FlashSSM-3D: A Distilled State Space Model for Lightning-Fast Dense 3D Understanding

Nyle Siddiqui*
Institute of Artificial Intelligence
University of Central Florida

Mubarak Shah
Institute of Artificial Intelligence
University of Central Florida

Abstract

3D geometric understanding is undergoing a major transition where rigid, deterministic, optimization-based pipelines are being replaced by large-scale, deep learning models. While recent methods achieve impressive performance on tasks such as multi-view 3D reconstruction and camera pose estimation, they suffer from two critical limitations: **(i):** current state-of-the-art (SOTA) models require industry-level compute and training corpora, severely limiting accessibility, and **(ii):** recent approaches remain constrained to modifying existing pretrained transformer models, since substantially changing their architecture would require retraining from scratch. We instead study a complementary direction: training an attention-free state space model for dense 3D understanding. We introduce **FlashSSM-3D**, the first large-scale, multi-view, 3D reconstruction state space model capable of performing accurate geometric reasoning over hundreds of images with linear computational complexity. However, training such a model from scratch with competitive performance is impractical under typical academic compute constraints. To address this, we train FlashSSM-3D by distilling knowledge from VGGT, avoiding the need to learn 3D geometric priors entirely from scratch. However, standard distillation losses are poorly suited for dense 3D reconstruction, where preserving real-world spatial structure is essential. We therefore propose a **geometry-aware knowledge distillation** method specifically tailored for large-scale 3D reconstruction models. Our method introduces two novel losses: a Procrustes-based alignment loss that enforces global structural consistency between student and teacher 3D point predictions, and a camera reprojection loss that directly supervises view-consistent camera geometry between teacher and student. Extensive experiments across multi-view 3D reconstruction and camera pose estimation benchmarks demonstrate that FlashSSM-3D achieves competitive performance while requiring only $\sim 25\%$ of the training data and less than 10% of the training compute used by VGGT. At inference time, FlashSSM-3D benefits from the practical efficiency of state space models, reconstructing scenes from 500+ views in under 10 seconds while maintaining competitive accuracy.

1 Introduction

3D geometric understanding lies at the core of many fundamental computer vision tasks, including autonomous navigation [1, 2], robotics [3, 4], and large-scale scene reconstruction [5]. Given one or more images (also referred to as views) as input, the goal of 3D geometric understanding is to produce structured geometric annotations of the scene - such as depth maps, 3D point clouds, and camera pose estimations - which jointly describe both scene structure and the spatial relationship between the camera and the environment. Classical 3D reconstruction and camera estimation methods were dominated by deterministic pipelines such as Structure-from-Motion (SfM) [5, 6] and Multi-View Stereo (MVS) [7, 8, 9]. However, they relied on handcrafted feature extraction, correspondence

*Corresponding Author: nyle.siddiqui@ucf.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

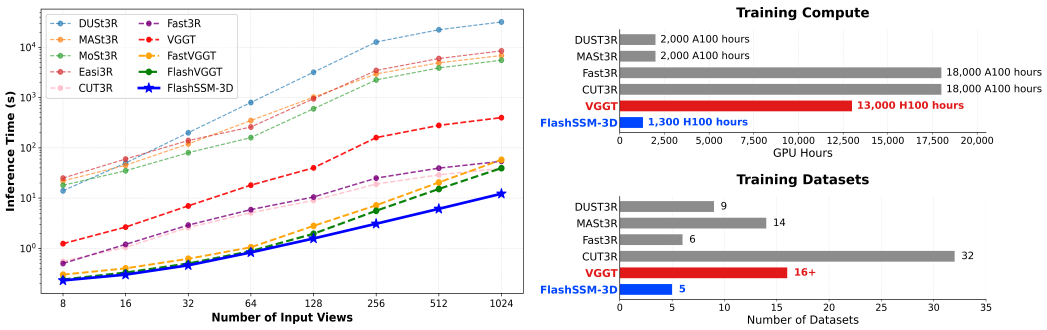


Figure 1: **FlashSSM-3D: a fast and training-efficient 3D reconstruction SSM.** FlashSSM-3D scales efficiently to hundreds of input views and requires *significantly* less compute and data to train than previous models.

matching between paired images, and global alignment, limiting their flexibility and scalability [10, 11]. The rise of deep learning fundamentally reshaped this landscape, with early learning-based approaches introducing feed-forward networks trained on large-scale, image-paired datasets with 3D supervision. These models [12, 13, 14, 15] leverage dense matching and learned geometric priors to recover 3D structure from image pairs and were the first learnable, end-to-end methods to generally frame 3D reconstruction as a pointmap regression problem. While effective, these methods are typically limited to a fixed and small number of views, constraining their applicability in multi-view settings and often requiring expensive post-processing steps like bundle adjustment. More recently, a new class of models has emerged that can process an arbitrary number of input images and perform global 3D reasoning across views [16, 17, 18, 19, 20]. These newer models demonstrate impressive reconstruction quality by jointly reasoning over multiple frames using large transformer-based architectures, enabling scalable multi-view reconstruction and significantly improving robustness to viewpoint diversity and scene complexity.

Despite these advances, current feed-forward 3D models remain difficult to scale beyond the transformer architectures on which they are built. For example, VGGT [16] achieves strong performance using a 1B+ parameter transformer trained for over $13,000$ GPU hours, making substantial architectural changes infeasible without retraining at industry scale. Moreover, the computational cost of transformer-based 3D reconstruction models scales poorly with the number of input images at inference time, due to their reliance on quadratically-scaling attention. In practice, inference on hundreds of images can take on the order of minutes, rendering large-scale, real-time reconstruction impractical. As a result, recent acceleration methods [21, 22] alleviate inference cost by modifying existing transformer models, but they do not answer a key question: **can we move beyond attention-based 3D reconstruction models entirely, while maintaining competitive performance?**

To address these challenges, we propose **FlashSSM-3D**: a lightning-fast reconstruction model that demonstrates the efficacy of SSMs for feed-forward 3D reconstruction. FlashSSM-3D is a fully Mamba-based 3D reconstruction model that uses linear-time state-space layers for sequence mixing, enabling efficient inference on large-scale, multi-view scenes. However, moving from the attention mechanism to selective scanning for sequence modeling introduces a substantial representational gap, making it difficult to recover the global geometric reasoning of large transformer models through standard training alone. To bridge this gap, we train FlashSSM-3D by distilling knowledge from VGGT, avoiding the need to learn 3D geometric priors entirely from scratch. Yet, we find that the naive application of knowledge distillation is insufficient for fully transferring strong geometric understanding into an attention-free architecture, presumably due to the complexity of 3D understanding. Therefore, we introduce a novel **geometry-aware** knowledge distillation framework for 3D reconstruction. To the best of our knowledge, this is the first cross-architecture, knowledge distillation approach explicitly designed for large-scale 3D reconstruction models. We propose two novel geometry-aware distillation losses: a Procrustes-based alignment loss that enforces global structural consistency between student and teacher 3D point predictions, and a camera reprojection loss that directly supervises view-consistent camera geometry between teacher and student. Together, these losses enable effective transfer of geometric reasoning into FlashSSM-3D, resulting in a fully Mamba-based 3D reconstruction model that benefits from the efficient long-sequence scaling of

SSMs while maintaining competitive reconstruction and camera pose accuracy (Fig. 1). Our main contributions are as follows:

1. We introduce FlashSSM-3D, a Mamba-based 3D reconstruction model that validates the applicability of SSMs for multi-view 3D understanding.
2. We demonstrate that traditional knowledge distillation is insufficient for transferring strong 3D geometric priors across architectures, and propose a geometry-aware distillation framework tailored specifically for cross-architecture 3D understanding distillation.
3. We demonstrate that FlashSSM-3D achieves competitive multi-view reconstruction and camera pose estimation performance while requiring $\sim 25\%$ of the training data and $< 10\%$ training compute than VGGT.

