

OnlineSplatter: Pose-Free Online 3D Reconstruction for Free-Moving Objects

Mark He Huang^{1,3}, Lin Geng Foo², Christian Theobalt², Ying Sun³, De Wen Soh¹

¹Singapore University of Technology and Design, Singapore

²Max Planck Institute for Informatics, Saarland Informatics Campus, Germany

³Institute for Infocomm Research (I²R) & Centre for Frontier AI Research, A*STAR, Singapore
he_huang@mymail.sutd.edu.sg

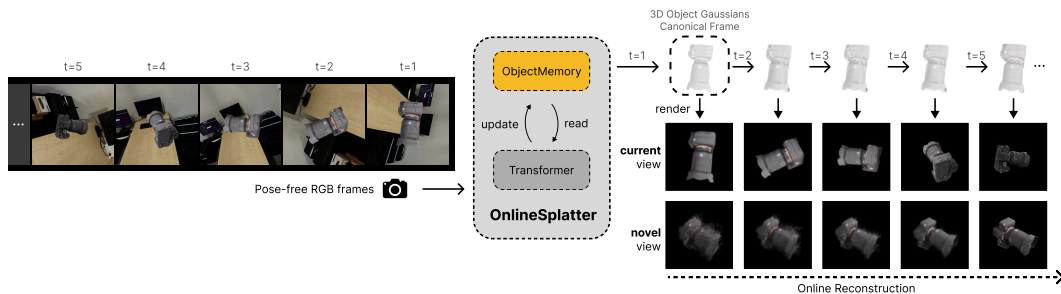


Figure 1: Outline of proposed **OnlineSplatter**. From the incoming stream of pose-free RGB frames, OnlineSplatter "splats" the observations into a canonical cloud of 3D Gaussians anchored to the first frame. Every new frame triggers a single, $O(1)$ **memory-and-time** update that immediately improves the reconstruction of the object. The system copes seamlessly with **freely moving objects** and requires neither pre-computed poses nor depth maps. The result is a continually improving 3D representation suitable for real-time applications.

Abstract

Free-moving object reconstruction from monocular video remains challenging, particularly without reliable pose or depth cues and under arbitrary object motion. We introduce OnlineSplatter, a novel online feed-forward framework generating high-quality, object-centric 3D Gaussians directly from RGB frames without requiring camera pose, depth priors, or bundle optimization. Our approach anchors reconstruction using the first frame and progressively refines the object representation through a dense Gaussian primitive field, maintaining constant computational cost regardless of video sequence length. Our core contribution is a dual-key memory module combining latent appearance-geometry keys with explicit directional keys, robustly fusing current frame features with temporally aggregated object states. This design enables effective handling of free-moving objects via spatial-guided memory readout and an efficient sparsification mechanism, ensuring comprehensive yet compact object coverage. Evaluations on real-world datasets demonstrate that OnlineSplatter significantly outperforms state-of-the-art pose-free reconstruction baselines, consistently improving with more observations while maintaining constant memory and runtime. Project page: markhh.com/OnlineSplatter

1 Introduction

Real-time 3D reconstruction of freely moving objects from monocular video remains a fundamental challenge in computer vision, with far-reaching implications for robotics, augmented reality, and interactive 3D applications [31, 33, 34, 45, 41, 53]. Although recent work achieves impressive results on static scenes, real-world deployments are often more demanding: objects move freely, undergoing arbitrary rotations and translations while a moving camera observes them. This dynamic setting violates the static-scene assumptions, reliable pose or depth cues, that underpin most existing methods. The challenge is especially acute in an online setting, where systems must update their understanding of objects as every new frame arrives, a capability essential for autonomous robots and AR devices operating in unpredictable environments.

Recent advances in 3D object reconstruction follow two main paradigms. The first leverages diffusion-based generative models and large-scale object-level priors [11, 23, 36, 22, 21, 25, 38] to synthesize plausible 3D assets from single or a few images. While generating visually compelling geometry, these methods rely on their learned priors to hallucinate unseen parts of the object. Although beneficial for 3D asset generation, this approach limits their applicability in real-time perception. The second paradigm comprises pointmap-based feed-forward methods [42, 8, 24, 48, 39, 2, 51] that directly regress pixel-aligned pointmaps from unposed images. Although effective for stationary scenes and accurate surface recovery, they falter when confronted with objects undergoing unrestricted motion.

A common thread among existing approaches is their reliance on camera poses, depth information, or global optimization [20, 9, 31, 44]. Such limitations not only restrict their applicability in online settings but also hinder their performance in environments where additional sensor data is unavailable.

Motivated by these challenges, we propose *OnlineSplatter*, a feed-forward framework for online reconstruction of freely moving objects. Anchoring reconstruction in the canonical coordinate system defined by the first frame, our network predicts a dense field of per-pixel Gaussian primitives and refines the object model causally as new RGB frames arrive (Fig. 1). Moreover, to control memory growth as observations accumulate, we propose an attention-based memory module that fuses incoming frame features with a compact latent state, eliminating the overhead of bundle adjustment or additional data processing. Central to this design is a dual-key memory retrieval strategy that combines appearance-geometry features with explicit spatial cues, providing robust temporal fusion and comprehensive spatial coverage.

Our contributions are: (i) a novel feed-forward framework for object-centric online 3D reconstruction that operates on monocular RGB streams in real-time, eliminating the need for camera poses, depth priors, or global optimization while maintaining constant computational complexity regardless of sequence length; (ii) a dual-key memory module that pairs latent features with spatial cues for efficient spatial-temporal fusion and principled sparsification; (iii) extensive experiments on GSO [7] and HO3D [10] demonstrate state-of-the-art performance on freely moving objects, paving the way for practical online reconstruction in dynamic environments.

2 Related Works

Pose-free 3D Reconstruction. Eliminating camera pose as input remains a key challenge in 3D reconstruction. Classical pipelines such as COLMAP [30] recover structure and poses via incremental SfM with MVS. Recent methods like 3D Gaussian Splatting [16], when combined with joint-pose optimization [14, 12, 17], can handle unposed images but still require static scenes. These optimization-based approaches fail to reconstruct moving objects due to their reliance on cross-frame correspondences, even when object masks are available. In contrast, our approach processes each frame causally in a feed-forward manner, eliminating the need for cross-frame optimization while handling freely moving objects. Recent feed-forward methods like DUS3R [42] predict aligned pointmaps from pairs of unposed images within a shared coordinate space and merge them via global alignment. While this avoids explicit optimization, DUS3R and its variants (e.g., Spann3R [39], VGGT [40], etc. [8, 51]) still assume static scenes and treat moving objects as outliers due to low cross-frame consistency. Their implicit reliance on large background surfaces further limits their applicability in dynamic, online reconstruction environments. Our method differs by focusing solely on object reconstruction without requiring background information, making it suitable for online reconstruction of freely moving objects.

Optimization-based Object Reconstruction. Optimization-based methods typically require global bundle adjustment across all frames, which poses fundamental limitations for real-time applications. Early approaches such as BARF [20] employ a coarse-to-fine registration strategy to jointly optimize shape and pose for NeRF using unposed RGB frames. Similarly, Hampali *et al.* [9] propose a splitting-and-merging strategy tailored for in-hand object scanning, enabling reconstruction of freely moving objects. However, both methods rely on global bundle optimization over all frames, rendering them unsuitable for online, causal settings. Fmov [31] leverages a virtual camera model to narrow the camera pose search space during initial joint 3D shape and pose optimization, followed by global optimization across frames. BundleSDF [44] utilizes a keypoint matching network and explicit frame selection strategy to optimize an object-centric neural signed distance field in a causal manner. While it can handle freely moving objects, it requires ground truth depth information as input during optimization. In contrast, our method employs an attention-based memory module that combines relative pose and latent features for purely feed-forward reconstruction, eliminating the need for depth input and computationally expensive cross-frame optimization.

Few-view Object Reconstruction. Recent advances in few-view reconstruction focus on generating 3D assets from limited views. Earlier methods like FvOR [50] combine learned object pose priors with alternating pose-shape optimization to reconstruct unknown objects from just a few images. FORGE [13] proposes a framework that jointly infers relative camera poses and fuses per-view 3D voxel features into a neural radiance field, achieving category-agnostic object reconstruction. However, these methods implicitly assume fully visible, centered objects and struggle with occlusions, sub-optimal observations, or dense view sequences. More recent large-scale data-driven approaches [36, 11, 54, 35, 46, 38, 37, 25, 56, 47, 23, 22] generate plausible geometry but typically focus on single-shot reconstruction and do not incorporate temporal observations for incremental refinement. In contrast, our framework continuously updates its representation as new frames arrive, yielding high-fidelity reconstructions that improve over time while remaining real-time and pose-free.

3 Method

The goal of our method is to perform an online reconstruction of a freely moving rigid object using monocular RGB images without relying on known camera poses or any object prior (such as depth, shape, or category). The term "online" implies that our approach processes incoming data causally, updating the reconstructed object representation incrementally as new frames become available. Fig. 2 shows an overview of our proposed OnlineSplatter framework. In the following sections, we elaborate on the details of our framework’s pipeline (Sec. 3.1), the design of our dual-key 3D Object Memory (Sec. 3.2), and the training and inference procedure for our model (Sec. 3.3).

