Generalizable 3D Foundation Models: Advancements in Geometry and View Synthesis

Shouchen Zhou
Tsinghua University
2025213446
zhousc25@mails.tsinghua.edu.cn

Abstract

Recent progress in 3D computer vision increasingly favors generalization across scenes instead of per-scene optimization. A growing number of works leverage large-scale pre-training and Transformer-based architectures to support multiple 3D tasks through data-driven inference, reducing reliance on per-scene fitting and handcrafted geometric pipelines. This report reviews the designs of such methods, emphasizing how supervision signals, intermediate representations, and the degree of explicit 3D inductive bias shape both geometric reconstruction and novel view synthesis. We further discuss current limitations and highlight open directions toward unified 3D backbones that jointly support reconstruction and rendering.

1 Introduction

A long-standing goal in 3D computer vision is to infer scene structure and synthesize novel observations from limited visual measurements. Classical pipelines achieve this goal by combining explicit projective geometry with multi-view constraints, while recent methods often rely on per-scene optimization to obtain high-fidelity reconstructions. Despite strong performance under per-scene settings, these paradigms face persistent obstacles when deployed at scale: sensitivity to camera calibration and pose accuracy, limited transfer across scenes, and substantial computational cost when adapting to new environments. These issues become critical as modern applications demand 3D reasoning and rendering across diverse, unconstrained data distributions.

The recent emergence of large pre-trained models has renewed interest in generalization for 3D vision. These changes indicated that the geometric and photometric regularities can be captured as learned priors from data, rather than being enforced exclusively through hand-designed constraints. However, the extent to which geometry can be learned implicitly remains unclear. In particular, current methods reveal a fundamental tension between the amount and form of supervision, the intermediate representation used to couple views and predict 3D quantities, and the degree of explicit 3D inductive bias retained to ensure consistency and controllability.

This report surveys representative progress from 2024-2025 through four recent methods. For geometric reconstruction, we focus on DUSt3R [1] and VGGT [2], which pursue generalizable geometry prediction via learned multi-view reasoning and direct 3D regression. For novel view synthesis, we review LVSM [3] and RayZer [4], which move toward photorealistic synthesis with reduced explicit 3D modeling and, in some cases, weaker 3D supervision. Together, these works provide a compact but informative slice of the emerging design space for general-purpose 3D backbones.

Rather than treating each paper in isolation, we aim to analyze the motivation, core design choices, and key contributions of each method, and then synthesize them under a unified comparison, thereby clarifying which ingredients appear necessary for robust 3D inference and high-quality synthesis.

Report of 84761063 Deep Learning Final Project.

2 Background and Preliminaries

For 3D computer vision tasks, most methods follow the same paradigm: multi-view images provide 2D observations of a shared 3D scene, and camera geometry maps pixels to 3D viewing rays. The goal is to infer scene structure and appearance that is consistent across views, enabling reconstruction and novel view synthesis under a unified ray-based formulation.

2.1 Problem Setting and Notation

We consider a set of N images $\{I_i\}_{i=1}^{N}$ observing a static scene. For image I_i with resolution $H_i \times W_i$, a pixel is denoted by $\mathbf{p} = (u, v)^\top$ and its homogeneous form by $\tilde{\mathbf{p}} = (u, v, 1)^\top$. Under the pinhole camera model (Fig. 1), each pixel corresponds to a unique viewing ray in 3D: intuitively, light travels from a 3D point through the optical center to the image plane.

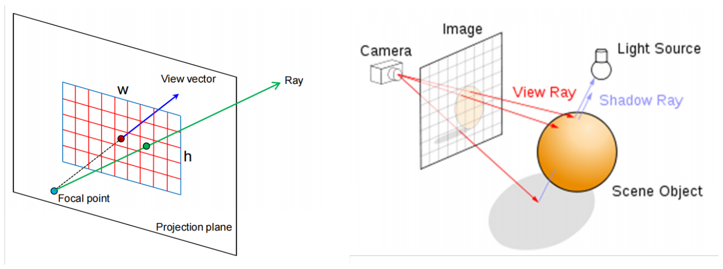

Figure 1: Pinhole camera geometry: a pixel on the image plane of size $W \times H$ corresponds to a unique viewing direction, defining a ray that starts from the camera center and then passes through the pixel on the projection plane.