2 Related Works

2.1 State Space Models

Structured state space models (SSMs) [23, 24, 25] have recently gained significant attention as efficient sequence modeling architectures for vision applications, ranging from image classification [26] to video understanding [27]. Their appeal comes in part from their dual parameterization form: SSMs can be implemented as a recurrence for near-linear scaling with sequence length, or equivalently as a convolution for parallel GPU execution. Conceptually, SSMs map an input sequence to an output sequence by maintaining a learned latent state that evolves over time. Given an input signal $x(t) \in \mathbb{R}^L$, the hidden state $h(t) \in \mathbb{R}^N$ is updated according to a latent dynamical process and used to generate an output $y(t) \in \mathbb{R}^L$. This process is commonly formulated as a continuous-time linear system defined by parameters $(\mathbf{A}, \mathbf{B}, \mathbf{C})$: $h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$, and $y(t) = \mathbf{C}h(t)$, where $\mathbf{A} \in \mathbb{R}^{N\times N}$ controls the evolution of the latent state, while $\mathbf{B} \in \mathbb{R}^{1\times N}$ and $\mathbf{C} \in \mathbb{R}^{N\times 1}$ control input and output projections, respectively, of the hidden state. For discrete input sequences, these continuous dynamics are converted into a discrete system using a learnable step size Δ. Under a zero-order hold discretization, the parameters become:

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}), \qquad \bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}\left(\exp(\Delta\mathbf{A}) - I\right)\Delta\mathbf{B}, \qquad \bar{\mathbf{C}} = \mathbf{C}. \tag{1}$$

Thus, an SSM can be evaluated through two mathematically equivalent views. The recurrent form updates the hidden state sequentially (left), while the convolutional form applies a structured kernel constructed from repeated applications of the state transition matrix (right):

$$\begin{aligned} h_t &= \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t, & \bar{\mathbf{K}} &= (\bar{\mathbf{C}}\bar{\mathbf{B}}, \bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{B}}, \cdots, \bar{\mathbf{C}}\bar{\mathbf{A}}^t\bar{\mathbf{B}}), \\ y_t &= \bar{\mathbf{C}}h_t, & y &= x * \bar{\mathbf{K}}. \end{aligned}$$

In practice, the convolutional formulation is often preferred during training due to its parallelizability, while the recurrent form is advantageous at inference time for efficient sequential evaluation with constant memory cost.

2.2 3D Reconstruction and Multi-View Geometry

A long-standing goal in 3D vision is to recover scene structure (e.g., depth, dense correspondences, and 3D point maps) and camera motion from one or more images. Classical pipelines rely on geometric optimization [5, 28, 29, 30, 31] but are often brittle under texture-less regions, dynamic content, or extreme viewpoint changes [32, 33]. Furthermore, they often suffer heavily from their reliance on iterative, post-processing techniques (e.g. bundle adjustment) to align their predictions. Recent feed-forward approaches instead learn to directly predict dense geometry from images, substantially improving robustness and enabling faster inference at scale.

Early progress in learned dense reconstruction, such as DUSt3R [12] and MASt3R [13], matched predicted dense 3D point maps between two views, enabling downstream alignment and reconstruction without classical feature matching. While pairwise methods are effective and simple, extending them to longer sequences typically requires additional fusion or optimization to ensure global consistency across many views. To address scalability beyond two views, recent methods

predict globally consistent geometry from an arbitrary number of input images. VGGT [16] leverages larger amounts of data paired with dense prediction heads to learn multi-view aggregation for dense 3D reconstruction and camera estimation. Fast3R [18] similarly targets efficient multi-view reconstruction, emphasizing practical speed/accuracy trade-offs for dense 3D prediction over multiple images. Recent works have also explored [21, 22] improving inference efficiency through attention and token optimizations. Tangentially, some works focus on producing reconstructions from image streams, where images arrive sequentially and memory/compute must remain bounded [20, 19, 34]. While effective, these methods exclusively build upon existing pretrained transformer backbones, leaving unresolved whether fundamentally different sequence models can support feed-forward 3D reconstruction. This motivates our exploration of state space models; however, moving from transformers to SSMs introduces a challenging cross-architecture distillation problem, where the student must inherit geometric reasoning from an architecturally distinct teacher. To address this, we propose FlashSSM-3D: a large 3D reconstruction SSM trained with a geometry-aware distillation framework designed specifically for transferring 3D geometric priors across architectures.

3 Methodology

FlashSSM-3D is an attention-less feed forward network capable of taking a set of images as input and producing camera pose and point map predictions. We start by providing some brief background and problem setup in Sec. 3.1, followed by architecture and training details in Sections 3.2 and 3.3, and discussion of our geometry-aware knowledge distillation framework in Sec. 3.4.

3.1 Background and notation

The problem of geometric understanding revolves around taking a set of unordered and unposed images $\mathbf{X} = \{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^{H \times W \times 3}$ with height H and width W, and producing various 3D annotations as output. The objective of FlashSSM-3D is to take $\mathbf{X}$ as input and accurately produce the corresponding 3D annotations for a given scene: FlashSSM-3D: $\mathbf{X} \rightarrow (\mathbf{C}, \mathbf{P})$, where $\mathbf{C} = \{c_i\}_{i=1}^N$ is the set of all camera pose predictions for each image, and $\mathbf{P} = \{p_i\}_{i=1}^N$ is the set of all 3D point predictions for each pixel in each image. Specifically, we parameterize the camera predictions for a given image $c_i = [r_i, t_i, f_i,]$ following [16, 35], where the rotation quaternion $r_i \in \mathbb{R}^4$, translation vector $t_i \in \mathbb{R}^3$, and field of view $f_i \in \mathbb{R}^2$, respectively, are concatenated together. The predicted set of 3D points of an image $p_i \in \mathbb{R}^{H \times W \times 3}$ simply predicts the location of each pixel in the image in a 3D Euclidean space (x, y, z).

3.2 Model Architecture & Training

FlashSSM-3D is constructed with block-level correspondence to VGGT, allowing knowledge to be transferred from the transformer teacher into our attention-free student by replacing frame-wise and global self-attention with selective scan state-space layers. This substitution transforms the model from a quadratic-complexity transformer into a subquadratic state-space architecture, enabling scalable inference over hundreds of views. We use the same image encoder as VGGT (DINO [36]) to extract visual features for each image; yet, this has a negligible impact on runtime, which is dominated by the actual reconstruction section of the model (Fig. 2).

However, directly replacing attention with state-space layers introduces a substantial representational gap. While both architectures perform sequence mixing, attention relies on explicit pairwise token interactions, whereas selective scan performs recurrent updates in a learned latent space. As a result, training the student model from scratch would require prohibitively large-scale data and compute to recover VGGT's geometric reasoning capabilities, especially considering the resources used to train VGGT (16+ datasets, $13,000+$ GPU hours). To solve this, we leverage knowledge distillation to instill FlashSSM-3D with knowledge from VGGT. Crucially, because of the architectural mismatch between teacher and student and the complex nature of 3D geometric understanding, standard knowledge distillation does not fully suffice (see Sec. 4.5). Therefore, we adopt a multi-stage cross-architecture distillation strategy that progressively transfers geometric reasoning from the attention-based teacher to our SSM-based student.

3.3 Subquadratic Distillation

Our distillation process begins by aligning internal representations between teacher and student before imposing specific geometric supervision. We adopt the three-stage subquadratic distillation framework of [37], where the first two stages focus on representation-level alignment. We make significant changes to the vital third stage of distillation as discussed in Sec. 3.4.