3.1 OnlineSplatter Pipeline

Image Encoding. At each timestep t , we observe an RGB view V_t . While most reconstruction methods [42] implicitly rely on static scenes and background surfaces for stability, our approach explicitly handles freely moving objects in isolation. To focus on the object of interest, we leverage an off-the-shelf video object segmentation model XMem [3] to obtain the object mask M_t , and apply it to each frame to remove the background. The masked frame V'_t is then encoded into patch features via a hybrid strategy that concatenates two complementary encoders:

$$f_{vt} = \text{Concat}(\text{Encoder}_1^I(V'_t), \text{Encoder}_2^I(V'_t)) \quad (1)$$

where Encoder_1^I is a frozen DINO backbone [1, 28, 32], providing strong self-supervised appearance cues. However, DINO alone lacks the 3D awareness required for accurate reconstruction. We therefore introduce a learnable counterpart, Encoder_2^I , which adopts the same architecture but is trained end-to-end within OnlineSplatter to capture complementary geometric cues. This dual-encoder design enables our model to leverage both rich visual priors and geometry-aware features.

OnlineSplatter Transformer. After encoding the input image at each timestep, these features are input into our proposed OnlineSplatter Transformer, which is designed to process three distinct types of tokenized inputs: *reference-view tokens* $\mathbf{T}_{ref}^{in}$ (encoded from the initial frame V_0), *source-view tokens* $\mathbf{T}_{src,t}^{in}$ (encoded from the current frame V_t), and *memory-readout tokens* $\mathbf{T}_{mem,t}^{in}$ (readout from the object memory at time t). To differentiate and contextualize these tokens, we introduce

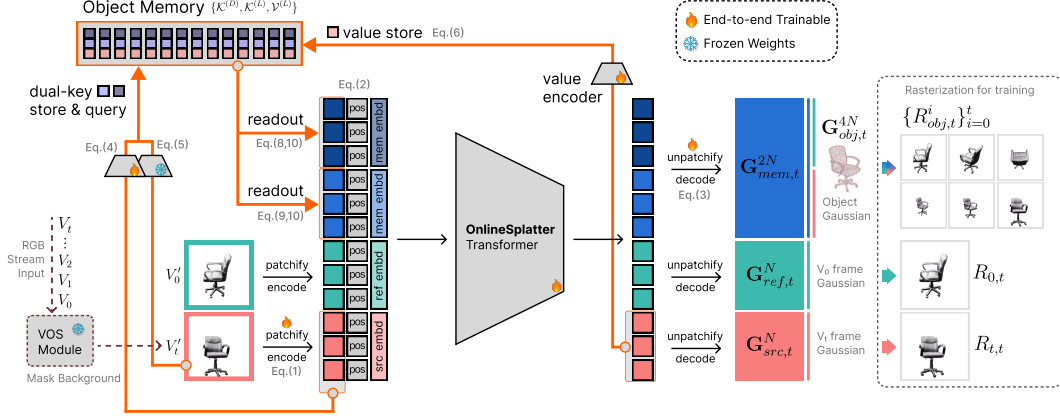


Figure 2: Overview of *OnlineSplatter* Pipeline. The input to our framework consists of a stream of RGB images $\{V_t\}_{t=0}^N$, where object masks $\{M_t\}_{t=0}^N$ are generated and applied to remove background on-the-fly using an off-the-shelf online video segmentation (OVS) module running alongside our framework. At each timestep t , *OnlineSplatter* processes the input frame V_t by first patchifying it into patch tokens. These tokens are then fed into a transformer-based architecture, which directly reasons and outputs pixel-aligned 3D Gaussian representations in a canonical space. Central to our method is object memory, an implicit module based on cross-attention, which is queried and updated at every timestep. This memory enables the incremental reconstruction of the object, consistently refining the object representation ($G_{obj,t}^{4N}$) as new observations arrive in a fully feed-forward manner.

corresponding learnable embeddings to each of the tokenized features. These embeddings are added with their respective encoded features and positional embeddings prior to transformer processing.

$$\mathbf{T}_{view}^{in} = f_{view} + f_{pos}^{emb} + f_{view}^{emb}, \quad \forall view \in \{ref, src, mem\} \quad (2)$$

Gaussian Decoding. The three types of token $\{\mathbf{T}_{ref}^{in}, \mathbf{T}_{src,t}^{in}, \mathbf{T}_{mem,t}^{in}\}$ are processed jointly by our unified *OnlineSplatter* Transformer that produces the output tokens $\{\mathbf{T}_{ref}^{out}, \mathbf{T}_{src,t}^{out}, \mathbf{T}_{mem,t}^{out}\}$, which are then decoded into 3D Gaussian parameters through a trainable unpatchifier.

$$\mathbf{G}_{obj,t}^{4N} := \{\mathbf{G}_{mem,t}^{2N}, \mathbf{G}_{ref,t}^N, \mathbf{G}_{src,t}^N\} = \text{Unpatchify}(\{\mathbf{T}_{mem,t}^{out}, \mathbf{T}_{v_0}^{out}, \mathbf{T}_{v_t}^{out}\}) \quad (3)$$

where each Gaussian set $\mathbf{G}^K$ consists of K Gaussian primitives defined as: $\{\boldsymbol{\mu}_k, \mathbf{r}_k, \mathbf{s}_k, \mathbf{c}_k, o_k\}_{k=1}^K$ with $N = H \times W$, corresponding to N pixels per RGB input $V \in \mathbb{R}^{H \times W \times 3}$. Note that, instead of predicting 3D primitives only for the newly observed frame at each timestep and accumulating predictions over time, our method fetches current understanding of the object from the memory and refines it with the latest observation. Consequently, our approach avoids accumulation or any explicit global aggregation operations entirely, which is advantageous for object-centric reconstruction as large observation overlapping is expected over time. Thus, at each timestep, the decoded Gaussian ($\mathbf{G}_{obj,t}^{4N}$) collectively form the updated representation of the object.

Rasterization. During training, we employ a differentiable rasterizer to render images for photometric supervision. Specifically, at each timestep, given the predicted Gaussian representation ($\mathbf{G}_{obj,t}^{4N}$) and the ground truth poses up to time t , we render $t + 1$ images $\{R_{obj,t}^i\}_{i=0}^t$. In addition, we render the frame-level subsets $\mathbf{G}_{ref,t}^N$ and $\mathbf{G}_{src,t}^N$ using the poses of V_0 and V_t , respectively, producing $R_{0,t}$ and $R_{t,t}$. These additional renders are critical for training stability, as they encourage each Gaussian subgroup to specialize in reconstructing the portion of the object visible in the corresponding input view. This design, in turn, incentivizes the memory tokens to encode features that are most useful for reconstructing a complete object representation.

3.2 Dual-Key 3D Object Memory

To enable object-centric online reconstruction, we need a memory bank that can store the most useful features of the object and can support producing a progressively better representation of the object at each timestep. Achieving this presents two primary challenges: (1) Naive accumulation of latent features in memory will result in an ever growing memory bank, which is computationally demanding

and memory inefficient as more frames are observed. (2) Naive accumulation of predictions in the output space will result in redundant overlapping predictions, which requires additional optimization steps to consolidate and simplify the representation.

To address these limitations, we propose a novel object-centric memory mechanism, Dual-Key 3D Object Memory, that consists of a key-value memory bank. Each memory entry consists of two keys and a value, where the two keys facilitate robust readout of the most relevant memory values, via our dual-key memory readout approach. In this subsection, we first discuss how we encode the memory into the memory bank, followed by how we read from the memory. Lastly, we also introduce our memory sparsification mechanism that helps keep the memory bank compact yet effective.

Memory Encoding Unlike conventional single-key memories, our method leverages two complementary keys—one *latent* key and one *direction* key. Intuitively, the latent key facilitates retrieval of information to guide visual-geometrical reasoning, while the direction key provides explicit spatial guidance to improve spatial coverage during memory reading. At every timestep t , we encode the latent key $\mathbf{k}_t^{(L)}$ from our tokenized features using a lightweight learnable key encoder (Encoder^K) to capture fundamental visual-geometrical cues:

$$\mathbf{k}_t^{(L)} := f_t^K = \text{Encoder}^K(f_{vt}) \quad (4)$$

We further leverage a lightweight pre-trained zero-shot 3D orientation estimator [43], denoted as Encoder^D , to encode our direction key $\mathbf{k}_t^{(D)}$. Specifically, $\mathbf{k}_t^{(D)}$ is computed based on the $\mathbb{R}^3$ axis-rotation of the object orientation $\{\theta_t, \phi_t, \gamma_t\}$:

$$\mathbf{k}_t^{(D)} := (\sin \phi_t \cos \theta_t, \sin \phi_t \sin \theta_t, \cos \phi_t) \mid \{\theta_t, \phi_t, \gamma_t\} = \text{Encoder}^D(V'_t) \quad (5)$$

where we use the Azimuth (θ_t) and Polar (ϕ_t) value from the prediction to convert to a unit directional vector as our directional key $\mathbf{k}_t^{(D)}$. Thus, $\mathbf{k}_t^{(L)}$ and $\mathbf{k}_t^{(D)}$ from Eq. 4 and Eq. 5 are keys used to retrieve the suitable memory values. Note that all keys and values in our object memory are at the token level, thus $\mathbf{k}_t^{(L)} \in \mathbb{R}^{p \times c}$ where p is number of patches per view, we also broadcast $\mathbf{k}_t^{(D)}$ from $\mathbb{R}^{1 \times 3}$ to $\mathbb{R}^{p \times 3}$ such that patches from the same view share the same directional key. Then, after OnlineSplatter produces output tokens at time t , we encode the output tokens to update the value $\mathbf{v}_t^{(L)}$ corresponding to the encoded keys in Eq. 4 and Eq. 5. Specifically, a trainable value encoder (defined as Encoder^V) takes output tokens $\mathbf{T}_{src,t}^{out}$ as input to produce the new value:

$$\mathbf{v}_t^{(L)} := f_t^V = \text{Encoder}^V(\mathbf{T}_{src,t}^{out}) \quad (6)$$

Lastly, our dual-key pairs with the encoded value form a new entry at time t : $(\mathbf{k}_t^{(L)}, \mathbf{k}_t^{(D)}, \mathbf{v}_t^{(L)})$. The stored in-memory dual-key-values as a whole are denoted as $\mathcal{K}^{(D)} \in \mathbb{R}^{(S \cdot P) \times 3}$ and $\mathcal{K}^{(L)}, \mathcal{V}^{(L)} \in \mathbb{R}^{(S \cdot P) \times c}$ where S represents the maximum size set for the object memory.