The ray formulation makes explicit the two ingredients needed to relate a 2D measurement to 3D space. First, we must determine the ray direction in the camera coordinate system, which is governed by the camera intrinsics K. Second, we must express this ray in a world frame, which requires the camera extrinsics (R, t). This decomposition motivates the following notation for intrinsics and extrinsics.

Camera intrinsics. The intrinsic matrix $K_i \in \mathbb{R}^{3\times 3}$ for the i-th image maps normalized camera coordinates to pixel coordinates, with parameters:

$$K_i = \begin{bmatrix} f_{x,i} & s_i & c_{x,i} \\ 0 & f_{y,i} & c_{y,i} \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where $(f_{x,i}, f_{y,i})$ are focal lengths in pixel units, $(c_{x,i}, c_{y,i})$ is the principal point, and s_i is the skew. Using K_i, we can convert a pixel $\tilde{\mathbf{p}}$ into a direction on the normalized image plane (Fig. 2) via $K_i^{-1}\tilde{\mathbf{p}}$.

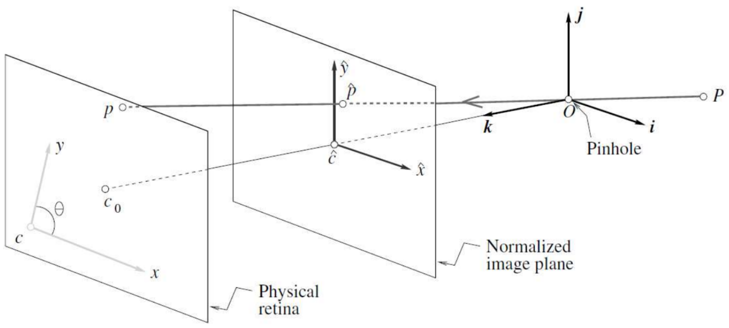

Figure 2: The pinhole projection model and the normalized image plane. A 3D point is projected through the optical center onto the image plane; by factoring out focal length and principal point, one obtains normalized image coordinates that simplify ray back-projection.

Camera extrinsics and coordinate frames. Let $\mathbf{X}^w \in \mathbb{R}^3$ denote a 3D point in the world coordinate system. Each camera i is associated with extrinsics (R_i, t_i) that transform world coordinates into camera coordinates (Fig. 3):

$$\mathbf{X}_i^c = R_i \mathbf{X}^w + t_i, \quad R_i \in SO(3), t_i \in \mathbb{R}^3. \tag{2}$$

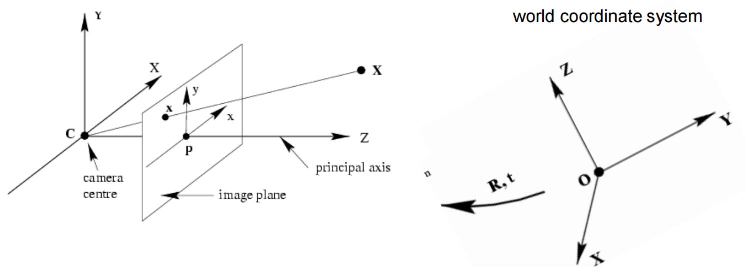

Figure 3: Coordinate frames and camera extrinsics. The camera coordinate system is related to the world coordinate system by a rigid transform (R, t); together with intrinsics K, this defines the mapping between 3D world points and 2D image measurements.

Combine the camera intrinsics and extrinsics all together, a point in the world coordinate system can be expressed as a homogeneous point in the camera coordinate system via

$$\lambda \tilde{\mathbf{p}} = K_i [R_i | t_i] \begin{bmatrix} \mathbf{X}^w \\ 1 \end{bmatrix}, \tag{3}$$

where λ is the depth of the corresponding point.

Accurate camera intrinsics and extrinsics are central to most multi-view 3D formulations: intrinsics determine how pixels map to viewing directions, while extrinsics align rays from different images into a shared 3D coordinate frame. When these parameters are know, or reliably estimated, the multi-view problem becomes one of finding a scene representation whose rendered projections along rays match the observed images.