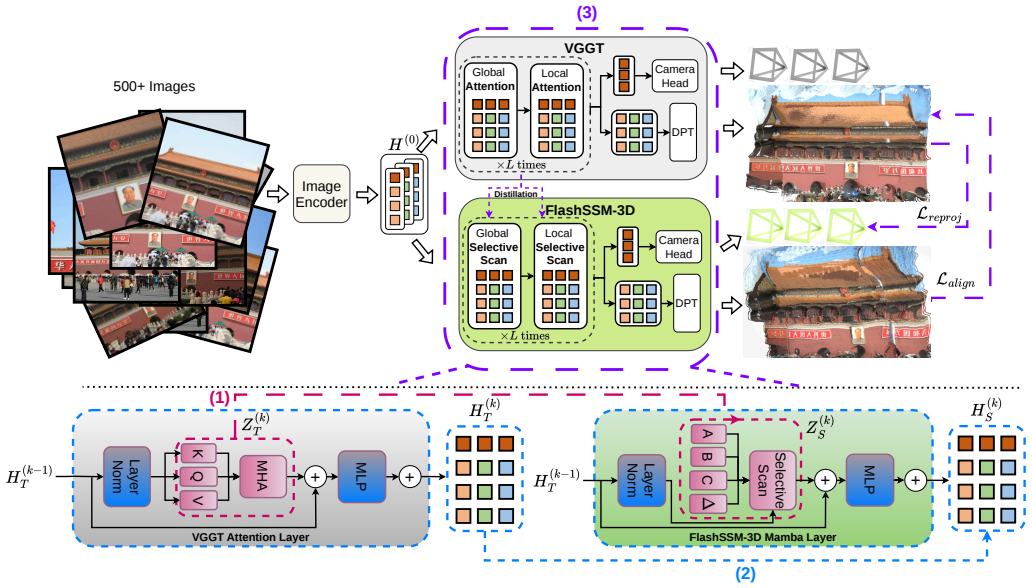


Figure 2: Distilling VGGT into FlashSSM-3D occurs over 3 stages: **(1): Matrix Mixer Orientation -** Aligns the student's internal selective scan output with the teacher's self-attention KQV output. **(2): Hidden-State Alignment -** Matches block-level representations between teacher and student by training all block components (e.g. linear layers) simultaneously. **(3): Geometry-Aware Knowledge Distillation -** Directly supervises 3D points and camera predictions using Procrustes alignment and reprojection losses. Collectively, this transfers VGGT's geometric reasoning into our student model.

3.3.1 Matrix Mixer Orientation.

The first stage of subquadratic distillation aims to align the student's *matrix mixer* mechanism with the teacher's *matrix mixer*. [37] defines a matrix mixer as the core sequence-mixing component of a block, such as the KQV matrices in attention or the state-evolution matrix $\mathbf{A}$ in SSMs. This stage aims to ensure that both student and teacher perform comparable token mixing within each block prior to downstream geometric supervision. Specifically, given $\mathbf{X}$ as input, both the teacher and student first map the images to hidden token embeddings through a shared image encoder: $\mathbf{H}^{(0)} = \mathrm{Enc}(\mathbf{X})$, where $\mathbf{H}^{(0)} \in \mathbb{R}^{B\times L\times d}$ denotes the initial token representations, B is the batch size, L is the sequence length, and d is the embedding dimension. The teacher consists of K stacked attention blocks $\mathcal{B}_T^{(k)}$ that produce corresponding hidden token representations $\mathbf{H}_T^{(k)}$ at each block:

$$\mathbf{H}_T^{(k)} = \mathcal{B}_T^{(k)}(\mathbf{H}_T^{(k-1)}), \quad k = 1, \ldots, K.$$

Let $\mathrm{TeacherMixer}(\cdot)$ denote the teacher's self-attention mixer and $\mathrm{StudentMixer}_{\boldsymbol{\phi}}(\cdot)$ denote the student's selective scan matrix mixer parameterized by $\boldsymbol{\phi} = \{\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}, \Delta\}$ (see Eq. 1). Within each block, the mixer operates on the incoming hidden tokens to produce intermediate mixed token representations before the remaining block components (e.g., projections and residual connections) are applied. We denote the teacher's mixer output at the k-th block (left) and the student's mixer output (right) as:

$$\mathbf{Z}_T^{(k)} = \mathrm{TeacherMixer}(\mathbf{H}_T^{(k-1)}), \qquad \mathbf{Z}_S^{(k)} = \mathrm{StudentMixer}_{\boldsymbol{\phi}}(\mathbf{H}_T^{(k-1)}),$$

where $\mathbf{Z}_T^{(k)}, \mathbf{Z}_S^{(k)} \in \mathbb{R}^{B\times L\times d}$. To isolate the effect of only the student mixer during this stage (shown in pink in Fig. 2), the student mixer receives the same teacher hidden tokens as input. We freeze the teacher and optimize only the student mixer parameters $\boldsymbol{\phi} = \{\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}, \Delta\}$ by minimizing the Frobenius distance between corresponding matrix mixers across all blocks.

$$\min_{\boldsymbol{\phi}} \sum_{k=1}^{K} \left\| \mathbf{Z}_S^{(k)} - \mathbf{Z}_T^{(k)} \right\|_F^2. \tag{2}$$

This stage essentially ensures that the token representations produced by the student's matrix mixer at each block emulate those produced by the teacher, i.e., $\mathbf{Z}_S^{(k)} \approx \mathbf{Z}_T^{(k)}$. While this establishes compatibility at the core token interaction level despite the shift from attention to state-space recurrence, it does not guarantee equivalence at the block output level.

3.3.2 Hidden-State Alignment

Following matrix mixer orientation, the second stage of subquadratic distillation aligns the block-level outputs of the student and teacher models. While stage 1 ensures comparable intermediate block representations, differences may still arise in the remaining block components. This stage therefore aims to match the final hidden state from each student block to that of the corresponding teacher block (shown in blue in Fig. 2). Let $\boldsymbol{\theta}_S$ denote the full set of student block parameters, for example including linear projection and layer normalization weights. The student block parameters $\boldsymbol{\theta}_S$ are optimized by minimizing the ℓ_2 distance between corresponding block outputs across all layers:

$$\min_{\boldsymbol{\theta}_S} \sum_{k=1}^{K} \left\| \mathbf{H}_S^{(k)} - \mathbf{H}_T^{(k)} \right\|_2^2, \tag{3}$$

where $H_S^{(k)}$ denotes the student block outputs computed via its own forward pass. Together, these two stages align the student and teacher at the intermediate feature level and can be performed in parallel across layers. The resulting objective encourages the student's Mamba blocks to mirror the output from the corresponding attention blocks in the teacher, providing a semi-aligned representation space to perform geometry-aware knowledge distillation. However, simple prediction alignment in standard knowledge distillation alone does not ensure geometric consistency (see Tab. 3). In 3D reconstruction, small discrepancies in intermediate features can accumulate across layers and manifest as large errors in global structure or camera pose, hence the need for stricter and more direct geometric supervision during distillation.