Spatial-Guided Dual-Key Memory Reading Our memory module maintains a large collection of tokens that encode different perspectives of the object’s appearance and geometry. However, using all memory tokens for every forward pass would be computationally expensive and potentially noisy. Instead, we need an efficient mechanism to select the most relevant memory features for reconstructing the object at each timestep.

While our latent key, derived from tokenized features through end-to-end training with 3D reasoning objectives, captures both visual and geometric information, relying solely on latent key-based attention may not be optimal for object-centric reconstruction. This is because our memory readout serves two critical purposes: (1) supporting prediction for newly observed regions, and (2) retrieving the most informative features for reconstructing the complete object. To address this dual objective, we introduce a directional key that provides explicit spatial guidance for memory readout.

To perform memory read from our Object Memory $\{\mathcal{K}^{(D)}, \mathcal{K}^{(L)}, \mathcal{V}^{(L)}\}$ at timestep t , we treat the current (at time t) encoded latent key as the querying latent query (i.e. $\mathbf{q}_t^{(L)} \leftarrow \mathbf{k}_t^{(L)}$). At the same time, to compute a direction query $\mathbf{q}_t^{(D)}$, we average the directions between the current view $\mathbf{k}_t^{(D)}$ and the reference view $\mathbf{k}_0^{(D)}$, which represents an orientation that is likely to already have good coverage:

$$\mathbf{q}_t^{(D)} \leftarrow \frac{\mathbf{k}_0^{(D)} + \mathbf{k}_t^{(D)}}{\|\mathbf{k}_0^{(D)} + \mathbf{k}_t^{(D)}\|} \quad (7)$$

At each timestep, we perform two complementary memory reading operations (Orientation-Aligned Read and Orientation-Complementary Read) to retrieve memory features $f_{mem,t}$ for our OnlineS-platter transformer to reason with the reference view and current source view.

— *Orientation-Aligned Read* emphasizes memory entries that closely match both the latent and directional query keys. The similarity measure ($s_{i,t}^{(align)}$) for the i -th memory entry is defined as:

$$s_{i,t}^{(align)} = (\mathbf{q}_t^{(L)\top} \mathbf{k}_i^{(L)}) \cdot \mathbf{q}_t^{(D)\top} \mathbf{k}_i^{(D)} \cdot \frac{1}{\tau_t} \quad (8)$$

where τ_t is a temperature coefficient to dampen potential inaccuracies produced by Encoder^D . We dynamically set $\tau_t = 2.5 - \sigma_t$, where $\sigma_t \in [0, 1]$ is the confidence value of [43] for observation V_t .

— *Orientation-Complementary Read* retrieves memory entries that closely match the latent key but significantly differ in orientation, thus capturing complementary viewpoints. The complementary similarity score $s_i^{(comp)}$ is computed as:

$$s_i^{(comp)} = (\mathbf{q}_t^{(L)\top} \mathbf{k}_i^{(L)}) \cdot (-\mathbf{q}_t^{(D)\top}) \mathbf{k}_i^{(D)} \cdot \frac{1}{\tau_t} \quad (9)$$

The final retrieved features are computed via attention-weighted sums:

$$f_{mem,t}^{(align)} = \sum_{i=1}^N \frac{\exp(s_i^{(align)})}{\sum_j \exp(s_j^{(align)})} \mathbf{v}_i, \quad f_{mem,t}^{(comp)} = \sum_{i=1}^N \frac{\exp(s_i^{(comp)})}{\sum_j \exp(s_j^{(comp)})} \mathbf{v}_i, \quad \text{where } i, j = 1, \dots, (S \cdot P) \quad (10)$$

Overall, our dual-key memory reading approach allows the memory readout to be guided by explicit spatial cues, encouraging spatial coverage on top of the visual-geometrical cues.

Memory Sparsification Mechanism Furthermore, we would like to maintain a bounded memory size while preserving good coverage of diverse viewpoints. To achieve this, inspired by attention-based memory design in [3, 39], we propose a memory sparsification strategy that leverages our dual key design. Specifically, when the memory reaches its max capacity S , we drop 20% of memory that is deemed least useful by considering two factors: (i) usage (cross-attention contribution), and (ii) spatial coverage (average angular distance to other entries). Specifically, for each memory entry i , we define the total usage U_i by averaging the accumulated cross-attention weight:

$$U_i = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} [\mathcal{A}_{i,t}^{(align)} + \mathcal{A}_{i,t}^{(comp)}], \quad (11)$$

where $\mathcal{T}_i$ is the set of timesteps in which entry i participated, and $\mathcal{A}_{i,t}^{(\cdot)} = \frac{\exp(s_i^{(\cdot)})}{\sum_j \exp(s_j^{(\cdot)})}$, i.e., $\mathcal{A}_{i,t}^{(\cdot)}$ denotes the normalized cross-attention weight derived from Eq. (10) for memory reads. Intuitively, U_i is larger if an entry has been repeatedly or strongly attended to, meaning that it is useful. Subsequently, we compute spatial coverage for each memory entry. Let $\mathbf{k}_i^{(D)} \in \mathbb{R}^3$ be the direction key for entry i . We define its coverage measure C_i as the *average* angular dot product to all other entries:

$$C_i = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^N \mathbf{k}_i^{(D)\top} \mathbf{k}_j^{(D)}, \quad (12)$$

Lastly, to prune object memory, we sort all entries by C_i and select the top 50% of entries as the *dense subset*, which represent viewpoints in memory that are spatially well covered. From this dense subset, we remove the 40% lowest-ranked entries with respect to usage U_i . Formally,

$$\text{PruneSubset} = \left\{ i \in \text{DenseSubset} \mid U_i \leq U_q \right\}, \quad (13)$$

where U_q is the usage value at the 40-th percentile within DenseSubset. These pruned entries each time collectively constitute 20% of the full memory cap, balancing between retaining unique coverage and discarding underused features.

3.3 Training and Inference

Staged Training. Our OnlineSplatter model is trained to optimize two complementary objectives: (1) learning relative object-camera pose relationships and predicting pixel-aligned 3D Gaussian

parameters, and (2) effectively utilizing our attention-based Object Memory to reason about the object in canonical coordinate space through memory encoding and reading. These objectives present a challenging optimization landscape, as the gradients for the second objective only become meaningful after the first objective reaches a certain level of convergence. To address this, we employ a two-stage training strategy: **(1) Warm-up Training:** We train the core reconstruction components without the Object Memory module. Specifically, we optimize the view encoder (Encoder_1^I), positional and view embeddings (f_{pos}^{emb} and f_{view}^{emb}), OnlineSplatter transformer, and unpatchify decoder in the first stage. **(2) Main Training:** We include the Object Memory module and train the entire network end-to-end, allowing the model to learn both reconstruction and memory simultaneously.

Training Loss. We employ a combination of both photometric and geometrical losses to train our model. First, the photometric loss $\mathcal{L}_{\text{photo}}$ minimizes the MSE between the ground truth images and rendered images from predicted 3D Gaussian parameters at ground truth camera poses, where $\mathcal{L}_{\text{photo}}$ is computed on object regions only. We include a background penalty term ($\mathcal{L}_{\text{bg}}$) in $\mathcal{L}_{\text{photo}}$ to penalize Gaussians’ color and opacity outside the object’s visual hull. While the photometric loss alone can theoretically supervise feedforward reconstruction tasks [51], geometrical supervision can often enhance convergence speed and reconstruction quality [18, 2, 14, 29]. Thus, we incorporate geometrical loss $\mathcal{L}_{\text{geo}}$, which includes a ray alignment component ($\mathcal{L}_{\text{ray}}$) to ensure that predicted 3D points lie along their corresponding camera rays, and a depth component ($\mathcal{L}_{\text{depth}}$) to minimize MSE between predicted and ground truth relative depths. The **overall training objective** is: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{photo}} + \lambda_g \mathcal{L}_{\text{geo}}$, where λ_g balances the contribution of geometrical supervision. We provide full details in the appendix.

Implementation Details. We train and evaluate our model on 256×256 resolution images. We use 8x A100 GPUs for 250K steps with a batch size of 64 in the *Warm-up Training* stage and 500K steps with a batch size of 16 in the *Main Training* stage. We sample 3-5 views per object during *warm-up* and 6-12 views per object during *main* training. We provide more details on implementation and hyperparameters in the appendix.

4 Experiments

This section evaluates our approach by outlining the evaluation protocol, describing the datasets for training and testing, comparing against state-of-the-art baselines, and conducting ablation studies to analyze each component’s impact.