2.2 Per-scene optimization

This ray-consistency principle underlies two influential per-scene optimization families: Neural Radiance Fields (NeRF [5]) and 3D Gaussian Splatting [6]. Both methods assume a posed image set and optimize a scene-specific representation so that rendering from each camera reproduces the corresponding view, yielding high-fidelity novel view synthesis, with given camera intrinsics and extrinsics.

2.2.1 NeRF: Neural Radiance Fields

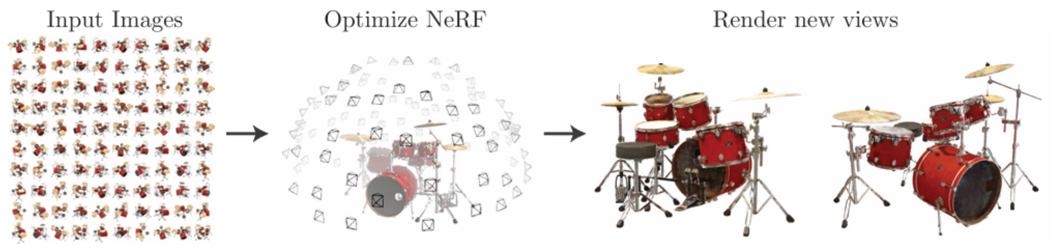

Figure 4: Per-scene optimization pipeline of NeRF. Given posed input images, NeRF optimizes a scene-specific radiance field by enforcing photometric consistency, enabling high-quality rendering from novel viewpoints.

NeRF's follows a per-scene optimization pipeline (Fig. 4): given a set of posed multi-view images(known cameras' intrinsic and extrinsics), rays are sampled from the input views, rendered through the current radiance field, and compared against ground-truth pixel colors using a rendering loss; gradients are backpropagated through the volume rendering equation to update θ. After convergence, the learned radiance field can synthesize photorealistic novel views from unseen camera poses within the same scene. This formulation yields high-quality results, but it typically requires optimizing a separate model for each scene and relies on accurate camera calibration and pose estimation.

For details, NeRF models a scene as a continuous radiance field parameterized by a neural network(Fig. 5). Concretely, an MLP F_θ takes as input a 3D location $\mathbf{x} \in \mathbb{R}^3$ and a 2D viewing direction $\mathbf{d}$, and outputs a volume density $\sigma(\mathbf{x})$ together with a view-dependent color $\mathbf{c}(\mathbf{x}, \mathbf{d})$.

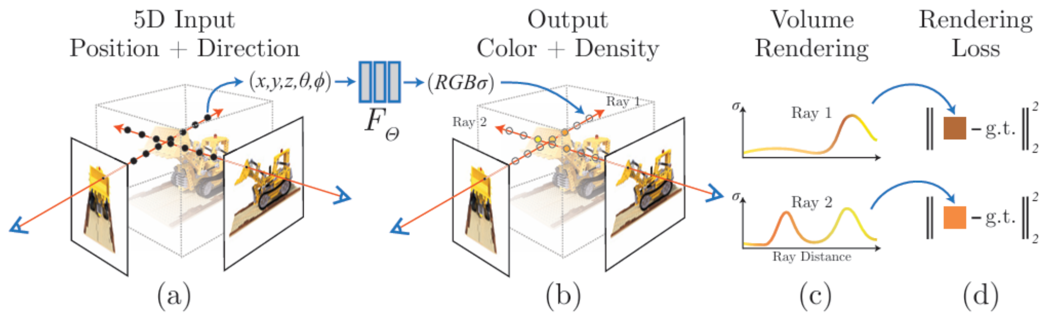

Figure 5: NeRF parameterization and training objective. An MLP takes 5D coordinates (3D position and viewing direction) as input and predicts color and density; differentiable volume rendering produces pixel colors that are matched to ground-truth images with a rendering loss.

To render a pixel, NeRF casts the corresponding camera ray and samples a set of points along it; the network predictions $(\sigma_i, \mathbf{c}_i)$ at these samples are then composited via differentiable volume rendering, where density determines the transmittance/opacity and colors are accumulated to form the final pixel value (Fig. 6).