3.4 Geometry-Aware Knowledge Distillation

Let f_T and f_S denote the teacher and student networks, respectively. Recall that given a set of N input views $\mathbf{X} = \{x_i\}_{i=1}^N$, both models produce per-view predictions: $f_T(\mathbf{X}) = (\mathbf{C}^T, \mathbf{P}^T)$ and $f_S(\mathbf{X}) = (\mathbf{C}^S, \mathbf{P}^S)$, where $\mathbf{C} = \{c_i\}_{i=1}^N$ are camera pose predictions, and $\mathbf{P} = \{p_i\}_{i=1}^N$ are dense 3D point maps. Since VGGT produces dense geometric outputs rather than class probabilities, we apply direct regression losses between student and teacher predictions:

$$\mathcal{L}_{\text{KD}} = \sum_{i=1}^{N} \left(\|p_i^S - p_i^T\|_1 + \|c_i^S - c_i^T\|_1 \right), \tag{4}$$

where superscripts S and T denote student and teacher outputs, respectively. The difficulty with applying *only* standard knowledge distillation is that 3D reconstruction ultimately requires consistency in a real geometric space, which may extend beyond simple point-wise regression between student and teacher. To ensure that geometric reasoning itself is transferred to the student, we introduce two novel geometry-aware auxiliary losses to supervise the student on a deeper structural level.

3.4.1 Procrustes Alignment Loss

To enforce global structural consistency between student and teacher 3D reconstructions, we introduce a Procrustes alignment loss on the 3D points. Prior work has explored monocular scale ambiguity by performing *scale-only* alignment between predicted and reference geometries. However, scale alignment alone does not ensure that the predicted 3D points from the student model are aligned with the same rotation and translation as the teacher (see Sec. 4.5). Therefore, we align student and teacher reconstructions using a *full similarity transform*. Let $p_{i,j}^S \in \mathbb{R}^3$ and $p_{i,j}^T \in \mathbb{R}^3$ denote the 3D point at pixel j of view i predicted by the student and teacher, respectively. We compute the optimal scale $s^* \in \mathbb{R}$, rotation $\mathbf{R}^* \in SO(3)$, and translation $\mathbf{t}^* \in \mathbb{R}^3$ via the closed-form, differentiable Umeyama algorithm [38]:

$$(s^*, \mathbf{R}^*, \mathbf{t}^*) = \arg\min_{s,R,t} \sum_{i=1}^{N} \sum_{j=1}^{HW} \left\| sRp_{i,j}^S + t - p_{i,j}^T \right\|_2^2. \tag{5}$$

The Procrustes alignment loss is then defined as

$$\mathcal{L}_{\text{align}} = \sum_{i=1}^{N} \sum_{j=1}^{HW} \left\| s^*\mathbf{R}^* p_{i,j}^S + \mathbf{t}^* - p_{i,j}^T \right\|_2^2. \tag{6}$$

This loss is invariant to global scale and coordinate frame differences alike, while directly supervising the global 3D points predicted by the student. This enforces that the student's predicted geometry is globally consistent with the teacher's up to a known similarity transform.

3.4.2 Camera Reprojection Loss

While the Procrustes alignment loss enforces global consistency between student and teacher 3D reconstructions, it operates purely in a relative 3D coordinate frame and remains agnostic to how these points are observed from individual camera views. Consequently, globally aligned point clouds do not necessarily imply accurate camera pose estimation. In particular, errors in predicted rotations or translations may persist even when the reconstructed structure appears well aligned. To explicitly supervise the student's camera predictions, we introduce a *camera reprojection loss* that enforces view-consistent projections of the teacher's 3D points.

Let $p_{i,j}^T \in \mathbb{R}^3$ denote the teacher-predicted 3D point corresponding to pixel j in view i, expressed in the teacher's world coordinate frame. Recall that camera predictions from both the student and teacher is parameterized as $\mathbf{c}_i = [r_i, t_i, f_i,] \in \mathrm{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^2$ and they share the same intrinsics matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$. Under a pinhole camera model Π, a single 3D point $p_{i,j}$ is projected to image coordinates $\tilde{u} = \mathbf{K}\left(r_i * p_{i,j} + \mathbf{t}_i\right)$ and homogeneously normalized: $\tilde{\mathbf{u}} = \mathbf{K}\left(r_i * p_{i,j} + \mathbf{t}_i\right), \quad \Pi(p_{i,j}; \mathbf{c}_i) = \left(\frac{\tilde{u}_x}{\tilde{u}_z}, \frac{\tilde{u}_y}{\tilde{u}_z}\right)$. This operation first transforms the 3D point into the camera coordinate frame and then applies perspective division to obtain pixel coordinates. The intuition behind this is to use the teacher's 3D points and camera predictions as a fixed geometric reference, thereby isolating the student's camera pose errors from its 3D reconstruction quality. To specifically better align the student's camera predictions with respect to the teacher, we project the same fixed teacher 3D point $p_{i,j}^T$ using both the student-predicted camera parameters $\mathbf{c}_i^S$ and teacher camera parameters $\mathbf{c}_i^T$, and penalize their discrepancy in the image plane:

$$\mathcal{L}_{\text{reproj}} = \sum_{i,j} \left\| \Pi(p_{i,j}^T; \mathbf{c}_i^S) - \Pi(p_{i,j}^T; \mathbf{c}_i^T) \right\|_1 .$$

Unlike Procrustes alignment, which enforces global structural consistency in 3D space, the reprojection loss enforces view-level consistency under perspective geometry in the student's camera predictions with respect to the teacher. Therefore, the two geometry-aware losses are complementary: the camera reprojection loss primarily improves camera pose accuracy, while the Procrustes loss improves global 3D structure. Together, they enable geometry-consistent knowledge transfer without additional annotations or optimizations. Thus, the final Stage-3 training objective is:

$$\mathcal{L} = \lambda_{\text{KD}}\mathcal{L}_{\text{KD}} + \lambda_{\text{reproj}}\mathcal{L}_{\text{reproj}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}.$$

4 Experiments

In this section, we evaluate the effectiveness of our proposed approach on both camera pose estimation and multi-view 3D reconstruction tasks. Importantly, we show **(1):** our geometry-aware knowledge distillation method effectively transfers VGGT's capabilities into FlashSSM-3D, **(2):** FlashSSM-3D achieves VGGT-level performance with less training data and compute than VGGT, and **(3):** FlashSSM-3D performs inference significantly faster than most models, highlighting the benefit of its linear complexity. We also provide training and testing details, as well as ablations on our knowledge distillation framework and number of inference views.

4.1 Training

FlashSSM-3D is trained on a much smaller subset of the datasets used to train VGGT. While VGGT was trained on $16+$ large-scale datasets spanning diverse indoor and outdoor environments, our model is trained using only five datasets: **MegaDepth** [39], **ScanNet** [40], **CO3Dv2** [41], **ARKitScenes** [42], and **BlendedMVS** [43]. These datasets collectively cover a wide range of scene types, camera motions, and geometric complexity. Furthermore, we only train FlashSSM-3D using 4 H100 GPUs for a total of $\sim 1,300$ GPU hours; a significant reduction in data and compute than the full VGGT training recipe (see Table 4). More implementation details can be found in the supplementary.

4.2 Evaluation Metrics

Following prior work [16, 18, 44, 45], we evaluate performance using standard metrics for both camera pose estimation and 3D reconstruction. For camera pose evaluation, we report *Relative Translation Error* (RTE) and *Relative Rotation Error* (RRE), which measure the angular error (in degrees) in translation and rotation, respectively, between the predicted and ground-truth camera. We

also report AUC@30, which is the area under the accuracy–threshold curve up to 30° computed across various thresholds on RTE and RRE. For 3D reconstruction quality, we evaluate predicted geometry using error and completion. Reconstruction error measures the average distance between predicted 3D points and the ground truth points, while reconstruction completion measures the average distance between the ground truth points and the predicted 3D points. Accuracy measures the percentage of ground truth surface points that are within a fixed distance threshold of the predicted points.