4.1 Experimental settings

Evaluation Protocol To properly evaluate our online object reconstruction framework, we need to assess how well it performs at different stages of observation accumulation. This is crucial because real-world applications often require reliable reconstruction even with limited initial observations. We therefore design a stage-wise evaluation protocol that examines performance across three distinct phases: **1) Early Stage** ($\mathcal{T}_{\text{early}} := \{1 \leq t \leq 4\}$): Tests the model’s ability to quickly establish an initial object representation with minimal observations; **2) Mid Stage** ($\mathcal{T}_{\text{mid}} := \{5 \leq t \leq 10\}$): Evaluates how well the model refines its reconstruction as more views become available; **3) Late Stage** ($\mathcal{T}_{\text{late}} := \{11 \leq t \leq T\}$): Assesses the model’s capability to maintain and improve reconstruction quality with extended observation sequences. For each test sequence of N frames $\{V_n\}_{n=1}^N$, we split the frames into two sets: **Input frames** ($\mathcal{V}_{\text{input}}$): A randomly sampled subset of $\frac{N}{2}$ frames used for input; **Target frames** ($\mathcal{V}_{\text{target}}$): The remaining $\frac{N}{2}$ frames reserved for NVS-based evaluation. Specifically, during evaluation, each frame V_t is provided to the model at each timestep $t \in \{1, \dots, T\}$ (where $T = \frac{N}{2}$) and we expect the model to output a 3D object representation $\hat{O}_t$ at each timestep t and produce novel view images $\hat{R}_{v,t}$ for all target viewpoints $v \in \mathcal{V}_{\text{target}}$. We then compare these renders against the corresponding ground-truth frames V_v . The stage-wise performance is then computed as: $\mathcal{M}_{\text{stage}} = \frac{1}{|\mathcal{T}_{\text{stage}}|} \sum_{t \in \mathcal{T}_{\text{stage}}} \sum_{v \in \mathcal{V}_{\text{target}}} \mathcal{M}(\hat{R}_{v,t}, V_v)$ where $\mathcal{M}$ represents standard image quality metrics used for 3D reconstruction [16, 51, 2]: PSNR, SSIM, and LPIPS.

Datasets. We train on 100K objects sampled from Objaverse [5, 4]. Unlike conventional few-view or image-to-3D setups that render from biased polar angles and fixed upright poses, our setting targets real-world freely moving object reconstruction, where objects appear in arbitrary poses relative to the camera. This difference is critical, as real-world data often includes partial views with unknown

GSO									
Method	Early-Stage			Mid-Stage			Late-Stage		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
FSO _{rand4}	21.358	0.861	0.177	21.919	<u>0.877</u>	0.181	21.737	0.855	0.181
FSO _{dist4}	22.365	0.874	<u>0.119</u>	<u>23.757</u>	0.862	<u>0.117</u>	23.751	0.873	<u>0.120</u>
NPS _{dist2}	22.986	0.859	<u>0.155</u>	23.050	0.863	<u>0.162</u>	22.949	<u>0.878</u>	<u>0.156</u>
NPS _{dist3}	<u>23.331</u>	0.862	0.149	23.206	0.861	0.138	<u>24.141</u>	0.863	0.125
Ours	26.329	0.921	0.084	27.553	0.933	0.066	31.737	0.969	0.075

HO3D									
Method	Early-Stage			Mid-Stage			Late-Stage		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
FSO _{rand4}	18.488	0.820	0.187	18.552	0.817	0.191	17.683	0.810	0.199
FSO _{dist4}	18.594	0.837	0.177	19.215	0.848	0.184	19.619	0.843	0.183
NPS _{dist2}	21.063	<u>0.855</u>	<u>0.160</u>	22.803	0.841	<u>0.158</u>	22.134	0.846	0.164
NPS _{dist3}	<u>21.134</u>	0.853	<u>0.162</u>	<u>22.967</u>	<u>0.869</u>	0.165	<u>22.947</u>	<u>0.860</u>	<u>0.163</u>
Ours	23.627	0.910	0.152	25.803	0.912	0.122	27.928	0.952	0.099

Table 1: Comparison of different baselines on two datasets. Results are shown for early-stage, mid-stage, and late-stage settings. Best results are **bolded** and second best results are underlined.

object centers, rendering naive pre-processing ineffective. To simulate such conditions, we develop a custom script (details in the appendix) that generates diverse trajectories with look-at jitter, varying focal lengths, and randomized lighting. Each object receives a unique trajectory, ensuring diverse motion patterns for training.

For evaluation, we use two datasets of unseen objects. First, we test on Google Scanned Objects (GSO) [7], rendering 36 frames per object using our training pipeline (each with distinct lighting and motion). Second, we assess generalization to real-world monocular videos with occlusions using the HO3D dataset [10], which contains hand-object interaction sequences.

Baselines. No prior feed-forward model supports pose-free, RGB-only reconstruction in online settings. We thus adapt two leading pose-free few-view methods, FreeSplatter [48] and NoPoSplat [51], to our online setting. Both consume uncalibrated RGB images and predict 3D Gaussians.

FreeSplatter-O [48] is a transformer-based, object-centric method trained on Objaverse. It jointly processes 4 pose-free views per sample. To adapt it online, we introduce two frame selection strategies for each timestep: (1) *rand4*: randomly selects 4 frames from past observations (**FSO_{rand4}**); (2) *dist4*: selects 4 frames with the largest feature differences using a DINO [1] encoder (**FSO_{dist4}**).

NoPoSplat [51] extends DUST3R [42] backbone with 3D Gaussian output heads and is fine-tuned on scene-level data with 2-3 input views. We adapt it for object-centric reconstruction by fine-tuning on Objaverse with object mask supervision, and apply the *dist* strategy to select 2 or 3 diverse frames, resulting in **NPS_{dist2}** and **NPS_{dist3}**.

4.2 Results

Table 1 compares our method against strong baselines on both GSO and HO3D datasets across different stages of observation accumulation. GSO offers statistical significance through its diverse set of unseen objects, while HO3D introduces real-world challenges, including complex interactions and occlusions. Across all metrics and stages, OnlineSplatter achieves superior performance—improving up to +7.596 PSNR and +0.106 SSIM on GSO, and +4.981 PSNR and +0.092 SSIM on HO3D in late-stage reconstruction. Two key insights emerge: good early-stage performance and temporal development. Even with fewer than four observations, OnlineSplatter significantly outperforms all baselines. Over time, a clear divergence appears. Baselines using explicit frame selection often exhibit unstable or stagnant performance. In contrast, OnlineSplatter consistently improves with more observations, as also shown qualitatively in Fig. 3, where our method delivers notably better visual quality and geometric accuracy from early to late stages. This underscores the strength of our Object Memory mechanism in leveraging temporal cues for progressive reconstruction refinement.

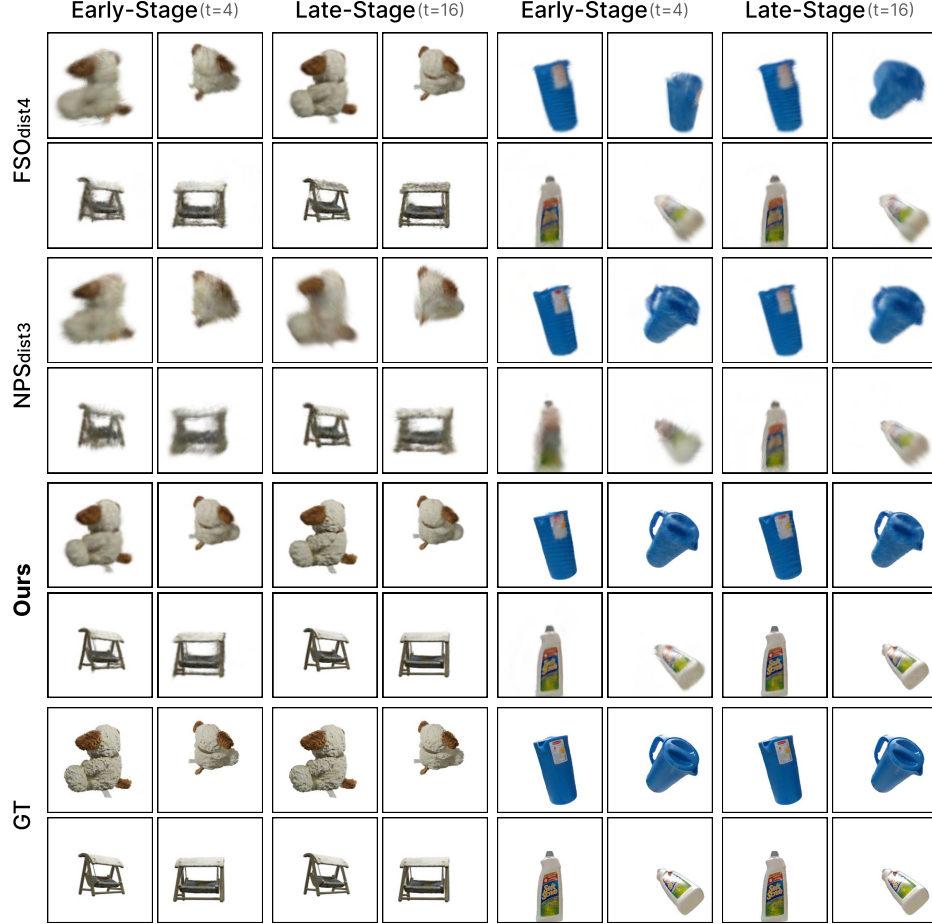


Figure 3: Qualitative results of different baselines and our method on the GSO (left) and HO3D (right) datasets. We visualize the results at inference timestep $t = 4$ and $t = 16$, which corresponds to the early-stage and late-stage settings, respectively. Our reconstructed outputs show significantly better visual quality and geometric accuracy as more observations become available.