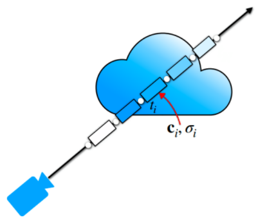

Figure 6: NeRF volume rendering along a camera ray. Points are sampled on a viewing ray and mapped to density σ_i and color $\mathbf{c}_i$; these samples are composited to produce the final pixel color via volumetric integration.

For details, a camera ray parameterized as

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, \qquad t \in [t_n, t_f], \tag{4}$$

With NeRF's network's outputs volume density $\sigma(\mathbf{x})$ and a view-dependent color $\mathbf{c}(\mathbf{x}, \mathbf{d})$ and the background color $\mathbf{c}_{\text{bg}}$. The rendered pixel color along the ray is:

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\mathrm{d}t + T(t_f)\mathbf{c}_{\text{bg}}, \tag{5}$$

where $T(t)$ is the accumulated transmittance (probability that the ray has not terminated before t),

$$T(t) = \exp\left(- \int_{t_n}^{t} \sigma(\mathbf{r}(s))\mathrm{d}s \right). \tag{6}$$

In practice, NeRF uses a discrete approximation with samples $\{t_i\}_{i=1}^N$ along the ray and intervals $\delta_i = t_{i+1} - t_i$. Let $\sigma_i = \sigma(\mathbf{r}(t_i))$ and $\mathbf{c}_i = \mathbf{c}(\mathbf{r}(t_i), \mathbf{d})$. Define

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i), \qquad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) = \prod_{j=1}^{i-1}(1 - \alpha_j). \tag{7}$$

Then the rendered color is

$$\mathbf{C}(\mathbf{r}) \approx \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i + T_{N+1}\mathbf{c}_{\text{bg}}, \qquad T_{N+1} = \exp\left(-\sum_{j=1}^{N} \sigma_j \delta_j\right). \tag{8}$$

2.2.2 3D Gaussian Splatting

NeRF and its subsequent methods share a common principle: given posed multi-view images, optimize a scene representation **implicitly** so that differentiable rendering reproduces the observations. Which cause to a significantly high computational complexity while rendering. Thus, 3D Gaussian Splatting adopts an **explicit** collection of 3D Gaussians rendered by differentiable splatting or rasterization, significantly improving efficiency without changing the overall per-scene optimization paradigm.

3D Gaussian Splatting (3DGS) represents a scene as an explicit set of 3D Gaussian primitives. Each Gaussian typically carries a 3D mean, a covariance (or scale and orientation) controlling its spatial extent, an opacity, and appearance parameters (e.g., color or learned features). Compared to NeRF's implicit volumetric field, 3DGS adopts a more graphics-friendly, explicit representation: for a given camera view, Gaussians are projected onto the image plane as 2D elliptical splats and composited via a differentiable rasterizer to produce the final rendering. Fig. 7 is an example to show how 3DGS can explicitly represent a scene.

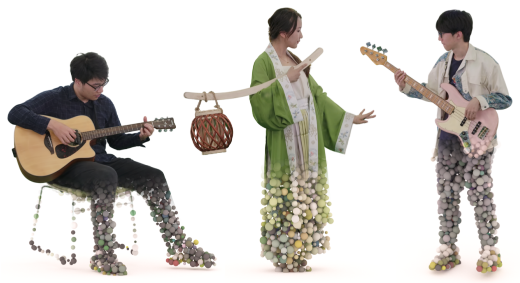

Figure 7: 3D Gaussian Splatting scene representation. A scene is modeled as a set of 3D Gaussians; the visualization shows the correspondence between the photorealistic rendering and the underlying Gaussian primitives (up/lower parts).

In practice, 3DGS is commonly trained in a per-scene optimization regime(Fig. 8). As illustrated by the pipeline, a structure-from-motion (SfM) system provides camera poses and sparse points to initialize the Gaussian set. The Gaussians are then optimized by minimizing an image reconstruction objective: rendered images from the current Gaussian representation are matched to posed multi-view observations, and gradients are backpropagated through the differentiable rasterization process to update both geometry and appearance parameters.