4.3 3D Reconstruction Results

We first evaluate FlashSSM-3D on large-scale multi-view 3D reconstruction benchmarks to assess its ability to recover accurate scene geometry compared to VGGT. Following [46], we evaluate FlashSSM-3D on 5 different datasets in Table 1 to evaluate its reconstructions across indoor, object-centric, and scene-level settings: DTU [47] for object-centric reconstructions, and 7-Scenes [48], Neural-RGBD [49], ScanNet [40], and TUM-RGBD [50] for indoor reconstructions at both the object and scene level. For each scene, we randomly sample 10 input views unless otherwise specified. As shown in Table 1, FlashSSM-3D achieves performance comparable to VGGT across all reconstruction metrics, despite being trained with roughly 25% of the data and 10% of the compute. Notably, these results demonstrate that geometry-aware distillation enables FlashSSM-3D to faithfully inherit VGGT's implicit geometric reasoning, even under a fundamentally different state-space architecture. For example, transformer-based models that are specifically tailored for speed such as CUT3R [19] and Fast3R [18] suffer from suboptimal performance compared to VGGT in Table 1 due to the trade-off between model speed and accuracy. However, FlashSSM-3D combines the efficiency of attention-free sequence modeling with the strong geometric capabilities inherited through distillation from VGGT. This allows us to increase the number of views during inference, slightly improving reconstruction quality with a negligible increase in inference speed compared to prior works. Since DTU only has 49 views per-scene, it is omitted for extremely dense reconstruction. Note that CUT3R is a streaming method and is evaluated here outside its native online setting.

Table 1: 3D reconstruction results across various indoor object-level, and scene-level datasets. Best results are in **bold**, with second best underlined. FlashSSM-3D outperforms all baselines in runtime due to its use of linear-time selective scans. While some models have competitive runtime, FlashSSM-3D significantly outperforms them in error due to VGGT knowledge distillation.

Method	TUM-RGBD		ScanNet		NRGBD		7-Scenes		DTU		Time
	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	
Dense Views (50)											
DUSt3R [12]	0.620	0.474	1.19	0.940	0.046	0.030	0.032	0.041	1.284	1.349	$\sim 500s$
MASt3R [13]	0.209	0.211	0.890	0.770	0.043	0.024	0.025	0.039	1.374	1.409	$\sim 450s$
CUT3R [19]	0.740	0.595	0.794	0.752	0.346	0.184	0.179	0.097	4.710	2.413	$\sim 4s$
Fast3R [18]	0.711	0.337	0.725	0.777	0.128	0.067	0.045	0.047	3.647	2.319	$\sim 6s$
VGGT [16]	**0.105**	**0.101**	**0.434**	**0.411**	**0.015**	**0.012**	**0.019**	**0.032**	**1.125**	**1.295**	$\sim 9s$
FlashSSM-3D	0.124	0.140	0.466	0.423	0.028	0.024	0.020	0.036	1.22	1.38	**0.6s**
Extremely Dense Views (300+)											
DUSt3R [12]	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
MASt3R [13]	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
CUT3R [19]	0.690	0.651	0.808	0.757	0.132	0.056	0.041	0.029	-	-	$\sim 22s$
Fast3R [18]	0.698	0.322	0.733	0.727	0.061	0.031	0.040	0.056	-	-	$\sim 35s$
VGGT [16]	**0.074**	**0.083**	**0.433**	0.406	**0.016**	**0.017**	**0.020**	**0.027**	-	-	$\sim 130s$
FlashSSM-3D	0.100	0.092	0.458	**0.399**	0.043	0.026	0.028	0.034	-	-	**3.5s**

4.4 Camera Pose Estimation Results

Following [16], we further evaluate camera pose estimation accuracy on the RealEstate10K [51] and CO3Dv2 [41] datasets. As shown in Table 2, FlashSSM-3D matches or closely approaches VGGT's camera pose accuracy while operating in a fully attention-less manner. Importantly, both VGGT and FlashSSM-3D perform better when the number of inference views increases, however VGGT suffers from inefficient quadratic scaling, impacting inference speed. FlashSSM-3D is effectively able to ingest 500 frames of size 518×336 and still perform camera pose estimation either faster or quite comparably against previous works, highlighting the importance of an accurate yet scalable 3D geometric understanding model (CO3Dv2 does not have such dense views per scene).

4.5 Training Ablation

To better evaluate the impact of our novel losses on our knowledge distillation framework, we ablate both our Procrustes alignment and camera reprojection losses on the ScanNet and RealEstate10K

Table 2: Comparison on Rel10K (unseen) and CO3Dv2. Parentheses indicate how many frames are used during inference.

Methods	Rel10K	CO3Dv2	Time
	AUC@30↑	AUC@30↑	
Colmap+SPSG [11]	45.2	25.3	∼15s
PixSfM [52]	49.4	30.1	>20s
PoseDiff [45]	48.0	66.5	∼7s
DUSt3R [12]	67.7	76.7	∼7s
MASt3R [13]	81.8	76.4	∼9s
VGGSfM v2 [35]	78.9	83.4	∼10s
MV-DUSt3R [53]	71.3	69.5	∼0.6s
CUT3R [19]	75.3	82.8	∼0.6s
FLARE [14]	78.8	83.3	∼0.5s
Fast3R [18]	72.7	82.5	∼0.2s
FastVGGT [21]	84.3	86.5	<0.5s
FlashVGGT [22]	85.3	86.8	<0.5s
VGGT [16]	85.3	88.2	0.7s
FlashSSM-3D	83.3	86.8	**0.2s**
VGGT (300)	95.1	**88.5**	131.4s
FlashSSM-3D (300)	93.4	87.6	**3.5s**
VGGT (500)	OOM	OOM	OOM
FlashSSM-3D (500)	**95.2**	-	**5.9s**

Table 3: Ablation study showing the contribution of each loss component across camera pose estimation and 3D reconstruction on Rel10K and ScanNet, respectively.

Losses			Pose	3D Recon	
$\mathcal{L}_{KD}$	$\mathcal{L}_{reproj}$	$\mathcal{L}_{align}$	AUC↑	Err.↓	Comp.↓
✓	✗	✗	80.2	0.551	0.456
✓	✓	✗	82.6	0.545	0.428
✓	✗	✓	80.8	0.507	0.418
✓	✓	✓	83.3	**0.458**	**0.399**
Baseline VGGT			85.3	0.433	0.406

Table 4: Training scale and time for recent popular models. Our geometry-aware knowledge distillation enables FlashSSM-3D to emulate VGGT performance on TUM-RGBD with significantly less data and compute.

Model	Params	Training Time	Datasets	Error (↓)
DUSt3R [12]	∼ 0.6B	∼ 2,000 A100 hours	9	0.620
MASt3R [13]	∼ 0.7B	∼ 2,000 A100 hours	14	0.209
Fast3R [18]	∼ 0.6B	∼ 18,000 A100 hours	6	0.711
CUT3R [19]	∼ 0.8B	∼ 18,000† A100 hours	32	0.740
VGGT [16]	∼ 1B	∼ 13,000 H100 hours	16+	**0.105**
FlashSSM-3D	∼ 1B	∼ **1,300** H100 hours	**5**	0.124

† Hours not directly reported, estimated given open-source codebase

datasets, respectively, in Table 3. Starting from only standard knowledge distillation, we observe moderate performance across both camera pose estimation and 3D reconstruction. Incorporating the camera reprojection loss yields a notable improvement in pose accuracy and slightly improves reconstruction quality, indicating that enforcing geometric consistency at the image formation level directly benefits camera estimation. In contrast, adding the Procrustes alignment loss primarily improves 3D reconstruction metrics, reducing reconstruction error and improving completeness. When both losses are combined, the model achieves the best overall performance, significantly reducing reconstruction error (0.458) and improving completeness (0.399) while also achieving the highest AUC@30 among the student variants. These results demonstrate that our two losses are complementary: Procrustes alignment strengthens global geometric consistency in 3D space, while camera reprojection refines pose accuracy through image-level supervision, and their joint optimization is critical for effective geometry-aware distillation.

