# ANDERSON ACCELERATION OF TEST-TIME TRAINING FOR FEED-FORWARD 3D RECONSTRUCTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Feed-forward 3D reconstruction predicts a scene representation from a few input views in a single forward pass, enabling efficient reconstruction without scene-specific optimization. Despite their efficiency, these methods often suffer from limited reconstruction quality and generalization to unseen scenes. To address this limitation, recent approaches introduce test-time training (TTT), which leverages supervision from test-time observations to adapt the model to each target scene. However, this scene-specific adaptation substantially increases inference time and diminishes the efficiency advantage of feed-forward reconstruction. To this end, we propose AA-TTT, a test-time training framework that formulates TTT as a fixed-point iteration and accelerates the process via Anderson acceleration. By directly exploiting the TTT trajectory, AA-TTT is applicable to a wide range of feed-forward 3D reconstruction systems. Extensive experiments on four representative TTT-based reconstruction systems demonstrate that AA-TTT consistently preserves reconstruction quality while achieving an average 1.4× test-time training speed across diverse reconstruction frameworks.

## 1 INTRODUCTION

In recent years, feed-forward 3D reconstruction has received considerable attention for its ability to predict scene representations directly from a small number of input images. Representative systems include DUSt3R (Wang et al., 2024), VGGT (Wang et al., 2025), pixelSplat (Charatan et al., 2024), and MVSplat (Chen et al., 2024). Unlike traditional per-scene optimization (Kerbl et al., 2023), these methods leverage pretrained models to infer geometry, appearance, or explicit 3D representations through a single forward pass, substantially reducing inference time.

Despite their efficiency, feed-forward predictions often remain limited by the geometric and appearance ambiguity caused by sparse observations. To improve reconstruction quality and generalization to unseen scenes, recent methods introduce test-time training (TTT) after the feed-forward stage, adapting the reconstruction to each target scene using supervision from test-time observations (Yuan et al., 2025; Zhou et al., 2026; Li et al., 2026a; An et al., 2026; Ren et al., 2026). These methods optimize scene-specific variables during test time, including prompts, adapters, extracted features, and explicit scene parameters. By adapting the reconstruction to each target scene, TTT improves reconstruction quality on unseen scenes.

Although effective, TTT requires iterative updates for each target scene and may require tens or even hundreds of updates for a single scene (Ren et al., 2026; An et al., 2026). This additional test-time training time diminishes the efficiency advantage of feed-forward reconstruction. This motivates a fundamental question: can test-time training be accelerated without compromising reconstruction quality?

To address this challenge, we propose AA-TTT. We observe that TTT procedures, despite optimizing different scene-specific variables, share a common iterative structure that can be naturally viewed as a fixed-point iteration. This perspective enables us to apply classical Anderson acceleration to TTT. By exploiting the history of previous TTT updates, AA-TTT accelerates test-time training while preserving reconstruction quality.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

AA-TTT operates on the shared TTT trajectory and is therefore compatible with diverse feed-forward reconstruction systems. Extensive experiments across four representative TTT-based reconstruction systems verify that AA-TTT maintains reconstruction quality while accelerating test-time training.

## 2 Related Work

**Feed-forward 3D reconstruction.** Traditional neural scene reconstruction methods perform scene-specific optimization for each input scene, such as neural radiance fields or 3D Gaussian representations (Mildenhall et al., 2020; Kerbl et al., 2023). To avoid expensive scene-specific optimization at inference, feed-forward reconstruction amortizes this process through offline training and directly predicts scene representations from sparse observations. Representative methods include DUSt3R and VGGT for geometry-aware reconstruction (Wang et al., 2024; 2025), as well as Gaussian-based approaches such as pixelSplat, MVSplat, and DepthSplat (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025). While these methods provide efficient single-pass reconstruction, their predictions are constrained by the training distribution and may contain scene-specific ambiguities. Recent works therefore explore recurrent refinement or optimization-aware initialization to further improve reconstruction quality (Xu et al., 2026; Chen et al., 2026; Liu et al., 2026; Li et al., 2026b).

**Test-time training for 3D reconstruction.** Test-time training improves a model at inference time by optimizing selected parameters using signals available from the test input, such as self-supervision or prediction entropy (Sun et al., 2020; Wang et al., 2021). This paradigm has also been introduced into feed-forward 3D reconstruction, where scene-specific variables are optimized using supervision derived from the test-time observations (Yuan et al., 2025; Zhou et al., 2026; Li et al., 2026a; An et al., 2026; Ren et al., 2026). Such test-time optimization enables feed-forward reconstruction systems to further resolve scene-specific ambiguities beyond their initial amortized predictions. TCO performs TTT on multiview reconstruction systems by adapting network adapters using prediction consistency together with optional geometric priors (Zhou et al., 2026). QuerySplat optimizes extracted features (Li et al., 2026a), while C3G and TokenGS refine explicit Gaussian parameters and implicit Gaussian tokens, respectively (An et al., 2026; Ren et al., 2026). Despite these different parameterizations, all of these methods rely on iterative scene-specific optimization using supervision from test-time observations. This shared optimization structure motivates a general acceleration mechanism that is not tied to a particular reconstruction representation.