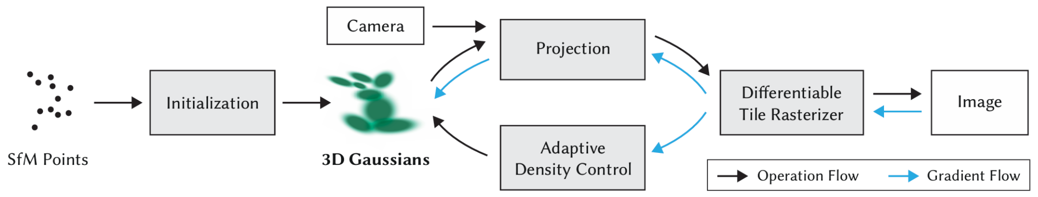

Figure 8: Per-scene optimization pipeline of 3D Gaussian Splatting. Sparse SfM points initialize 3D Gaussians, which are optimized under differentiable rasterization: projections to the image plane and density control enable efficient training and rendering.

3D Gaussian Splatting models a scene as an explicit set of 3D Gaussian primitives,

$$\mathcal{G} = \{(\boldsymbol{\mu}_i, \Sigma_i, \alpha_i, \mathbf{c}_i)\}_{i=1}^{N}, \tag{9}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is the Gaussian mean to represent position, Σ_i encodes its 3D shape and orientation, α_i is an opacity parameter, and $\mathbf{c}_i$ denotes color or learned appearance features. Rendering proceeds by projecting each 3D Gaussian into screen space under a camera with matrix M. Let $\Pi(\cdot)$ be the perspective projection and J_i the Jacobian of Π at $\boldsymbol{\mu}_i$. The projected mean and covariance can be written as

$$\boldsymbol{\mu}'_i = \Pi(M\boldsymbol{\mu}_i), \qquad \Sigma'_i = J_i M \Sigma_i M^\top J_i^\top \tag{10}$$

Each projected Gaussian contributes a 2D elliptical weight at pixel location $\mathbf{p}$:

$$G_i(\mathbf{p}) = \exp\Big(-\tfrac{1}{2}(\mathbf{p} - \boldsymbol{\mu}'_i)^\top \Sigma_i'^{-1}(\mathbf{p} - \boldsymbol{\mu}'_i)\Big). \tag{11}$$

Using front-to-back compositing, the effective alpha at $\mathbf{p}$ is typically expressed as $\tilde{\alpha}_i(\mathbf{p}) = \alpha_i G_i(\mathbf{p})$, and the rendered color is

$$\mathbf{C}(\mathbf{p}) = \sum_i T_i(\mathbf{p})\tilde{\alpha}_i(\mathbf{p})\mathbf{c}_i, \qquad T_i(\mathbf{p}) = \prod_{j<i}\left(1 - \tilde{\alpha}_j(\mathbf{p})\right), \tag{12}$$

A key component enabling high quality under limited compute is adaptive density control(Fig. 9). During training, the method dynamically increases capacity where necessary by cloning or splitting Gaussians in regions that require finer detail, while pruning or regularizing redundant primitives. This strategy refines the representation over time and helps balance rendering quality and efficiency.

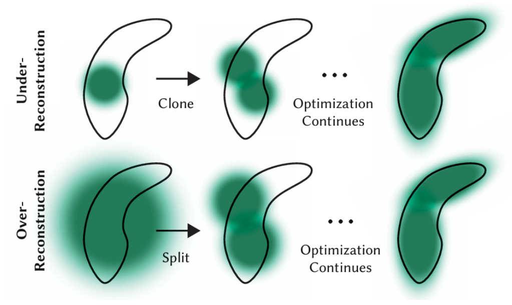

Figure 9: Adaptive density control in 3D Gaussian Splatting. During optimization, Gaussians are cloned or split to increase representational capacity where needed, improving reconstruction quality while maintaining efficiency.

Overall, 3DGS offers fast training and real-time rendering potential, but it still relies on accurate camera poses and typically requires optimizing a separate representation for each new scene, motivating the shift toward generalizable 3D foundation models.

