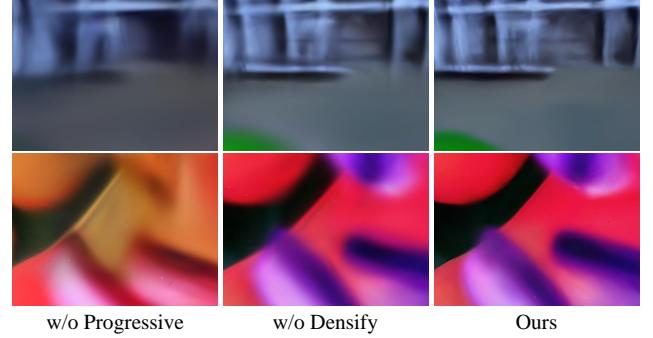


Figure 7. Visualization of Gaussian Splatting update.

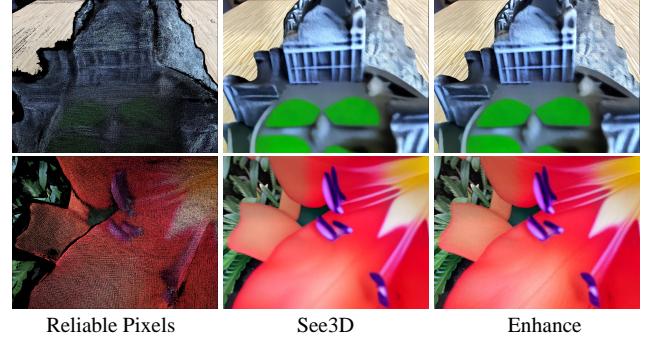


Figure 8. Refine process outcomes.

ablation studies in Fig. 7. And it can be observed that the update without progressive strategy fails to refine the close-up views, this demonstrates that each update round has its limitations, it is not feasible to achieve an extreme-scale close-up in a single round, and the progressive strategy ensures a robust updating process. Compared to updates without densify Gaussian primitives, our results exhibit greater clarity, highlighting the essential role of densifying the insufficient primitives under close-up views.