**Anderson acceleration.** Anderson acceleration (AA) accelerates fixed-point iterations by constructing extrapolated updates from recent iterates or residuals (Anderson, 1965; Walker & Ni, 2011). It has been applied to various optimization problems, including stochastic optimization, minimax learning, and constrained optimization (Wei et al., 2021; He et al., 2022; Mai & Johansson, 2019), while stabilization techniques have been developed to improve robustness under nonlinear or noisy dynamics (Scieur et al., 2018; Barré et al., 2022; Lupo Pasini et al., 2021). Recent studies have also explored the application of AA to deep learning, where stabilization techniques are introduced to address stochasticity and improve convergence robustness during neural network training (Lupo Pasini et al., 2021). However, existing applications of AA mainly focus on accelerating general optimization procedures or designing problem-specific acceleration methods, while its potential for accelerating test-time training trajectories in feed-forward reconstruction remains unexplored.

## 3 Preliminaries and Background

### 3.1 Test-Time Training in Feed-Forward Scene Reconstruction

Feed-forward scene reconstruction provides an efficient paradigm for building 3D scenes: a pretrained model directly predicts geometry, appearance, or explicit scene representations from a small set of input views through a single forward pass. Representative systems include DUSt3R (Wang et al., 2024), VGGT (Wang et al., 2025), pixelSplat (Charatan et al., 2024), and MVSplat (Chen et al., 2024). Compared with traditional per-scene optimization (Kerbl et al., 2023), this feed-forward prediction substantially reduces inference cost.

To further improve reconstruction quality, recent methods augment feed-forward prediction with test-time training (TTT) (Sun et al., 2020; Yuan et al., 2025; Zhou et al., 2026; An et al., 2026; Ren

et al., 2026). These methods further refine the initial feed-forward reconstruction by optimizing scene-specific variables with supervision derived from the test-time observations. During this process, the reconstructed scene is rendered from the estimated representation and compared with the available observations. The resulting reconstruction errors provide training signals that guide the update of scene-specific variables. Through this scene-specific adaptation, TTT reduces ambiguities caused by sparse observations and improves reconstruction quality on the target scene.

Given the input observations $\mathcal{D}_{\text{in}}$, let $z$ denote the scene-specific variables adapted during TTT. Depending on the reconstruction system, these scene-specific variables take different forms. For example, QuerySplat (Li et al., 2026a) adapts extracted features, TCO optimizes network adapters, C3G refines explicit Gaussian parameters, and TokenGS updates implicit Gaussian tokens. A feed-forward reconstruction model first predicts an initial scene representation from the input observations. This prediction also provides the initialization for the scene-specific variables optimized during TTT. We denote this initialization as $z_0$.

The TTT objective depends on the reconstruction system. For example, TokenGS optimizes scene tokens by minimizing the discrepancy between rendered images and input observations. In general, the objective can be written as

$$\min_{z} \mathcal{L}_{\text{TTT}}(z; \mathcal{D}_{\text{in}}), \tag{1}$$

where $\mathcal{L}_{\text{TTT}}$ denotes the method-specific test-time training objective. This formulation provides a unified view of TTT across different reconstruction architectures.

### 3.2 Anderson Acceleration

A fixed-point problem is a fundamental class of problems in numerical optimization. Given an operator $g$, the goal is to find a solution $w^\star$ that satisfies

$$w^\star = g(w^\star). \tag{2}$$

Many iterative methods solve such problems by repeatedly applying the operator:

$$w_{t+1} = g(w_t). \tag{3}$$

Although fixed-point iterations are simple and widely applicable, their convergence can be slow when the update operator provides limited progress near the fixed point.