3 Generalizable Geometric Reconstruction

While NeRF and 3D Gaussian Splatting demonstrate that photometric consistency under differentiable rendering can produce high-fidelity 3D representations, they fundamentally operate in a per-scene optimization regime. In practice, each new scene requires its own optimization, and performance depends heavily on accurate camera calibration and poses. These constraints limit scalability: the computational cost grows with the number of scenes, and robustness can degrade when camera metadata is noisy, incomplete, or unavailable.

This motivates a different objective: generalizable geometric reconstruction, where a model is trained once on large-scale data and then applied to unseen scenes via a single forward pass or minimal adaptation. Such general models amortize the cost of 3D inference across scenes, enabling efficient deployment at scale. They can also leverage diverse training distributions to learn transferable priors, improving robustness in different settings. Also, these methods treat reconstruction as data-driven inference rather than a per-scene fitting, they open the door to reduced reliance on carefully engineered pipelines. With this perspective, we next review two representative approaches DUSt3R and VGGT that exemplify the recent move toward generalizable geometric reconstruction.

3.1 DUSt3R

Classical 3D reconstruction typically follows a multi-stage structure from motion(SfM) or multiview stereo(MVS) pipeline: feature matching, minimal solvers, triangulation, pose estimation, and dense reconstruction. This sequential design has errors accumulation across stages, and the entire pipeline depends critically on accurate camera intrinsics and extrinsics, which are often unavailable or noisy in unconstrained photo collections. DUSt3R(Dense and Unconstrained Stereo 3D Reconstruction) proposes to invert this dependency: instead of first recovering cameras to enable triangulation, it directly regresses a dense 3D representation from image pairs without requiring calibration or known poses.(Fig 10)

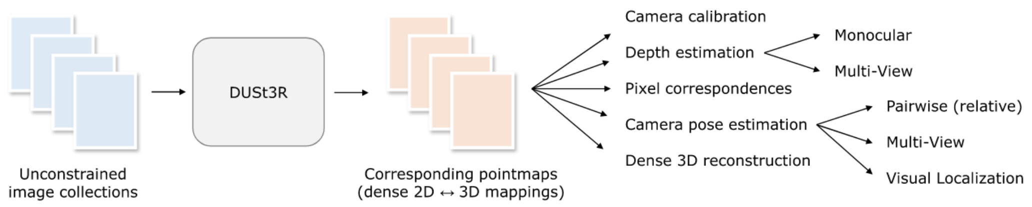

Figure 10: DUSt3R overview: from unconstrained image collections to dense pointmaps. DUSt3R predicts view-consistent pointmaps from uncalibrated, unposed images, enabling a range of downstream geometric tasks such as depth estimation, correspondence matching, camera calibration, pose estimation, and dense 3D reconstruction.

3.1.1 DUST3R's pipeline

DUSt3R introduces pointmaps as its core output. A pointmap is a dense 2D field of 3D points

$$\mathbf{X} \in \mathbb{R}^{W \times H \times 3}, \tag{13}$$

in one-to-one correspondence with image pixels, i.e., $I_{i,j} \leftrightarrow \mathbf{X}_{i,j}$. When camera intrinsics $\mathbf{K}$ and depth D are available (e.g., for supervision), the pointmap in the camera frame can be written as

$$\mathbf{X}_{i,j} = \mathbf{K}^{-1}[iD_{i,j}, jD_{i,j}, D_{i,j}]^{\top}. \tag{14}$$

For two views n, m, the same underlying geometry can be expressed across coordinate frames via

$$\mathbf{X}_{n,m} = \mathbf{P}_m \mathbf{P}_n^{-1} h(\mathbf{X}_n), \tag{15}$$

where $\mathbf{P}$ denotes a world-to-camera pose and $h(\cdot)$ converts points to homogeneous coordinates. DUSt3R leverages this view-coupled nature of pointmaps, but crucially does *not* require $\mathbf{K}$ or $\mathbf{P}$ at inference time.