Variants	Early-Stage $\mathcal{M}_{avg} \uparrow$	Mid-Stage $\mathcal{M}_{avg} \uparrow$	Late-Stage $\mathcal{M}_{avg} \uparrow$
Ours	0.699	0.734	0.810
w/o latent key	0.545	0.582	0.596
w/o direction key	0.699	0.701	0.723
w/ encode from gs	0.541	0.582	0.611
w/ random pruning	0.697	0.728	0.764

Table 2: Impact of dual-key object memory design. Results are reported on GSO dataset.

Variants	Early-Stage $\mathcal{M}_{avg} \uparrow$	Mid-Stage $\mathcal{M}_{avg} \uparrow$	Late-Stage $\mathcal{M}_{avg} \uparrow$
Ours	0.699	0.734	0.810
w/o staged training	0.545	0.582	0.588
w/o ray loss ($\mathcal{L}_{ray}$)	0.562	0.599	0.682
w/o bg penalty ($\mathcal{L}_{bg}$)	0.675	0.712	0.795
w/o depth loss ($\mathcal{L}_{depth}$)	0.691	0.728	0.805
w/ sequential sampling only	0.645	0.682	0.688
w/ random sampling only	0.697	0.728	0.764

Table 3: Impact of training strategy components. Results are reported on GSO dataset.

4.3 Ablations and Analysis

In this section, we ablate different components of our method and analyze the results. For better readability, we normalize and average metrics of PSNR, SSIM, and LPIPS to $[0, 1]$ and report the $\mathcal{M}_{avg}$ value, where a higher value indicates better performance, more details in the appendix.

Impact of Object Memory Design. We conduct ablation studies to validate our dual-key object memory design in Sec. 3.2, summarizing results on the GSO dataset in Table 2. Specifically: *Dual-key Design:* Removing the latent key severely degrades performance at all stages due to loss of visual-geometrical cues. Besides, removing the direction key significantly impacts later stages, highlighting its role in spatial coverage. *Memory Encoding:* Encoding memory from unpatchified

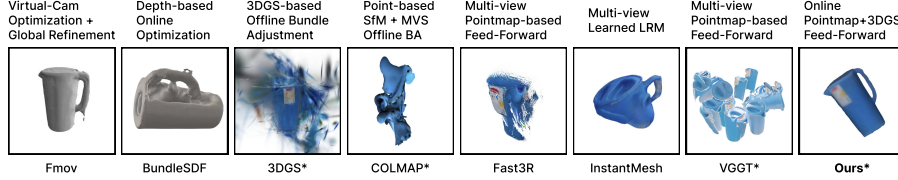


Figure 4: Visual comparison of mesh results between different methods. Methods marked with an asterisk (*) indicate that additional pre- or post-processing steps were applied to generate the visual results. More details in the appendix.

Gaussian parameters rather than direct Transformer token outputs leads to degraded performance, showing that using Transformer tokens for memory encoding better preserves learned representations. *Memory Sparsification*: Our sparsification strategy based on usage rate and spatial coverage outperforms random pruning, particularly in later stages, showing the efficacy of our pruning criteria.

Impact of Training Strategy. We evaluate our training strategy (discussed in Sec. 3.3) through ablation studies and show results in Table 3, demonstrating: *Staged Training*: Removing the two-stage training (warm-up followed by main training) by using a single stage only greatly lowers performance, validating our intuition that the memory module requires a well-initialized backbone. *Loss Components*: Removing the ray alignment ($\mathcal{L}_{\text{ray}}$) notably reduces convergence speed and stability, harming performance. Excluding the visual hull ($\mathcal{L}_{\text{bg}}$) penalty moderately degrades performance, underscoring its role in preserving object boundaries. Removing depth term ($\mathcal{L}_{\text{depth}}$) slightly impacts performance, as it primarily aids convergence during warm-up training. *Training Frame Sampling*: Our progressive sampling strategy (discussed in the appendix) outperforms sequential-only and random-only alternatives, underscoring the advantage of curriculum learning in our training.

Mesh Visual Comparison. To demonstrate our approach’s efficacy, we convert our final 3DGS representation into meshes and visually compare it comprehensively with state-of-the-art methods from different paradigms. We evaluate against representative methods: (1) those leveraging ground-truth depth information during optimization [44], (2) offline bundle adjustment techniques performing global optimization across frames [31, 16, 30], (3) diffusion-based approaches using learned 3D priors [47], and (4) recent feed-forward pointmap models operating offline [49, 40]. As shown in Fig. 4, despite being provided with object masks, many methods struggle to reconstruct freely moving objects. In contrast, our method achieves comparable quality to those requiring extensive optimization or additional depth supervision while retaining the benefits of a feed-forward, online framework.

5 Limitations and Future Work

Our current framework has some limitations that warrant attention. First, the framework outputs 3D Gaussian Splatting (3DGS) representations, which while efficient for object understanding and rendering, may not be directly suitable for certain downstream applications requiring explicit mesh representations. Converting 3DGS to meshes robustly remains challenging and is an active area of research. Future work could explore hybrid representations that maintain both rendering efficiency and mesh compatibility. Second, the framework’s performance may depend on the quality of the initial reference view. Poor initial observation, such as heavily occluded or blurry first frames, could impact subsequent reconstruction quality. Lastly, our framework is currently limited to rigid objects. Future work could explore modeling non-rigid objects and integrate it with downstream tasks like robotic manipulation.

6 Conclusion

We presented OnlineSplat, a novel framework for pose-free online 3D reconstruction of freely moving objects from monocular RGB video. Our dual-key memory module enables efficient temporal feature fusion with constant computational complexity. Extensive evaluation demonstrates superior reconstruction quality without requiring camera poses, depth priors, or global optimization. This makes our approach particularly suitable for robotics applications in dynamic environments requiring online continuous moving object perception and manipulation.

References

- [1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [2] 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*, pages 19457–19467, 2024.
- [3] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, pages 640–658. Springer, 2022.
- [4] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023.
- [5] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [7] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.
- [8] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024.
- [9] Shreyas Hampali, Tomas Hodan, Luan Tran, Lingni Ma, Cem Keskin, and Vincent Lepetit. In-hand 3d object scanning from an rgb sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17079–17088, 2023.
- [10] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3196–3206, 2020.
- [11] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.
- [12] Zhisheng Huang, Peng Wang, Jingdong Zhang, Yuan Liu, Xin Li, and Wenping Wang. 3r-gs: Best practice in optimizing camera poses along with 3dgs. *arXiv preprint arXiv:2504.04294*, 2025.
- [13] Hanwen Jiang, Zhenyu Jiang, Kristen Grauman, and Yuke Zhu. Few-view object reconstruction with unknown categories and camera poses. *International Conference on 3D Vision (3DV)*, 2024.
- [14] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. *arXiv preprint arXiv:2403.09413*, 2024.
- [15] Alexander Kaszynski. fast-simplification: Fast quadratic mesh simplification. <https://github.com/pyvista/fast-simplification>, 2025.
- [16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [17] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.
- [18] Lei Lan, Tianjia Shao, Zixuan Lu, Yu Zhang, Chenfanfu Jiang, and Yin Yang. 3dgs2: Near second-order converging 3d gaussian splatting. *arXiv preprint arXiv:2501.13975*, 2025.

- [19] Benjamin Lefaudeaux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. <https://github.com/facebookresearch/xformers>, 2022.
- [20] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5741–5751, 2021.
- [21] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10072–10083, 2024.
- [22] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024.
- [23] Ruoshi Liu, Rundui Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023.
- [24] Yuzheng Liu, Siyan Dong, Shuzhe Wang, Yingda Yin, Yanchao Yang, Qingnan Fan, and Baoquan Chen. Slam3r: Real-time dense scene reconstruction from monocular rgb videos. *arXiv preprint arXiv:2412.09401*, 2024.
- [25] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9970–9980, 2024.
- [26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [28] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023.
- [29] Hamza Pehlivan, Andrea Boscolo Camilletto, Lin Geng Foo, Marc Habermann, and Christian Theobalt. Second-order optimization of gaussian splats with importance sampling. *arXiv preprint arXiv:2504.12905*, 2025.
- [30] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [31] Haixin Shi, Yinlin Hu, Daniel Koguciuk, Juan-Ting Lin, Mathieu Salzmann, and David Ferstl. Free-moving object reconstruction and pose estimation with virtual camera. *arXiv preprint arXiv:2405.05858*, 2024.
- [32] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025.
- [33] Michael Strecke and Jorg Stuckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2019.
- [34] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021.
- [35] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024.

- [36] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024.
- [37] Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. Mvdifffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. In *European Conference on Computer Vision*, pages 175–191. Springer, 2024.
- [38] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2024.
- [39] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024.
- [40] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. *arXiv preprint arXiv:2503.11651*, 2025.
- [41] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025.
- [42] 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*, pages 20697–20709, 2024.
- [43] Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything: Learning robust object orientation estimation from rendering 3d models. *arXiv preprint arXiv:2412.18605*, 2024.
- [44] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Müller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–617, 2023.
- [45] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundation-stereo: Zero-shot stereo matching. *CVPR*, 2025.
- [46] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.
- [47] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.
- [48] Jiale Xu, Shenghua Gao, and Ying Shan. Freesplatter: Pose-free gaussian splatting for sparse-view 3d reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25442–25452, October 2025.
- [49] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025.
- [50] Zhenpei Yang, Zhile Ren, Miguel Angel Bautista, Zaiwei Zhang, Qi Shan, and Qixing Huang. Fvor: Robust joint shape and pose optimization for few-view object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2497–2507, 2022.
- [51] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.
- [52] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

- [53] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2410.03825*, 2024.
- [54] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024.
- [55] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.
- [56] Zixin Zou, Weihao Cheng, Yan-Pei Cao, Shi-Sheng Huang, Ying Shan, and Song-Hai Zhang. Sparse3d: Distilling multiview-consistent diffusion for object reconstruction from sparse views. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 7900–7908, 2024.