Anderson acceleration is a classical technique for addressing the slow convergence of fixed-point iterations. It accelerates the iteration by incorporating information from previous iterates and their residuals (Anderson, 1965; Walker & Ni, 2011). Instead of relying only on the current state, it combines a short history of iterates and residuals to estimate a more effective update direction. For the iterate $w_t$, we denote its fixed-point residual by

$$f_t = g(w_t) - w_t. \tag{4}$$

Given the most recent $p$ residuals, Anderson acceleration forms the residual matrix

$$F_t = [f_{t-p+1}, \ldots, f_t]. \tag{5}$$

The combination coefficients are obtained by solving the constrained least-squares problem

$$\boldsymbol{\alpha}_t = \arg\min_{\boldsymbol{\alpha}} \|F_t \boldsymbol{\alpha}\|_2^2 \quad \text{s.t.} \quad \sum_i \alpha_i = 1. \tag{6}$$

These coefficients are then used to combine previous fixed-point updates and construct an accelerated iterate:

$$w_{t+1}^{\mathrm{AA}} = \sum_{i=1}^{p} \alpha_{t,i} g(w_{t-p+i}). \tag{7}$$

By exploiting recent trajectory information, Anderson acceleration constructs extrapolated updates to reduce fixed-point residuals, providing a principled way to accelerate iterative procedures beyond the original update rule.

# 4 AA-TTT: Anderson Acceleration of Test-Time Training

Given a feed-forward 3D reconstruction system equipped with test-time training (TTT), our goal is to accelerate the test-time training process while preserving reconstruction quality.

Although reconstruction architectures and optimized variables differ across TTT procedures, they share an iterative update process toward a scene-specific solution. We interpret this process as a fixed-point iteration and apply Anderson acceleration, a classical technique for accelerating fixed-point iterations. However, directly applying Anderson acceleration may occasionally produce unreliable updates in practical reconstruction systems. To improve robustness, we introduce a simple safeguard strategy that preserves the original TTT objective.

## 4.1 Test-Time Training as Fixed-Point Iteration

Given input views $I$ and their camera information $c$, a feed-forward 3D reconstruction model first predicts an initial scene representation through a single forward pass,

$$y_0 = F_\phi(I, c), \tag{8}$$

where $F_\phi$ denotes the pretrained feed-forward reconstruction network parameterized by $\phi$, and $y_0$ is the initial scene representation predicted from the input views.

This feed-forward prediction provides an initialization for subsequent test-time training, where a method-specific set of variables is optimized using supervision available at test time,

$$\min_{z} \ \mathcal{L}_{\mathrm{TTT}}(z; \mathcal{D}_{\mathrm{in}}). \tag{9}$$

Following the general formulation of TTT, we denote its scene-specific optimization state by $z$, whose components may correspond to network adapters, extracted features, explicit Gaussian parameters, or implicit Gaussian tokens. Here $\mathcal{D}_{\mathrm{in}}$ denotes the test-time observations, and $\mathcal{L}_{\mathrm{TTT}}$ denotes the method-specific TTT objective.

Despite differences in reconstruction architectures and optimized variables, these TTT procedures share a common optimization structure: scene-specific variables are iteratively updated based on gradients of a test-time objective. For clarity, we illustrate this shared structure with a basic gradient update. Given a step size $\eta > 0$, the illustrative update takes the form

$$z_{k+1} = z_k - \eta \nabla_z \mathcal{L}_{\mathrm{TTT}}(z_k; \mathcal{D}_{\mathrm{in}}). \tag{10}$$

Here, $\nabla_z \mathcal{L}_{\mathrm{TTT}}$ represents the iterative step. For a fixed test scene, $\mathcal{D}_{\mathrm{in}}$ remains unchanged during TTT iterations. We therefore omit it below and write

$$z_{k+1} = z_k - \eta \nabla_z \mathcal{L}_{\mathrm{TTT}}(z_k). \tag{11}$$

Equation 11 illustrates the shared structure using a basic gradient update. We denote this complete native update by $\mathcal{T}$:

$$z_{k+1} = \mathcal{T}(z_k). \tag{12}$$

The entire TTT procedure is therefore a repeated application of the native update map:

$$z_{k+1} = \mathcal{T}(z_k), \qquad z_0 \xrightarrow{\mathcal{T}} z_1 \xrightarrow{\mathcal{T}} \cdots \xrightarrow{\mathcal{T}} z_K. \tag{13}$$

The specific form of $z$, the test-time objective, and the optimizer may vary across reconstruction systems, while the iterative formulation above remains unchanged. This formulation reveals the connection between TTT and fixed-point iteration. In the basic gradient case, a stationary point $z^\star$ of the TTT objective satisfies

$$\nabla_z \mathcal{L}_{\mathrm{TTT}}(z^\star) = 0 \quad \Longleftrightarrow \quad z^\star - \eta \nabla_z \mathcal{L}_{\mathrm{TTT}}(z^\star) = z^\star. \tag{14}$$

More generally, a converged state of the complete native update satisfies $\mathcal{T}(z^\star) = z^\star$. Thus, optimizing the TTT objective can be viewed as finding a fixed point of its native update operator $\mathcal{T}$.