4.6 Ablating the Number of Inference Views

Table 5 ablates the number of frames used at inference time and runtime for camera pose estimation. While both VGGT and FlashSSM-3D benefit from more views during inference, VGGT severely suffers from a quadratically growing runtime, even with certain GPU-level attention optimizations in PyTorch. However, FlashSSM-3D benefits both from increasing performance with more views as well as small increases in runtime. This especially highlights the benefit of our geometry-aware knowledge distillation, as we are able to both leverage the capabilities of VGGT while simultaneously harnessing the efficiency of SSMs; an attribute not easily attained with regular distillation as we showed in the previous section.

4.7 Qualitative Results

We provide qualitative comparisons on unseen scenes in Figure 3 (with additional visualizations in Sec. A.4). FlashSSM-3D produces dense 3D geometry that is visually comparable to VGGT, at a significantly lower runtime. For 50+ input views, FlashSSM-3D reconstructs the scene nearly 16× faster than VGGT, highlighting the practical benefit of replacing quadratic attention with linear-time selective scan layers. Visually, FlashSSM-3D preserves fine geometric structure and completeness across complex architectural regions, which are common failure cases for fast reconstruction models. These results further demonstrate that, despite the architectural shift from attention to selective scanning with state space modeling, our geometry-aware distillation framework effectively transfers the teacher's geometric understanding while enabling substantially faster inference.

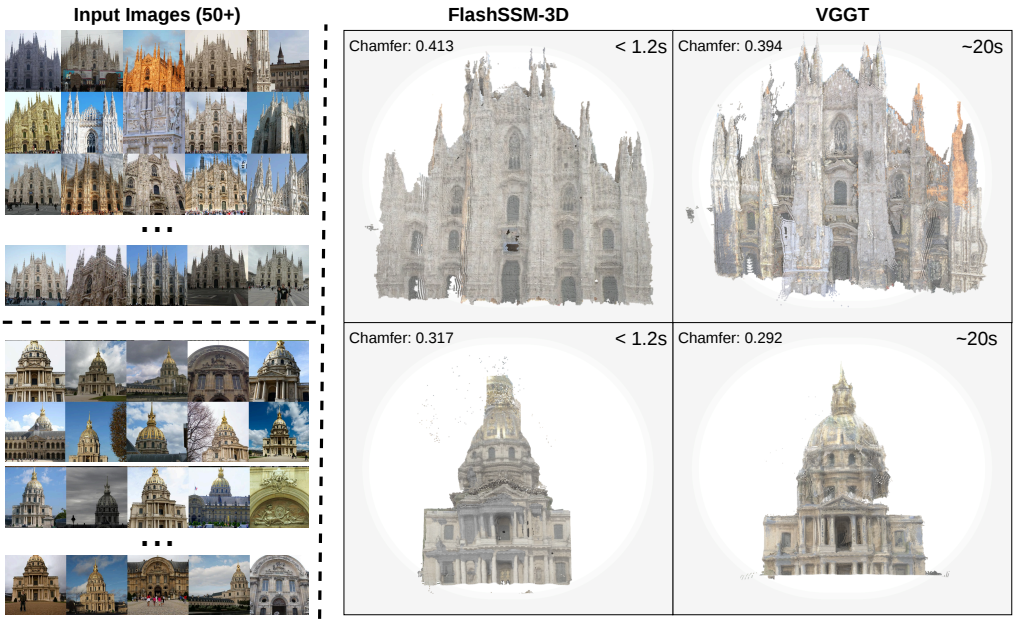


Figure 3: Qualitative results on **unseen** scenes. FlashSSM-3D retains VGGT-level accuracy (shown using Chamfer distance) while being 16x faster when reconstructing a dense number of views.

Table 5: Camera pose estimation results with various number of views at inference. On Rel10K, using more frames significantly improves performance at the cost of runtime. FlashSSM-3D is capable of leveraging more frames while also keeping runtime low, while VGGT's runtime grows quadratically.

Model	# Images	Rel10K					CO3Dv2					Time
		AUC@30 ↑	RTE ↓	RRE ↓	acc@10 ↑	acc@20 ↑	AUC@30 ↑	RTE ↓	RRE ↓	acc@5 ↑	acc@10 ↑	
VGGT	10	85.3	1.08	3.99	81.4	91.1	88.2	1.06	3.9	63.7	**99.8**	0.7s
	20	89.4	**1.03**	2.64	89.4	97.1	**88.3**	1.01	**3.6**	**72.5**	**100**	1.9s
	50	93.8	1.07	1.54	96.7	98.6	**88.3**	**0.93**	**3.6**	71.6	**100**	8.0s
	100	94.6	1.06	**1.08**	**98.2**	99.1	87.1	0.99	3.7	71.6	**100**	27.7s
	200	**94.7**	1.06	**1.08**	**98.2**	**99.2**	86.1	1.06	3.9	65.6	**100**	101.9s
FlashSSM-3D	10	83.3	**0.97**	5.6	78.0	89.8	86.8	1.10	5.9	58.5	91.2	**0.2s**
	20	86.3	1.02	3.33	85.7	96.8	87.1	1.14	5.3	**70.1**	92.4	**0.3s**
	50	90.3	0.99	1.79	94.1	97.6	87.2	1.08	4.3	66.2	93.0	**0.6s**
	100	92.4	1.00	1.50	95.6	97.2	**87.3**	**1.02**	**4.2**	67.6	**93.1**	**1.2s**
	200	**93.4**	1.00	**1.16**	**96.6**	**98.2**	87.1	**1.02**	**4.2**	67.6	**93.1**	**2.7s**

5 Conclusion

We introduce FlashSSM-3D, the first fully attention-less, feed-forward model for large-scale 3D reconstruction and camera pose estimation. Departing from existing pretrained transformer backbones, FlashSSM-3D explores whether SSMs can serve as a viable alternative architecture for dense multi-view geometric reasoning. By replacing self-attention with selective scan state-space layers, FlashSSM-3D achieves near-linear scaling complexity with respect to the number of input views while preserving strong geometric reasoning. However, this architectural shift introduces a challenging cross-architecture distillation problem. To address this, we propose a geometry-aware distillation framework combining Procrustes alignment and camera reprojection losses, enabling effective transfer of structural and camera pose consistency to a student model. Extensive experiments demonstrate that FlashSSM-3D achieves competitive performance while requiring significantly less data and compute, with substantial speedups when scaling to hundreds of views. These results suggest that attention-free, state-space architectures can provide a scalable path toward accurate feed-forward 3D reconstruction.

6 Acknowledgments

Supported by Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number 140D0423C0074. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

[1] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.

[2] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE transactions on robotics*, 37(6):1874–1890, 2021.

[3] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12786–12796, 2022.

[4] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022.

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

[6] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12716–12725, 2019.

[7] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE international conference on computer vision*, pages 873–881, 2015.

[8] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14194–14203, 2021.

[9] Yikang Ding, Wentao Yuan, Qingtian Zhu, Haotian Zhang, Xiangyue Liu, Yuanjiang Wang, and Xiao Liu. Transmvsnet: Global context-aware multi-view stereo network with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8585–8594, 2022.

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[11] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[12] 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.

[13] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024.

[14] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21936–21947, 2025.

[15] 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.

[16] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[17] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *The Fourteenth International Conference on Learning Representations*, 2026.

[18] 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. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21924–21935, 2025.

[19] 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.