A Notations and Definitions

Please refer to Table 4 for the list of symbols and their definitions used in the paper.

B Additional Details on Training Losses

In Section 3.3 of the main paper, we introduced our losses used for training. Below, we give more details.

Photometric Supervision For photometric supervision, we employ a differentiable rasterizer to render 2D images from the predicted 3D Gaussian parameters ($\mathbf{G}_{obj,t}^{4N}$) and ground truth camera poses. The photometric loss $\mathcal{L}_{photo}$ consists of two components: $\mathcal{L}_{photo} = \mathcal{L}_{masked} + \lambda_{bg}\mathcal{L}_{bg}$, where $\mathcal{L}_{masked}$ is the masked MSE loss computed only on object regions:

$$\mathcal{L}_{masked} = \frac{1}{|\mathcal{S}_t|} \sum_{p \in \mathcal{S}_t} \|R_t(p) - V_t(p)\|_2^2 \quad (14)$$

Here, $\mathcal{S}_t$ denotes the set of pixels belonging to the object silhouette at time t , $R_t(p)$ and $V_t(p)$ respectively refer to the rendered color and ground truth color at pixel p at time t . The background penalty term $\mathcal{L}_{bg}$ penalizes Gaussians’s color and opacity outside the object’s visual hull:

$$\mathcal{L}_{bg} = \frac{1}{|\mathcal{G}_t|} \sum_{g \in \mathcal{G}_t} (\|c_g\|_2^2 + \alpha o_g) \quad (15)$$

where $\mathcal{G}_t \subset \mathbf{G}_{obj,t}^{4N}$ is the subset of predicted Gaussians that is outside of the object’s visual hull defined by object mask from reference view (M_0) and current view (M_t), o_g is the opacity of Gaussian g , and α is a weighting factor.

Geometrical Supervision While the photometric loss alone can theoretically supervise feedforward reconstruction tasks [51], we incorporate geometrical supervision to enhance convergence speed and reconstruction quality. This is motivated by prior work [18, 2, 14, 29] showing that Gaussian means significantly impact 3DGS convergence. To handle potential missing or noisy depth ground truth, we decompose the geometrical loss into two terms: $\mathcal{L}_{geo} = \mathcal{L}_{ray} + \lambda_d \mathcal{L}_{depth}$. The ray alignment loss $\mathcal{L}_{ray}$ ensures that predicted 3D points lie along their corresponding camera rays:

$$\mathcal{L}_{ray} = \frac{1}{|\mathcal{S}_t|} \sum_{p \in \mathcal{S}_t} (1 - r_p \cdot \hat{r}_p) \quad (16)$$

where r_p is the normalized predicted ray from camera center to pixel p . The normalized depth loss $\mathcal{L}_{depth}$ compares relative depths:

$$\mathcal{L}_{depth} = \frac{1}{|\mathcal{S}_t|} \sum_{p \in \mathcal{S}_t} \left\| \frac{d_p}{\bar{d}} - \frac{z_p}{\bar{z}} \right\|_2^2 \quad (17)$$

where d_p and z_p are predicted and ground truth depths, and $\bar{d}$ and $\bar{z}$ are their respective means.

Overall training objectives. The final training objective combines both photometric and geometrical losses: $\mathcal{L}_{total} = \mathcal{L}_{photo} + \lambda_g \mathcal{L}_{geo}$ where λ_g balances the contribution of geometrical supervision.

C Additional Analysis

C.1 Impact of Training Data Scaling and Selection.

While the complete Objaverse [5, 4] dataset contains over $10M$ objects, we conduct a systematic study of our method’s scalability under computational constraints by examining two critical factors: (1) the relationship between dataset size and model performance, scaling from $1K$ to $100K$ objects, and (2) the impact of data curation strategies on reconstruction quality. To evaluate these factors, we compare two sampling approaches: random selection from the full Objaverse dataset versus quality-based selection using aesthetic scores from [46]. All experiments are conducted with identical training configurations—500K training steps and consistent hyperparameters—to ensure a fair and controlled comparison.

Section	Symbol	Definition
Input and Output	V_t	RGB input frame at timestep t
	M_t	Object mask at timestep t for frame V_t
	V'_t	Masked RGB frame at timestep t with background removed
	$\hat{O}_t$	Predicted 3D object representation at timestep t
	$\hat{R}_{v,t}$	Rendered novel view image for viewpoint v at timestep t
Token Representations	$\mathbf{T}_{ref}^{in}$	Reference-view input tokens (from initial frame)
	$\mathbf{T}_{src,t}^{in}$	Source-view input tokens at timestep t
	$\mathbf{T}_{mem,t}^{in}$	Memory-readout input tokens at timestep t
	$\mathbf{T}_{view}^{in}$	Generic input tokens for view $\in \{ref, src, mem\}$
	$\mathbf{T}_{ref}^{out}$	Reference-view output tokens
	$\mathbf{T}_{src,t}^{out}$	Source-view output tokens at timestep t
	$\mathbf{T}_{mem,t}^{out}$	Memory-readout output tokens at timestep t
	f_{view}	Encoded features for a view $\in \{ref, src, mem\}$
	f_{pos}^{emb}	Positional embeddings
	f_{view}^{emb}	View-specific embeddings for view $\in \{ref, src, mem\}$
Gaussian Parameters	$\mathbf{G}_{obj,t}^{4N}$	Complete object Gaussian representation at timestep t
	$\mathbf{G}_{mem,t}^{2N}$	Memory-based subset of Gaussians at timestep t
	$\mathbf{G}_{ref,t}^N$	Reference-view subset of Gaussians at timestep t
	$\mathbf{G}_{src,t}^N$	Source-view subset of Gaussians at timestep t
	$\mathbf{G}^K$	Set of K Gaussian primitives
	μ_k	Mean position of k -th Gaussian
	$\mathbf{r}_k$	Rotation of k -th Gaussian
	$\mathbf{s}_k$	Scale of k -th Gaussian
	$\mathbf{c}_k$	Color / Spherical harmonics coefficients of k -th Gaussian
	o_k	Opacity of k -th Gaussian
Memory Module	$\mathbf{k}_t^{(L)}$	Latent key at timestep t
	$\mathbf{k}_t^{(D)}$	Direction key at timestep t
	$\mathbf{v}_t^{(L)}$	Memory value at timestep t
	$\mathcal{K}^{(D)}$	Set of stored directional keys
	$\mathcal{K}^{(L)}$	Set of stored latent keys
	$\mathcal{V}^{(L)}$	Set of stored memory values
	$\mathbf{q}_t^{(L)}$	Latent query at timestep t
	$\mathbf{q}_t^{(D)}$	Direction query at timestep t
	$s_{i,t}^{(align)}$	Alignment similarity score for entry i at timestep t
	$s_i^{(comp)}$	Complementary similarity score for entry i
	τ_t	Temperature coefficient at timestep t
	σ_t	Confidence value from orientation estimator
	U_i	Usage measure for memory entry i
	C_i	Coverage measure for memory entry i
	$\mathcal{T}_i$	Set of timesteps for memory entry i
	$\mathcal{A}_{i,t}^{(\cdot)}$	Normalized cross-attention weight
Evaluation Metrics	$\mathcal{M}$	Generic image quality metric (PSNR, SSIM, LPIPS)
	$\mathcal{M}_{stage}$	Stage-wise performance metric
	$\mathcal{M}_{avg}$	Normalized average of metrics
	$\mathcal{V}_{input}$	Set of input frames
	$\mathcal{V}_{target}$	Set of target frames
	$\mathcal{T}_{stage}$	Set of timesteps in a stage
Dimensions and Constants	N	Number of pixels per frame ($H \times W$)
	S	Maximum memory size
	P	Number of patches per view
	C	Feature dimension

Table 4: List of mathematical notations used throughout the paper.

As shown in Fig. 5, our analysis reveals two key findings. First, our method demonstrates strong scaling behavior, with performance continuing to improve as the dataset size increases to $100K$ objects without showing signs of saturation. This suggests significant potential for further gains with larger datasets. Second, we find that careful data curation through aesthetic quality filtering yields substantial performance improvements, indicating that strategic data selection can be an effective approach for optimizing model performance under limited computational resources.

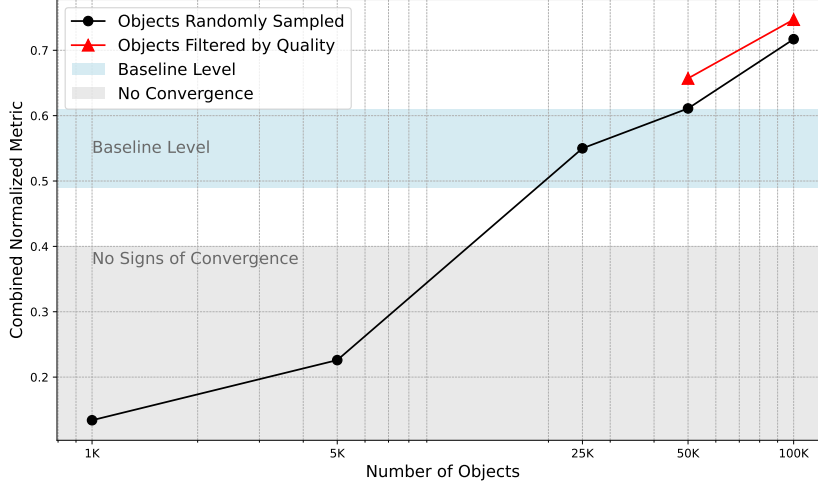


Figure 5: Impact of Training Data Quantity and Quality.