### 4.2 Anderson-Based Training Prediction

Anderson acceleration is a classical method for accelerating fixed-point iterations. It improves convergence by incorporating information from previous iterates and their residuals.

Following this principle, we apply Anderson acceleration to the TTT update process. For the update operator $\mathcal{T}$, we define the residual at iteration $i$ as

$$r_i = \mathcal{T}(z_i) - z_i. \tag{15}$$

The recent residuals provide historical information about the local behavior of the TTT updates. We exploit information from the recent TTT trajectory to accelerate test-time training. Given the latest $M$ residuals, Anderson acceleration computes combination coefficients by solving

$$\alpha_k = \arg\min_{\mathbf{1}^\top \alpha = 1} \left\| \sum_i \alpha_i r_i \right\|_2^2. \tag{16}$$

The resulting coefficients are used to combine historical residuals and construct an accelerated update. The predicted TTT update is then obtained as

$$\hat{z}_{k+1} = \sum_i \alpha_{k,i} \mathcal{T}(z_i) = \sum_i \alpha_{k,i}(z_i + r_i). \tag{17}$$

We present the affine form of Anderson acceleration for clarity. In implementation, we use its equivalent difference-form realization; their equivalence is shown in Appendix B.

Different variables optimized during TTT may have different scales. We therefore normalize residuals across parameter blocks before solving the Anderson coefficient problem and apply the resulting coefficients to the original residuals.

### 4.3 Stabilizing Accelerated Updates

Under mild conditions, Anderson acceleration converges when initialized sufficiently close to a solution (Toth et al., 2017; Toth & Kelley, 2015). In contrast, when the iterates are far from the solution, Anderson acceleration can produce unstable extrapolations, leading to slow convergence or stagnation away from the desired solution (Potra & Engler, 2013; Walker & Ni, 2011).

Although TTT in feed-forward reconstruction starts from an initial scene estimate provided by the feed-forward model rather than an arbitrary initialization, Anderson acceleration may still produce overly aggressive extrapolations in some cases, especially when historical updates do not accurately represent the local update behavior. Such extrapolations may lead to unstable TTT updates.

To improve robustness, we introduce a simple safeguard strategy. Given an Anderson prediction $\hat{z}_{k+1}$, we evaluate its TTT objective and compare it with the most recent objective evaluated before the

**Algorithm 1** AA-TTT training loop

**Require:** initial state $z_0 = F_\phi(I)$; TTT steps $K$; history depth $M$; warm-up $W$

$\mathcal{D} \leftarrow \emptyset$ ▷ AA history
$\bar{z} \leftarrow \emptyset$ ▷ latest TTT update
$\ell_{\text{ref}} \leftarrow \emptyset$ ▷ loss before the last AA prediction
**for** $k = 0, 1, \ldots, K-1$ **do**
  $\ell_k \leftarrow \mathcal{L}_{\text{TTT}}(z_k)$
  **if** $\ell_{\text{ref}} \neq \emptyset$ **then**
    **if** $\ell_k > \ell_{\text{ref}}$ **then**
      $z_k \leftarrow \bar{z}$; $\mathcal{D} \leftarrow \emptyset$; $\ell_{\text{ref}} \leftarrow \emptyset$
      **continue**
    **end if**
    $\ell_{\text{ref}} \leftarrow \emptyset$
  **end if**
  $\bar{z} \leftarrow \mathcal{T}(z_k)$; $d_k \leftarrow \bar{z} - z_k$
  $m_k \leftarrow \min\{M, |\mathcal{D}|\}$
  **if** $k < W$ or $m_k = 0$ **then**
    $z_{k+1} \leftarrow \bar{z}$
  **else**
    $\hat{z}_{k+1} \leftarrow \text{Anderson}(\bar{z}, \mathcal{D})$
    $z_{k+1} \leftarrow \hat{z}_{k+1}$
    $\ell_{\text{ref}} \leftarrow \ell_k$
  **end if**
  $\mathcal{D} \leftarrow \text{Update}(\mathcal{D}, (z_k, d_k), M)$
**end for**
**return** reconstructed scene decoded from $z_K$

prediction. Let $\ell_k = \mathcal{L}_{\text{TTT}}(z_k)$ denote this reference objective. The Anderson prediction is accepted only when

$$\mathcal{L}_{\text{TTT}}(\hat{z}_{k+1}) \leq \ell_k. \tag{18}$$

Otherwise, we discard the prediction, restore the most recent update state. This safeguard prevents unreliable extrapolation while preserving the original TTT objective.

### 4.4 Choice of History Depth

The history depth $M$ determines the amount of previous update information used for Anderson prediction. A short history may provide insufficient trajectory information to characterize the local update behavior. A larger history allows Anderson acceleration to incorporate more information when estimating the local behavior of the TTT update process. However, excessively long histories may include outdated updates and increase the computational cost of solving the coefficient problem.

