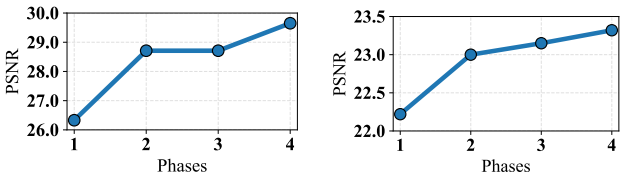
PDF-GS: Progressive Distractor Filtering for Robust 3D Gaussian Splatting

Supplementary Material

A. Additional Experiments

A.1. Phase-wise Reconstruction Quality.



(a) RobustNeRF dataset [20] (b) NeRF On-the-go dataset [18]

Figure A1. **Phase-wise evolution of reconstruction quality.** Both datasets show a gradual increase in reconstruction quality as the model advances through the phases of our progressive filtering framework, with improvements accumulating across successive phases.

In addition to evaluating the final performance for different numbers of filtering phases, we further analyze how reconstruction quality evolves during training by reporting the phase-wise PSNR averaged over all scenes in both the RobustNeRF [20] and NeRF On-the-go [18] datasets (Fig. A1).

Across both benchmarks, the PSNR consistently increases as the model progresses through successive phases. Each phase removes additional multi-view inconsistent signals, enabling cleaner supervision for the subsequent phases. Thus, the progressive design leads to cumulative quality improvements beyond single-pass optimization (i.e., training performed in one continuous optimization).

The first phase shows relatively low PSNR due to the use of a structure-oriented loss, which emphasizes local structures and reduces sensitivity to photometric cues affected by distractors. In addition, no mask is available at this stage and distractors have not yet been removed, resulting in considerably noisier supervision than in later phases. As training proceeds through the filtering phases, the masks and Gaussian representations are progressively refined, and by the final phase the model benefits from these improvements together with standard 3DGS objective, yielding the highest reconstruction quality.

A.2. Progressive Filtering on Single-Pass Baselines

We examine how our progressive filtering strategy can improve methods that perform distractor removal and reconstruction in a single optimization pass, using RobustSplat [4] as a representative example. While such single-pass approaches aim to complete filtering and reconstruction in one continuous process, our framework conducts these steps progressively across multiple filtering phases followed by a reconstruction phase. This difference motivates evaluating whether progressive filtering can enhance a method originally designed for a single-pass pipeline.

Table A1. Evaluation of applying our progressive filtering procedure to a single-pass baseline (RobustSplat [4]) on the RobustNeRF dataset [20]. We report results using both DINOv2 [15] and DINOv3 [22] backbones, and $*$ indicates our reproduced results.

Method	PSNR	SSIM	LPIPS
RobustSplat$*$ [4] (DINOv2)	29.24	0.89	0.13
+ Progressive Filtering (DINOv2)	29.32	0.90	0.14
RobustSplat$*$ [4] (DINOv3)	29.14	0.89	0.13
+ Progressive Filtering (DINOv3)	29.43	0.89	0.15

Specifically, we augment RobustSplat [4] by running its training loop for three successive phases while keeping its original architecture unchanged. Each phase is trained for 15k iterations, summing to 45k iterations in total, which is comparable to the 40k iterations used in our full framework. In each phase, we apply only the progressive filtering procedure, where phase-wise masks are computed from the discrepancy between the ground-truth images and the rendered training views of the preceding phase. Since RobustSplat [4] is designed to perform reconstruction within a single optimization pass, no additional reconstruction stage is introduced beyond these repeated optimization loops.

We observe this simple form of integration improves reconstruction quality (Tab. A1), indicating that our progressive filtering can strengthen the reconstruction process even when applied to methods originally designed for a single optimization pass.

A.3. Integration of Learned Mask Predictors

Table A2. Comparison of different masking strategies within our framework on the NeRF On-the-go dataset [18].

Masking Method	PSNR	SSIM	LPIPS
Discrepancy-based (Ours)	23.32	0.82	0.15
Learned predictor (RobustSplat)	23.39	0.82	0.14

Beyond augmenting existing baselines with our progressive filtering strategy, we also investigate a complementary direction that incorporates mechanisms from previous work into the mask generation step, using RobustSplat [4]

as a representative example. Our framework uses a simple discrepancy-based masking scheme in conjunction with mechanisms that leverage and amplify the inherent tendency of 3DGS to suppress view-inconsistent signals during optimization. By feeding progressively cleaner supervision back into subsequent phases, this mechanism reinforces the natural filtering behavior of 3DGS. Despite its simplicity, this masking approach achieves strong performance.

At the same time, our framework is compatible with more sophisticated masking strategies, including those that incorporate learned predictors. We therefore integrate the masking strategy of RobustSplat [4], which incorporates a learned MLP predictor, and use it as the masking component within our multi-phase pipeline. As shown in Tab. A2, employing RobustSplat's strategy within our framework further improves reconstruction quality, illustrating the complementary nature of our method and its compatibility with masking mechanisms developed in prior works.

A.4. Training Speed Comparison

Table A3. Wall clock training time comparison on the RobustNeRF [20] dataset.

Method	Android	Crab2	Statue	Yoda
RobustSplat [4]	21.6 min	24.5 min	28.7 min	24.2 min
Ours	25.2 min	23.5 min	28.1 min	23.9 min

We compare the wall clock training time of our method with RobustSplat [4] on the RobustNeRF [20] dataset, measuring end-to-end training time for each scene under the same hardware setting (Tab. A3). Although our method uses 40k optimization iterations, which is more than the 30k iterations in RobustSplat, the overall training time remains comparable. This is largely because RobustSplat [4] performs feature extraction at every iteration, whereas our method computes features only between phases. This per-phase design decouples feature extraction from the inner optimization loop, allowing it to be performed significantly less frequently and making it feasible to use heavier features or more advanced techniques. Exploring such extensions is an interesting direction for future work.

A.5. Effect on Different Masking Metrics.

We next study the effect of different feature transformations $F(\cdot)$ in Eq. 1 of the main paper for computing the discrepancy map between rendered and ground-truth images. This choice directly affects how distractor regions are localized and masked out.

As shown in Tab. A4, DINOv3 features yield the best overall performance (PSNR = 29.65 dB), outperforming earlier versions such as DINOv2 (29.42 dB) and simple low-level metrics like SSIM or PSNR. However, note that even when low-level metrics as PSNR and SSIM are employed (i.e., without any pretrained model), our method consistently surpasses vanilla 3DGS across all evaluation metrics. This confirms that the proposed progressive filtering strategy itself is intrinsically effective, while stronger feature representations such as DINOv3 further amplify its robustness and accuracy.

Table A4. Quantitative results with different masking metrics. † indicates our default setting. Our method with DINOv3 yields the best performance. Even when using low-level metrics such as PSNR and SSIM, our method outperforms vanilla 3DGS.

Method	PSNR	SSIM	LPIPS
DINOv3†	29.65	0.90	0.13
DINOv2	29.42	0.89	0.14
SSIM	28.88	0.89	0.13
PSNR	28.71	0.89	0.14