C.2 Impact of Ray Alignment Loss in Geometrical Supervision.

While photometric RGB-based loss can effectively supervise 3D Gaussian positions when they are already well-aligned with the ground truth and visible from the rendered pose, it struggles to guide Gaussians that are far from their optimal positions. This limitation becomes particularly problematic in object-centric reconstruction, where a significant portion of the region in observed frame consists of plain background regions that do not receive meaningful supervision. In such cases, predicted Gaussians may either float arbitrarily in space or be suppressed by the background penalty term ($\mathcal{L}_{bg}$), failing to contribute meaningfully to object reconstruction.

To address this challenge and accelerate convergence, we introduce a ray alignment term $\mathcal{L}_{ray}$ that explicitly regularizes the 3D Gaussian positions (as detailed in Sec. B). This term ensures that predicted Gaussians align with their corresponding camera rays in the object region, providing crucial geometric guidance even when photometric supervision is insufficient.

To demonstrate the effectiveness of the ray alignment loss, we conduct a comparative analysis between training with and without $\mathcal{L}_{ray}$, visualized in Fig. 6. The visualization shows two camera views from a training sample at different stages (1K-10K training steps). In each view, blue lines represent ground-truth per-pixel camera rays, while red lines indicate predicted 3D rays from the ground-truth camera center to the predicted Gaussian means. The comparison reveals two key findings:

- Without $\mathcal{L}_{ray}$, many Gaussians drift away from the object region, effectively becoming "dead" primitives that contribute little to reconstruction.
- With $\mathcal{L}_{ray}$, Gaussians maintain better alignment with ground-truth rays in object regions, and background Gaussians actively migrate toward object regions to participate in reconstruction.

These observations confirm that the ray alignment loss serves as an effective regularizer for 3D Gaussian positions and significantly improves convergence speed during training. The quantitative results also reflect this significant improvement (See Table 5).

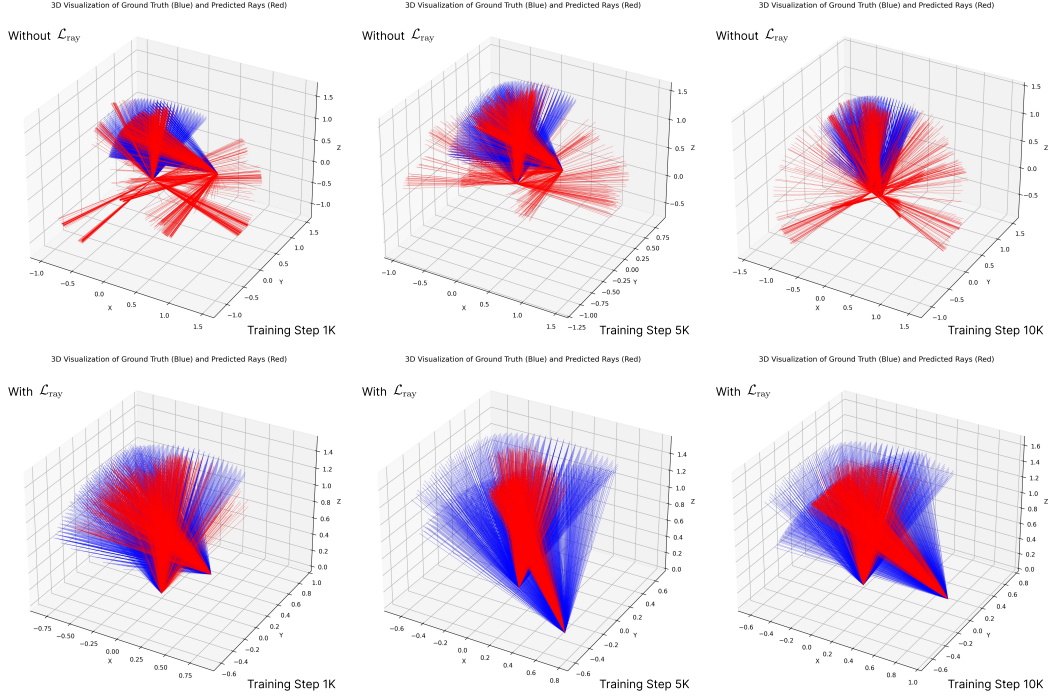


Figure 6: Visualization of the effect of without (top row) and with (bottom row) ray alignment loss $\mathcal{L}_{\text{ray}}$ over $1K - 10K$ training steps. The visualization shows both the ground-truth per-pixel camera rays (in blue) and the predicted 3D ray pointing from the ground-truth camera center to the predicted 3D point (in red). The qualitative visualization shows clearly that the ray alignment loss is effective in regularizing 3D gaussians positions and converges quickly.

C.3 Additional Notes on Ablation Study.

In the main paper, we use the averaged metrics ($\mathcal{M}_{\text{avg}}$) of PSNR, SSIM, and LPIPS to report results in ablation studies (Table 2 and Table 3) due to space limitation and better readability. Note that $\mathcal{M}_{\text{avg}}$ is computed as $\mathcal{M}_{\text{avg}} = \frac{1}{3} \left[\text{clip}\left(\frac{\text{PSNR}-20}{20}, 0, 1\right) + \text{SSIM} + \left(1 - \text{clip}\left(\frac{\text{LPIPS}}{0.6}, 0, 1\right)\right) \right]$ where we normalize PSNR, SSIM, and LPIPS to $[0, 1]$ by clipping and scaling such that a higher $\mathcal{M}_{\text{avg}}$ value indicates relatively better performance. Here in Table 5, we provide the detailed original results for completeness.

Impact of Dual-key Object Memory Design									
Method	Early-Stage			Mid-Stage			Late-Stage		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Ours	26.329	0.921	0.084	27.553	0.933	0.066	31.737	0.969	0.075
w/o latent key	23.260	0.764	0.176	23.947	0.805	0.153	24.212	0.822	0.147
w/o direction key	26.320	0.917	0.084	26.385	0.923	0.083	31.147	0.749	0.083
w encode from gs	23.237	0.759	0.179	23.983	0.804	0.154	24.538	0.837	0.138
w/ random pruning	26.261	0.919	0.086	27.089	0.927	0.080	31.443	0.852	0.080

Impact of Training Strategy Components									
Method	Early-Stage			Mid-Stage			Late-Stage		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Ours	26.329	0.921	0.084	27.553	0.933	0.066	31.737	0.969	0.075
w/o staged training	23.257	0.766	0.177	24.016	0.803	0.155	24.169	0.808	0.151
w/o ray loss	23.528	0.788	0.166	24.312	0.823	0.145	25.962	0.905	0.094
w/o bg penalty	25.844	0.896	0.097	26.562	0.937	0.077	31.714	0.949	0.091
w/o depth loss	26.147	0.917	0.091	28.301	0.921	0.150	31.643	0.960	0.077
w/ sequential sampling only	25.268	0.866	0.117	25.929	0.904	0.093	26.182	0.905	0.091
w/ random sampling only	26.249	0.920	0.085	27.587	0.923	0.097	31.695	0.906	0.119

Table 5: Expanded version of Table 2 and Table 3 in the Main Paper

D Additional Training and Implementation Details

In Section 3.3 of the main paper, due to space constraints, we provided only the most significant training and implementation details. Here, we provide comprehensive details on our training (Section D.1 and D.2) and implementation (Section D.3).

D.1 Progressive Training Frame Sampling.

Most methods that leverage monocular video data rely on implicit temporal continuity—i.e., minimal changes between consecutive frames—during training [39, 9, 51], which substantially simplifies the learning problem. In contrast, our setting involves dynamic motion from both the camera and the target object, with no assumptions made about the nature of the interacting agent (e.g., human hand or robotic manipulator). As a result, the relative object-camera pose, especially the angular velocity, may vary significantly between adjacent frames. To enable the model to learn across a wide range of motion dynamics, we leverage a **curriculum-based progressive frame sampling strategy**. Training begins with temporally-close frames, gradually increasing the frame interval (thereby reducing co-visibility), and eventually transitions to fully random frame sampling. This staged approach ensures exposure to diverse object motions throughout training. As a result, the trained model generalizes well to both sequential video data and unordered image sets.

D.2 Training Data Rendering Pipeline.

For each object sampled from the Objaverse dataset, we generate a unique fly-around sequence using a *custom Blender script* that introduces controlled randomization across camera motion, optical parameters, and scene illumination. Our pipeline operates as follows:

First, we construct a smooth camera trajectory by sampling $K_1=4$ elevation angles $\{\theta_k\}$ and $K_2=8$ radius values $\{r_k\}$ within a spherical shell $[d_{\min}, d_{\max}]$. This sampling strategy ensures comprehensive coverage of both polar and azimuthal viewing angles. Through linear interpolation of these key points combined with uniformly distributed azimuth angles ϕ , we generate $N=100$ waypoints that exhibit smooth transitions in both vertical and radial dimensions. We then construct a periodic cubic spline through these waypoints and uniformly sample it to obtain the final camera positions. By re-seeding the random number generator for each object, we ensure that every sequence features a unique trajectory with object-specific zoom patterns.

During rendering, we employ a Track-To constraint to maintain camera focus on the object while introducing a small per-frame *look-at jitter* (± 5 cm on each axis) to prevent the object from remaining perfectly centered. To further enhance diversity, we randomly sample focal lengths from $\{30, 35, 40, 45, 50\}$ mm and configure a combination of area and point lights with randomized positions and intensities drawn from broad uniform distributions. This comprehensive approach yields training sequences characterized by diverse motion patterns, controlled scale variations, and rich illumination conditions.