In practice, we observe that a small history size provides a good trade-off between acceleration performance and robustness. For TokenGS, we use $M = 5$ by default.

The complete procedure is summarized in Algorithm 1.

## 5 Analysis

### 5.1 Local Linear Structure of Feed-Forward TTT

Unlike traditional optimization methods, a distinctive property of test-time training in feed-forward reconstruction is that it does not start from an arbitrary initialization. The feed-forward model provides an initial scene estimate, and subsequent TTT updates further adapt this estimate toward the target-scene solution.

Let $\mathcal{T}$ denote the native TTT update introduced in Section 4.1. As shown in Section 4.1, optimizing the TTT objective amounts to finding a fixed point of the native update map. We denote this solution as $z^\star$, which satisfies

$$\mathcal{T}(z^\star) = z^\star. \tag{19}$$

We define the training error at iteration $k$ as

$$e_k = z_k - z^\star. \tag{20}$$

A Taylor expansion of the native TTT update around the converged solution $z^\star$ gives

$$\mathcal{T}(z_k) = \mathcal{T}(z^\star) + J_\star(z_k - z^\star) + R_k, \qquad J_\star = \nabla_z \mathcal{T}(z^\star), \tag{21}$$

where $R_k$ denotes the higher-order nonlinear terms. Under the local smoothness of $\mathcal{T}$, the remainder satisfies

$$\|R_k\| \le \frac{L_J}{2}\|z_k - z^\star\|^2 = \frac{L_J}{2}\|e_k\|^2. \tag{22}$$

Using $\mathcal{T}(z^\star) = z^\star$ and $z_{k+1} = \mathcal{T}(z_k)$, we obtain

$$e_{k+1} = J_\star e_k + R_k. \tag{23}$$

When the feed-forward initialization lies sufficiently close to the fixed point, the TTT trajectory remains in the local regime. As the trajectory approaches the fixed point, the higher-order remainder becomes progressively smaller relative to the first-order term. Consequently, the TTT dynamics are increasingly governed by the local linear operator $J_\star$,

$$e_{k+1} \approx J_\star e_k. \tag{24}$$

This local linear structure is particularly suitable for Anderson acceleration. For linear fixed-point systems, Anderson acceleration is closely related to GMRES and exploits residual history to construct minimum-residual updates over the corresponding Krylov subspace (Walker & Ni, 2011). Therefore, once the TTT trajectory enters this local regime, its recent update history provides informative approximations of the underlying local dynamics, which motivates applying Anderson acceleration to predict future TTT updates.

## 6 Experiments

We evaluate AA-TTT across four representative TTT-based reconstruction systems to assess its applicability across systems with different scene-specific variables. We further compare the test-time training time required by AA-TTT and other TTT update methods at comparable reconstruction quality. Finally, we conduct ablation studies on the safeguard strategy and the Anderson history length to analyze the effectiveness of the proposed stabilization mechanism and design choices.

For each reconstruction system, we use its available pretrained model and TTT configuration, and keep the data, test-time objective, and evaluation setting identical between the baseline and AA-TTT. The baseline for each system is its default TTT configuration. We measure the test-time training time required by each method to reach a reconstruction quality comparable to that of the corresponding baseline.

### 6.1 Acceleration across Different Reconstruction Systems

We first evaluate the applicability of AA-TTT across different reconstruction methods. The four evaluated systems optimize substantially different scene-specific variables: TCO adapts LoRA adapters, QuerySplat optimizes extracted features, C3G refines explicit Gaussian parameters, and TokenGS updates implicit Gaussian tokens. Despite these differences, AA-TTT uses the same formulation by operating on their common iterative TTT structure.

Tables 1 and 2 summarize the results on four representative systems. Compared with the original TTT procedure, AA-TTT consistently reduces the required test-time training time while maintaining comparable reconstruction quality. Across the four systems, AA-TTT achieves an average $1.4\times$ speedup in test-time training while preserving or improving every reported quality metric. These results demonstrate the consistent effectiveness of AA-TTT across diverse TTT procedures and reconstruction frameworks.