[20] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

[21] You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. Fastvggt: Training-free acceleration of visual geometry transformer. *arXiv preprint arXiv:2509.02560*, 2025.

[22] Zipeng Wang and Dan Xu. Flashvggt: Efficient and scalable visual geometry transformers with compressed descriptor attention. *arXiv preprint arXiv:2512.01540*, 2025.

[23] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[24] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[25] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

[26] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[27] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *European conference on computer vision*, pages 237–255. Springer, 2024.

[28] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] John Oliensis. A critique of structure-from-motion algorithms. *Computer Vision and Image Understanding*, 80(2):172–214, 2000.

[30] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017.

[31] Shaohui Liu, Yidan Gao, Tianyi Zhang, Rémi Pautrat, Johannes L Schönberger, Viktor Larsson, and Marc Pollefeys. Robust incremental structure-from-motion with hybrid features. In *European Conference on Computer Vision*, pages 249–269. Springer, 2024.

[32] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building rome on a cloudless day. In *European conference on computer vision*, pages 368–381. Springer, 2010.

[33] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[34] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. In *2025 International Conference on 3D Vision (3DV)*, pages 78–89. IEEE, 2025.

[35] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21686–21697, 2024.

[36] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[37] Aviv Bick, Kevin Li, Eric Xing, J Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *Advances in Neural Information Processing Systems*, 37:31788–31812, 2024.

[38] Jim Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the kabsch-umeyama algorithm. *Journal of research of the National Institute of Standards and Technology*, 124:1, 2019.

[39] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018.

[40] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[41] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10901–10911, 2021.

[42] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

[43] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1790–1799, 2020.

[44] Weining Ren, Hongjun Wang, Xiao Tan, and Kai Han. Fin3r: Fine-tuning feed-forward 3d reconstruction models via monocular knowledge distillation. *arXiv preprint arXiv:2511.22429*, 2025.

[45] Jianyuan Wang, Christian Rupprecht, and David Novotny. Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9773–9783, 2023.

[46] Wenyan Cong, Yiqing Liang, Yancheng Zhang, Ziyi Yang, Yan Wang, Boris Ivanovic, Marco Pavone, Chen Chen, Zhangyang Wang, and Zhiwen Fan. E3d-bench: A benchmark for end-to-end 3d geometric foundation models. *arXiv preprint arXiv:2506.01933*, 2025.

[47] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[48] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2930–2937, 2013.

[49] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10986–10995, 2019.

[50] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[51] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[52] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5987–5997, 2021.

[53] Zhenggang Tang, Yuchen Fan, Dilin Wang, Hongyu Xu, Rakesh Ranjan, Alexander Schwing, and Zhicheng Yan. Mv-dust3r+: Single-stage scene reconstruction from sparse views in 2 seconds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5283–5293, 2025.

[54] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

A Appendix

A.1 Limitations

While FlashSSM-3D demonstrates that state space models can serve as effective backbones for feed-forward 3D reconstruction, several limitations remain. First, current deep learning frameworks and GPU kernels are still far more optimized for transformer attention than for selective scan operations, making SSM performance more sensitive to implementation details and backend support. Second, our model relies on VGGT as a teacher, meaning the quality of the distilled student is ultimately bounded by the geometric priors and failure modes of the teacher. However, we do believe that using stronger teachers such as π^3 [17] or DepthAnything 3 [54] could serve as promising future work. Third, although FlashSSM-3D substantially improves inference speed, its memory usage remains comparable to recent transformer-acceleration methods rather than consistently lower. Further GPU optimizations to drop peak memory usage below transformer models remains an important direction for future work.

A.2 Architecture Details

FlashSSM-3D follows the same overall backbone structure as VGGT, differing only in the sequence mixing mechanism. We retain the same image encoder and tokenization pipeline as VGGT, where each input image is encoded using a pretrained DINO backbone to produce visual tokens. In VGGT, these tokens are processed by a stack of 24 transformer-style blocks, each with one frame-wise

self-attention layer and one global self-attention, resulting in 48 layers. Following the subquadratic distillation framework of [37], we replace each self-attention layer in every VGGT block with a Mamba layer while keeping the remaining block components (e.g., normalization layers, linear projections, and residual connections) unchanged. This results in a one-to-one mapping between teacher and student blocks, allowing the student to mirror the representational structure of VGGT while replacing quadratic attention with linear-time state-space sequence mixing. All prediction heads (camera and pointmap heads) and other architectural details remain unchanged from VGGT.

A.3 Comparison with Contemporary Work

To further evaluate inference scalability, we compare FlashSSM-3D against very recent VGGT acceleration methods released contemporaneously with this work in Table A1. FastVGGT and FlashVGGT reduce the cost of VGGT through token and attention optimizations, respectively, while FlashSSM-3D departs from attention entirely by using state-space sequence mixing. Across all tested input sizes, FlashSSM-3D achieves the lowest wall-clock runtime, with the gap increasing as the number of views grows. For example, at 1000 input images, FlashSSM-3D runs in 11.93s compared to 38.10s for FlashVGGT and 79.31s for FastVGGT. FlashSSM-3D does not achieve the lowest peak memory, but its memory usage remains comparable to these optimized transformer baselines.

Table A2 extends this comparison to reconstruction accuracy, evaluating FastVGGT and FlashVGGT under the same protocol and datasets as Table 1. Neither method uses distillation: both keep VGGT's pretrained transformer weights and speed up inference by reducing the tokens passed to quadratic attention, with FastVGGT being training-free and FlashVGGT requiring fine-tuning of its descriptor-attention modules. FlashSSM-3D remains competitive with both accelerators while being the fastest method at every view count. At 50 views, all three methods sit slightly below VGGT, which remains the accuracy ceiling, and no single accelerator is best across all datasets. The picture changes as the number of views grows. Token-reduction methods lower VGGT's runtime constant but keep its quadratic scaling, whereas FlashSSM-3D changes the complexity class. At 300+ views, FlashSSM-3D runs faster than both FlashVGGT and FastVGGT, while attaining the lowest error and completion of the three on TUM-RGBD and NRGBD, and the best ScanNet completion of any method. FastVGGT remains stronger on 7-Scenes and on ScanNet error. These results support our main claim that geometry-aware distillation enables an attention-free SSM backbone that is not only accurate, but also practically efficient for dense multi-view reconstruction.

Table A1: Comparison of runtime and GPU memory consumption across different numbers of input images. Best results are in bold.

Metric	Model	200	400	600	800	1000
Time (s) ↓	FastVGGT [21]	6.45	16.63	32.19	52.01	79.31
	FlashVGGT [22]	4.05	9.84	17.25	26.44	38.10
	FlashSSM-3D	**2.76**	**4.80**	**7.17**	**9.55**	**11.93**
GPU Mem. (GB) ↓	FastVGGT [21]	19.34	32.66	45.97	59.29	72.60
	FlashVGGT [22]	**16.97**	**27.92**	**38.83**	**49.76**	**60.68**
	FlashSSM-3D	21.80	36.63	50.95	65.26	79.57

A.4 Additional Qualitative Results

We provide additional qualitative comparisons between VGGT and FlashSSM-3D in Figure A1, extending the results in Figure 3 of the main paper to more unseen scenes with a large number of input views. These scenes include both outdoor environments and cluttered indoor scenes with many objects. In some cases, FlashSSM-3D achieves a lower Chamfer distance than VGGT and covers more of the scene. Across these diverse dense reconstruction settings, FlashSSM-3D produces geometry that is visually comparable to its teacher. This further shows that our geometry-aware distillation framework transfers VGGT's 3D geometric understanding to an attention-free SSM.