D.3 Implementation Details.

We provide comprehensive implementation details of our OnlineSplatter framework below.

Model Architecture. Our model consists of several key components:

- **Image Encoders:** We use a frozen DINO backbone [1] as Encoder_1^I with patch size 8 and output dimension 768. The learnable Encoder_2^I follows the same architecture but is output dimension is 256. The concatenation of Encoder_1^I and Encoder_2^I provides a 1024 dimensional feature vector for each patch token.
- **OnlineSplatter Transformer:** The transformer processes tokenized inputs through 24 layers, each with 16 attention heads and hidden dimension 1024. We use layer normalization and a dropout rate of 0.05.
- **Memory Module:** The object memory maintains a maximum of $S = 1024 \times 20$ entries, with each entry containing token-level features of dimension $C = 1024$. The number of patches per view $P = 1024$ is determined by the input resolution 256×256 and patch size 8. The key and value encoders (Encoder^K and Encoder^V) are implemented as 3-layer MLPs with 1024 hidden units. We use OrientAnything [43] as the pre-trained direction key encoder (Encoder^D) and keep it frozen.
- **Model Size:** The total number of parameters in the model is around $488M$, of which $402M$ is trainable.
- **Rasterization:** We use the CUDA differentiable rasterizer implemented in the original 3DGS [16] to render the predicted 3D Gaussians. The rendering resolution matches the input resolution. We use near plane 0.1 and far plane 100.0 for rasterization.

Training Configuration. We employ a two-stage training strategy:

- **Optimizer:** We use AdamW [27] optimizer with learning rate $1e-4$, weight decay 0.05, and Cosine Annealing [26] learning rate schedule. The learning rate is warmed up for 2000 steps.
- **Training Losses:** The overall training objective (detailed in Section B) combines photometric and geometrical losses: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{photo}} + \lambda_g \mathcal{L}_{\text{geo}} = \mathcal{L}_{\text{masked}} + \lambda_{\text{bg}} \mathcal{L}_{\text{bg}} + \lambda_g (\mathcal{L}_{\text{ray}} + \lambda_d \mathcal{L}_{\text{depth}})$.
- **Warm-up Training Stage:**
 - **Steps:** Trained for $250K$ steps with effective batch size 64. We train the model without the memory module during this stage.
 - **Loss Weights:** $\lambda_g = 0.3$, $\lambda_{\text{bg}} = 0.3$, $\lambda_d = 0.5$
 - **View-specific Embedding:** At this stage, we train the reference-view embeddings ($f_{\text{ref}}^{\text{emb}}$) and source-view embeddings ($f_{\text{src}}^{\text{emb}}$) while keeping the memory-view embeddings not involved.
 - **Initialization:** All weights are initialized randomly using the truncated normal distribution with mean 0 and standard deviation 0.02. Note that the pre-trained DINO encoder weights (i.e., Encoder_1^f) are frozen and not updated, while the learnable encoder (Encoder_2^f) is being updated.
 - **Input Sampling:** We sample 3 – 5 views per sequence sample, with a sampling schedule as described in Sec. D.1.
- **Main Training Stage:**
 - **Steps:** Trained for $500K$ steps with effective batch size 16, incorporating the memory module.
 - **Loss Weights:** $\lambda_g = 0.3$, $\lambda_{\text{bg}} = 0.3$, $\lambda_d = 0.0$ ($\mathcal{L}_{\text{depth}}$ removed)
 - **View-specific Embedding:** To differentiate the two kinds of memory read-out tokens, we initialize two sets of memory-view embedding (i.e. $\{f_{\text{mem}1}^{\text{emb}}, f_{\text{mem}2}^{\text{emb}}\}$), one for orientation-aligned memory read-out and the other for orientation-complementary memory read-out.
 - **Initialization:** We initialize the transformer weights using the warm-up stage weights and we copy the source-view embeddings ($f_{\text{src}}^{\text{emb}}$) weights from the warm-up stage to initialize the memory-view embeddings ($\{f_{\text{mem}1}^{\text{emb}}, f_{\text{mem}2}^{\text{emb}}\}$). While the memory key encoder and value encoder weights are initialized randomly using the truncated normal distribution with mean 0 and standard deviation 0.02. Note that the pre-trained direction key encoder weights (i.e., Encoder^D) are frozen and not updated.
 - **Input Sampling:** We sample 6 – 12 views per sequence sample, with a sampling schedule as described in Sec. D.1.

Data Processing and Inference Details.

- **Image Processing:** Input images are resized to 256×256 and normalized using ImageNet statistics [6]. We apply random augmentations including mirroring with a probability of 0.5.
- **Camera Normalization:** We preprocess all the ground truth poses in a sequence to be relative to the reference view, such that the reference view pose is the identity matrix.
- **Memory Sparsification:** The memory is pruned when reaching the maximum memory size, removing 20% of entries based on usage and coverage metrics. The temperature coefficient τ_t is dynamically adjusted based on orientation confidence: $\tau_t = 2.5 - \sigma_t$, where $\sigma_t \in [0, 1]$ is the inferred confidence from the orientation estimator [43].
- **Object Masking:** At inference time, we use the off-the-shelf video online segmentation model from Xmem [3] to estimate the object mask based on the initial object mask from the reference view.
- **Rendering:** At inference time, we filter out the predicted Gaussians that are either low in opacity (i.e., $o_k < 0.0001$) or the 0th-degree spherical harmonics coefficients are close to the background color (set to $[1, 1, 1]$ for rendering object against white background).
- **Training Environment:** We train our model on 8x NVIDIA A100 GPUs with 80GB memory. Our implementation uses Python 3.10, PyTorch 2.1.2, torchvision 0.16.2, and we leverage xFormers [19] 0.0.23 for efficient attention computation.
- **Evaluation Environment:** We run all inference on a single L40S GPU with 48GB memory.

E Additional Discussions

E.1 Additional Notes on Mesh Visual Comparison.

In Figure 4 of the main paper, we convert our final 3D Gaussian Splatting (3DGS) representation into meshes and conduct a comprehensive visual comparison with state-of-the-art methods across different paradigms. Methods

marked with an asterisk (*) indicate that additional pre- or post-processing steps were applied to generate the visual results. Below, we detail the mesh generation process for each method and provide additional analysis.

- For **Fmov** [31], **BundleSDF** [44], **Fast3R** [49], and **InstantMesh** [47], we either utilized their official implementations to generate meshes or obtained results directly from their published papers or official websites.
- **3DGS***: Since the original 3DGS [16] implementation lacks support for pose-free object reconstruction and joint camera pose optimization, we employed the open-source gsplat [52] library with MCMC [17] random initialization for both 3DGS parameters and camera poses. We incorporated ground-truth object masks to optimize the 3DGS parameters.
- **COLMAP***: We attempted reconstruction using the official COLMAP [30] implementation both with and without ground-truth object masks. However, neither approach converged successfully, resulting in failed reconstructions.
- **VGGT***: Using the official VGGT [40] implementation, we preprocessed the input images by applying ground-truth object masks to remove background content. We utilized the Depthmap and Camera Branch outputs, which their paper indicates provide superior performance compared to the Pointmap branch. We additionally filtered out redundant background predictions to generate the final results.
- **Ours***: To convert our OnlineSplatter framework’s output into mesh format, we rendered 30 uniformly distributed views of the predicted 3D Gaussians, generating corresponding RGB-D images and masks. These were fused with their camera poses in a TSDF volume, and we extracted the mesh using Open3D [55]. The resulting mesh was further simplified using the open-source mesh simplification library [15]. Note that we only generate the mesh for a rough visual comparison, our method does not focus on mesh generation, we leave it as a future work.

From the visual comparison, our method demonstrates reconstruction quality comparable to approaches that require extensive optimization or additional depth supervision, while maintaining the advantages of a feed-forward, online framework. While optimization-based methods such as COLMAP* and 3DGS* excel at reconstructing static scenes, they face significant challenges when applied to freely moving objects, even with access to ground-truth object masks and the ability to perform global optimization across all frames. This observation highlights both the inherent difficulty of reconstructing freely moving objects from monocular videos and the promising capabilities of our approach. A promising future direction could be combining our online feed-forward framework with optimization-based refinement to achieve higher-quality mesh reconstructions.

E.2 Broader Impacts

Our work on online 3D reconstruction has several societal implications. On the positive side, the technology could democratize 3D content creation by making real-time 3D scanning more accessible to the general public, while also improving efficiency in manufacturing through real-time quality control and inspection. The real-time nature of our system could benefit assistive technologies and education by enabling interactive 3D visualization and understanding of objects. However, there are potential concerns regarding privacy and intellectual property, as the ability to quickly reconstruct 3D models could be misused for unauthorized scanning or copying of physical objects. We recommend implementing appropriate usage guidelines and access controls when deploying the technology in sensitive contexts, while encouraging responsible development that considers both privacy and intellectual property rights.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Our introduction clearly states the claims made, including the contributions made in the paper and important assumptions.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss the limitations of the work in the paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We disclose comprehensive training and implementation details in the main paper and appendix to facilitate reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We provide reference implementation where possible.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We provide comprehensive training and implementation details in the main paper and appendix to facilitate reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: We report average performance over multiple runs with different random seeds.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We fully disclose all compute resources used in the experiments in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification: We have checked the Code of Ethics, and our research fully conforms to it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts of our work in the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All papers and code used by us are properly cited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.