Table 1: Results on TCO. Lower Acc and Comp are better, whereas higher NC is better. Speed is measured relative to the baseline.

| Method | Time (s)↓ | Speed↑ | Acc↓ | Comp↓ | NC↑ |
|---|---|---|---|---|---|
| Baseline | 98.52 | 1.00× | 0.0141 | 0.0153 | 0.8683 |
| AA-TTT | **67.66** | **1.46**× | **0.0138** | **0.0150** | **0.8755** |

Table 2: Results on QuerySplat, C3G, and TokenGS. Higher PSNR and SSIM are better, whereas lower LPIPS is better. Speed is measured relative to the corresponding baseline.

| System | Method | Time (s)↓ | Speed↑ | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|---|---|
| C3G | Baseline | 41.82 | 1.00× | 29.6240 | 0.9082 | 0.1561 |
| C3G | AA-TTT | **28.76** | **1.45**× | **29.9056** | **0.9115** | **0.1556** |
| TokenGS | Baseline | 14.10 | 1.00× | 24.6291 | 0.7935 | 0.2971 |
| TokenGS | AA-TTT | **8.86** | **1.59**× | **24.7090** | **0.7939** | **0.2950** |
| QuerySplat | Baseline | 122.40 | 1.00× | 21.2019 | 0.6327 | 0.2686 |
| QuerySplat | AA-TTT | **104.80** | **1.17**× | **21.2363** | **0.6343** | **0.2685** |

### 6.2 Quality-matched Comparison with TTT Update Methods

We compare AA-TTT with commonly used TTT update methods on TokenGS under a quality-matched setting on a subset of 40 scenes. For each alternative optimizer, the learning rate is tuned on a small set of scenes and then fixed for evaluation, with details provided in the appendix. We measure the test-time training time required to reach the default TokenGS reconstruction quality, while further requiring AA-TTT to achieve quality no worse than the compared methods.

Table 3: Quality-matched comparison of TTT update methods on TokenGS.

| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Time (s)↓ |
|---|---|---|---|---|
| Baseline | 25.3069 | 0.8045 | 0.2815 | 14.31 |
| AA-TTT | **25.4528** | **0.8107** | **0.2728** | **9.79** |
| AdamW | 25.4088 | 0.8073 | 0.2773 | 10.14 |
| Prodigy | 25.4028 | 0.8068 | 0.2782 | 13.56 |
| RMSProp | 25.4324 | 0.8076 | 0.2737 | 14.54 |
| SignSGD | 25.3403 | 0.8059 | 0.2787 | 22.00 |
| SGD–Nesterov | 25.4237 | 0.8079 | 0.2777 | 36.05 |

#### 6.2.1 Effect of Safeguard Strategy

We evaluate AA-TTT with and without the safeguard strategy on TokenGS under the same adaptation setting to analyze its effect on reconstruction quality and test-time training efficiency.

The results show that removing the safeguard leads to worse reconstruction quality across all evaluation metrics while also requiring more test-time training time. In contrast, the safeguard achieves

consistently better reconstruction quality with less training time, demonstrating its effectiveness in improving both reconstruction quality and training efficiency.

Table 4: Ablation of the safeguard strategy on TokenGS. Reconstruction quality and test-time training time are compared with and without the safeguard.

| Safeguard | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Time (s) ↓ |
|---|---|---|---|---|
| w/o safeguard | 24.603 | 0.7884 | 0.3090 | 9.13 |
| w/ safeguard | **24.709** | **0.7939** | **0.2950** | **8.86** |

### 6.2.2 Effect of Anderson History Length

We evaluate different history lengths on TokenGS to identify an appropriate history size and analyze its impact on acceleration performance.

Table 5 reports reconstruction quality under different history lengths with identical acceleration settings. Each history-length setting is evaluated on a 50-scene subset with two repeated runs, i.e., 50 scenes $\times$ 2 repetitions. The results show that too short a history provides insufficient trajectory information, while too long a history may introduce irrelevant past updates and interfere with local estimation. A moderate history length therefore achieves the best performance.

Table 5: Sensitivity analysis of the Anderson history length $M$ on TokenGS. The reconstruction quality obtained with different history lengths is compared under the same adaptation setting. Each setting uses a 50-scene subset with two repeated runs.

| History depth $M$ | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| 2 | 24.819 | 0.7969 | 0.2950 |
| 3 | 24.843 | 0.7961 | 0.2948 |
| 5 | 25.012 | 0.8025 | 0.2853 |
| 7 | 24.697 | 0.7909 | 0.3011 |
| 9 | 24.655 | 0.7905 | 0.3003 |

## 7 Conclusion

We presented AA-TTT, an Anderson acceleration framework for test-time training in feed-forward 3D reconstruction. AA-TTT formulates TTT as a fixed-point iteration and exploits recent training history to accelerate the process, with a simple safeguard for stable prediction. The formulation is applicable across diverse reconstruction systems with different optimized variables. Experiments on four reconstruction systems show that AA-TTT consistently reduces test-time training time while maintaining reconstruction quality.