Table A2: 3D reconstruction comparison with contemporaneous VGGT acceleration methods, following the same evaluation protocol as Table 1. Best results are in **bold**, with second best underlined. FlashSSM-3D is the fastest method at both view counts, and its accuracy advantage over the token-reduction accelerators grows with the number of views.

Method	TUM-RGBD		ScanNet		NRGBD		7-Scenes		DTU		Time
	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	Error ↓	Comp ↓	
Dense Views (50)											
VGGT [16]	**0.105**	**0.101**	**0.434**	**0.411**	**0.015**	**0.012**	**0.019**	0.032	**1.125**	**1.295**	∼9s
FastVGGT [21]	0.126	0.141	0.463	0.419	0.029	0.018	0.020	**0.027**	1.278	1.307	1.2s
FlashVGGT [22]	0.138	0.146	0.475	0.436	0.031	0.018	0.024	0.031	1.314	1.408	0.8s
FlashSSM-3D	0.124	0.140	0.466	0.423	0.028	0.024	0.020	0.036	1.22	1.38	**0.6s**
Extremely Dense Views (300+)											
VGGT [16]	**0.074**	**0.083**	**0.433**	0.406	**0.016**	**0.017**	**0.020**	**0.027**	-	-	∼130s
FastVGGT [21]	0.157	0.162	0.452	0.422	0.045	0.029	0.021	0.029	-	-	10.2s
FlashVGGT [22]	0.144	0.152	0.458	0.426	0.049	0.036	0.029	0.034	-	-	7.1s
FlashSSM-3D	0.100	0.092	0.458	**0.399**	0.043	0.026	0.028	0.034	-	-	**3.5s**

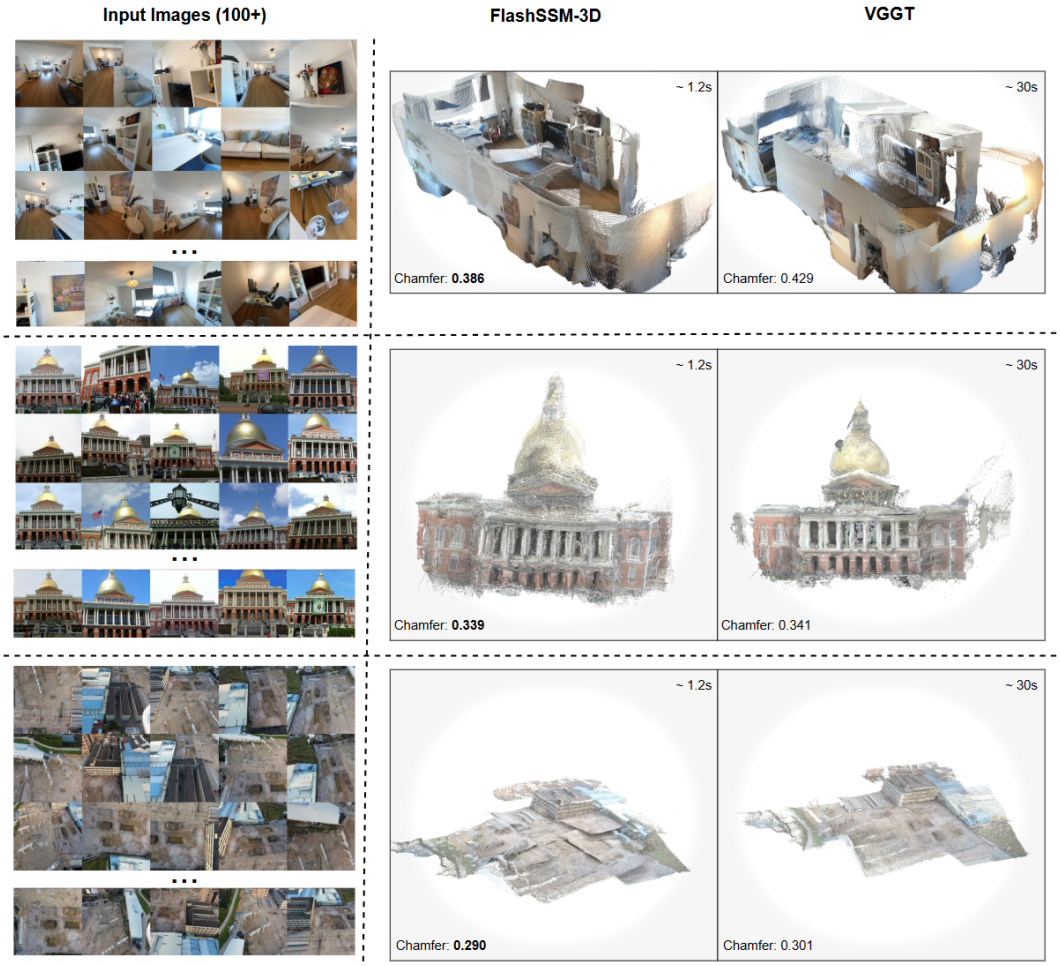


Figure A1: **Best viewed with zoom.** More qualitative results comparing VGGT and FlashSSM-3D on a large number of unseen images. In some cases, FlashSSM-3D performs better (lower Chamfer Distance) than VGGT with more coverage across the entire scene. This demonstrates that FlashSSM-3D has strongly recovered VGGT's 3D geometric understanding across diverse dense reconstruction scenarios, such as outdoor scenes and indoor scenes with many objects.

A.5 Training Settings

We provide additional details regarding the training configuration in Table A3. All stages are optimized using the AdamW optimizer with gradient clipping set to 1.0 and gradient accumulation

of 4 steps. Training proceeds in three stages with progressively smaller learning rates and longer training schedules. The effective batch size is kept consistent during the first two stages and slightly reduced in the final stage to accommodate a larger number of input views. Across stages, the number of views used during training is gradually increased, enabling the model to first learn stable representations from small view sets before scaling to more complex multi-view reasoning. During Stage 3, we additionally perform a short warmup where training begins at a resolution of 224×224 for approximately 5% of the training steps before switching to the full 518×336 resolution for the remainder of training. All experiments are conducted on NVIDIA H100 GPUs with 80GB memory. Training is implemented in PyTorch using distributed training with the NCCL backend. The student network is wrapped using Fully Sharded Data Parallel (FSDP) with the `FULL_SHARD` strategy to reduce GPU memory consumption.

Table A3: Optimization settings used in each training stage.

Setting	**Stage 1**	**Stage 2**	**Stage 3**
Optimizer	AdamW	AdamW	AdamW
Learning Rate	5×10^{-4}	2×10^{-4}	1×10^{-4}
Gradient Accumulation	4	4	4
Effective Batch Size	16	16	8
Number of Views	2-4	2-8	2-16
Gradient Clipping	1.0	1.0	1.0
Training Steps	50k	100k	200k
Image Resolution	224×224	224×224	224×224 / 518×336

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: We discuss training a state space model for 3D reconstruction, and our paper follows this main point throughout.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: A limitations section is provided in the Appendix.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: No deep theoretical results are presented in this work.

Guidelines:

- The answer [N/A] 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: There is an implementation details section in the appendix, with important details in the main text as well. Code is also provided with the submission.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: Code is provided with the submission.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Implementation details are in the appendix, with important details also in the main text.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: Confidence intervals and error bars are not commonly reported in this domain, hence we also choose to omit them.

   Guidelines:

   - The answer [N/A] 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: Discussions about compute and comparisons with existing works are present throughout the paper.

   Guidelines:

   - The answer [N/A] 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: There are no ethical issues surrounding our work.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: There are not major societal impacts surrounding our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There is little/no risk of our work being misused in a dangerous manner.

    Guidelines:

    - The answer [N/A] 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 datasets used are properly cited and allow free-use for research purposes.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: No new assets are produced in this work.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Core development does not include LLMs.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.