## AI Use Statement

In this work, we used generative AI tools to assist with improving the readability and clarity of the manuscript, checking and refining LaTeX formatting, and organizing and formatting references. We also used generative AI tools for language editing, including grammar correction and polishing of written descriptions. We have not used generative AI tools to generate experimental results, develop the core methodology, or make scientific decisions. All AI-assisted content was carefully reviewed and verified by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

## REFERENCES


Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyunah Ko, Junsu Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Learning compact 3D representations from feed-forward novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 363–373, 2026.

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12 (4):547–560, 1965. doi: 10.1145/321296.321305.

Mathieu Barré, Adrien Taylor, and Alexandre d'Aspremont. Convergence of a constrained vector extrapolation scheme. *SIAM Journal on Mathematics of Data Science*, 4(3):979–1002, 2022. doi: 10.1137/21M1428030.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3D gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 2024.

Tianyu Chen, Wei Xiang, Kang Han, Yu Lu, Di Wu, Gaowen Liu, and Ramana Rao Kompella. GIFSplat: Generative prior-guided iterative feed-forward 3D gaussian splatting from sparse views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26031–26040, 2026.

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 *European Conference on Computer Vision*, 2024.

Huan He, Shifan Zhao, Yuanzhe Xi, Joyce C. Ho, and Yousef Saad. GDA-AM: On the effectiveness of solving min-imax optimization via anderson mixing. In *International Conference on Learning Representations*, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Yinglong Li, Donghui Shen, Xiaoyu Zhang, Zhichao Ye, Hongyu Wu, Aimin Hao, Guofeng Zhang, and Haomin Liu. Querysplat: Decoupling geometry and appearance representations in 3DGS prediction. *arXiv preprint arXiv:2608.01186*, 2026a.

Yuke Li, Weihang Liu, Cheng Zhang, Yuefeng Zhang, Jiadi Cui, Zixuan Wang, Junran Ding, Haoyu Wu, Yujiao Shi, Jingyi Yu, and Xin Lou. Foresplat: Optimization-aware foresight for feed-forward 3D gaussian splatting. *arXiv preprint arXiv:2605.22020*, 2026b.

Yueh-Cheng Liu, Jozef Hladký, Matthias Nießner, and Angela Dai. Diff3r: Feed-forward 3D gaussian splatting with uncertainty-aware differentiable optimization. *arXiv preprint arXiv:2604.01030*, 2026.

Massimiliano Lupo Pasini, Junqi Yin, Viktor Reshniak, and Miroslav Stoyanov. Stable anderson acceleration for deep learning. *arXiv preprint arXiv:2110.14813*, 2021.

Vien V. Mai and Mikael Johansson. Nonlinear acceleration of constrained optimization algorithms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4903–4907, 2019. doi: 10.1109/ICASSP.2019.8682962.

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 *European Conference on Computer Vision*, pp. 405–421, 2020.

Florian A. Potra and Hans Engler. A characterization of the behavior of the anderson acceleration on linear problems. *Linear Algebra and its Applications*, 438(3):1002–1011, 2013. doi: 10.1016/j.laa.2012.09.008.

Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. TokenGS: Decoupling 3D gaussian prediction from pixels with learnable tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15365–15375, 2026.

Damien Scieur, Edouard Oyallon, Alexandre d'Aspremont, and Francis Bach. Online regularized nonlinear acceleration. *arXiv preprint arXiv:1805.09639*, 2018.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Alex Toth and C. T. Kelley. Convergence analysis for anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015. doi: 10.1137/130919398.

Alex Toth, J. Austin Ellis, Tom Evans, Steven Hamilton, C. T. Kelley, Roger Pawlowski, and Stuart Slattery. Local improvement results for anderson acceleration with inaccurate function evaluations. *SIAM Journal on Scientific Computing*, 39(5):S47–S65, 2017. doi: 10.1137/16M1080677.

Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011. doi: 10.1137/10078356X.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, 2024.

Fuchao Wei, Chenglong Bao, and Yang Liu. Stochastic anderson mixing for nonconvex stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22995–23008, 2021.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16453–16463, 2025.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. ReSplat: Learning recurrent gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2026.

Yuheng Yuan, Qiuhong Shen, Shizun Wang, Xingyi Yang, and Xinchao Wang. Test3R: Learning to reconstruct 3D at test time. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3360. URL https://openreview.net/forum?id=UcKYgtOIuX.

Lei Zhou, Haoyu Wu, Akshat Dave, and Dimitris Samaras. Learning 3D reconstruction with priors in test time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36550–36560, 2026.

## A Learning-rate tuning.

For each alternative optimizer, we evaluate five learning rates around its default setting on a small set of scenes and select one for the full comparison. Table 6 reports the resulting reconstruction metrics. "–" indicates collapsed runs, and the best value for each metric is bolded.

Table 6: Learning-rate sensitivity of alternative TTT update methods on TokenGS. Five learning rates around the default setting are evaluated for each method.

| Method | LR | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|
| AdamW | $10^{-2}$ | 21.522 | 0.6265 | 0.4671 |
| | $10^{-3}$ | 25.683 | 0.7902 | 0.2758 |
| | $10^{-4}$ | 25.573 | 0.7925 | 0.2825 |
| | $3\times10^{-3}$ | 24.333 | 0.7529 | 0.3201 |
| | $3\times10^{-4}$ | **25.788** | **0.8022** | **0.2710** |
| Prodigy | 0.5 | **25.793** | 0.7924 | 0.2769 |
| | 1 | 25.726 | **0.7965** | **0.2760** |
| | 2 | 25.688 | 0.7951 | 0.2768 |
| | 4 | 25.611 | 0.7965 | 0.2781 |
| | 8 | 25.644 | 0.7954 | 0.2789 |
| SignSGD | $10^{-3}$ | 22.663 | 0.7078 | 0.3227 |
| | $10^{-4}$ | **25.270** | **0.7865** | 0.2896 |
| | $3\times10^{-3}$ | 20.834 | 0.6034 | 0.4116 |
| | $3\times10^{-4}$ | 24.873 | 0.7788 | **0.2881** |
| | $3\times10^{-5}$ | 25.045 | 0.7780 | 0.2988 |
| SGD–Nesterov | $10^{-1}$ | 24.844 | 0.7730 | 0.3031 |
| | $10^{-2}$ | 24.388 | 0.7596 | 0.3137 |
| | $10^{-3}$ | 24.225 | 0.7519 | 0.3163 |
| | $3\times10^{-1}$ | **25.014** | **0.7799** | **0.2982** |
| | $5\times10^{-2}$ | 24.647 | 0.7670 | 0.3074 |
| RMSProp | $10^{-2}$ | – | – | – |
| | $10^{-3}$ | 21.596 | 0.6326 | 0.4044 |
| | $10^{-4}$ | **25.483** | **0.7832** | **0.2811** |
| | $3\times10^{-3}$ | 19.243 | 0.5126 | 0.5444 |
| | $3\times10^{-4}$ | 24.773 | 0.7616 | 0.2868 |

## B Equivalence Between the Affine and Difference Forms of Anderson Acceleration

In the main text, we present Anderson acceleration in its affine-combination form for clarity. Here, we briefly show its equivalent difference-form representation used in our implementation.

Given the most recent $m+1$ iterates $\{z_{k-m}, \dots, z_k\}$ and residuals

$$r_i = \mathcal{T}(z_i) - z_i, \tag{25}$$

the affine form computes coefficients $\alpha$ satisfying $\mathbf{1}^\top \alpha = 1$ and forms the accelerated iterate as

$$z_{k+1}^{\mathrm{AA}} = \sum_{j=0}^{m} \alpha_j \mathcal{T}(z_{k-m+j}). \tag{26}$$

Define the iterate and residual differences as

$$\Delta Z_k = [z_{k-m+1} - z_{k-m}, \dots, z_k - z_{k-1}], \tag{27}$$

and

$$\Delta R_k = [r_{k-m+1} - r_{k-m}, \dots, r_k - r_{k-1}]. \tag{28}$$

Under the change of variables between the affine coefficients $\alpha$ and difference coefficients $\gamma$, the residual combination can be rewritten as

$$\sum_{j=0}^{m} \alpha_j r_{k-m+j} = r_k + \Delta R_k \gamma, \tag{29}$$

while the iterate combination becomes

$$\sum_{j=0}^{m} \alpha_j z_{k-m+j} = z_k + \Delta Z_k \gamma. \tag{30}$$

Substituting these expressions into Equation 26 gives the equivalent difference form

$$z_{k+1}^{\mathrm{AA}} = z_k + r_k + (\Delta Z_k + \Delta R_k)\gamma. \tag{31}$$

The corresponding coefficients can therefore be obtained from the unconstrained least-squares problem

$$\gamma_k = \arg\min_{\gamma} \|r_k + \Delta R_k \gamma\|_2^2 . \tag{32}$$

Thus, the affine-combination and difference forms represent the same Anderson acceleration update. We use the difference form in implementation because it converts the constrained affine coefficient problem into a small unconstrained least-squares system over recent TTT updates, making it more convenient to solve in practice.

## C HARDWARE AND SOFTWARE ENVIRONMENT.

All experiments are run on a single NVIDIA GeForce RTX 4090 GPU (24 GB). Our implementation uses Python 3.11.15 and PyTorch 2.6.0 with CUDA 12.4, running on Ubuntu 20.04.6.