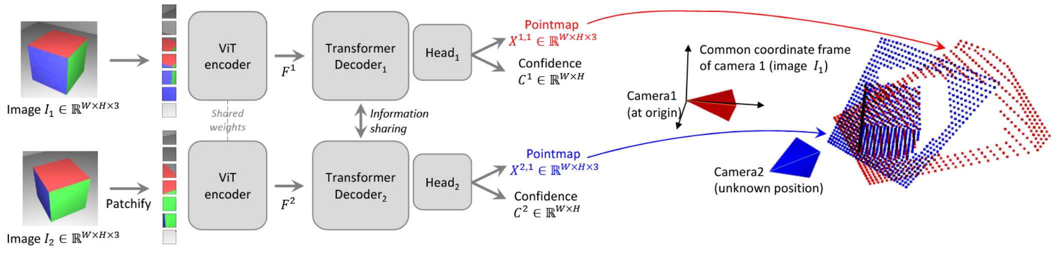
Image $I_1 \in \mathbb{R}^{W\times H\times 3}$
Image $I_2 \in \mathbb{R}^{W\times H\times 3}$
Patchify
ViT encoder
Shared weights
ViT encoder
F^1
F^2
Transformer Decoder$_1$
Information sharing
Transformer Decoder$_2$
Head$_1$
Head$_2$
Pointmap $X^{1,1} \in \mathbb{R}^{W\times H\times 3}$
Confidence $C^1 \in \mathbb{R}^{W\times H}$
Pointmap $X^{2,1} \in \mathbb{R}^{W\times H\times 3}$
Confidence $C^2 \in \mathbb{R}^{W\times H}$
Common coordinate frame of camera 1 (image I_1)
Camera1 (at origin)
Camera2 (unknown position)

Figure 11: DUSt3R architecture and pointmap prediction in a common frame. A ViT encoder extracts features from an image pair, followed by Transformer decoders with cross-view information exchange and regression heads that output pointmaps and per-pixel confidence maps. Both predicted pointmaps are expressed in the coordinate frame of the first camera, enabling direct alignment without requiring known camera poses.

For its network(Fig 11), given two RGB images (I_1, I_2), DUSt3R predicts two pointmaps and confidence maps

$$(I_1, I_2) \mapsto (\mathbf{X}_{1,1}, \mathbf{C}_{1,1}), (\mathbf{X}_{2,1}, \mathbf{C}_{2,1}), \tag{16}$$

where both pointmaps are expressed in a common coordinate frame tied to the first image, i.e. regard I_1 as the canonical space. This design relaxes strict projective constraints and allows the model to encode the pixel-to-3D mapping and inter-view relations directly in its outputs.

For the architectural, DUSt3R uses a ViT encoder with shared weights to embed both images into token features, followed by two Transformer decoders that repeatedly exchange information through cross-attention. Two regression heads then output dense pointmaps and per-pixel confidence scores. The architecture is inspired by CroCo[7]-style cross-view completion, enabling effective use of strong Transformer pretraining.

3.1.2 Training objective

DUSt3R's training is fully supervised with a simple 3D regression loss. Because the predicted reconstruction is scale-ambiguous, DUSt3R normalizes both prediction and ground truth by global scale factors. Let $\mathcal{D}_v$ be the set of valid pixels in view $v \in \{1, 2\}$, and define the normalization

$$\operatorname{norm}(\mathbf{X}_1, \mathbf{X}_2) = \frac{1}{|\mathcal{D}_1| + |\mathcal{D}_2|} \sum_{v \in \{1,2\}} \sum_{i \in \mathcal{D}_v} \|\mathbf{X}_{v,i}\|. \tag{17}$$

Then the per-pixel regression term is

$$\ell_{\text{regr}}(v, i) = \left\| \frac{1}{z} \mathbf{X}_{v,1,i} - \frac{1}{\bar{z}} \bar{\mathbf{X}}_{v,1,i} \right\|, \tag{18}$$

where $z = \operatorname{norm}(\mathbf{X}_{1,1}, \mathbf{X}_{2,1})$ and $\bar{z} = \operatorname{norm}(\bar{\mathbf{X}}_{1,1}, \bar{\mathbf{X}}_{2,1})$. To handle ill-defined regions and varying difficulty, DUSt3R predicts confidence $\mathbf{C}$ and optimizes a confidence-weighted objective

$$\mathcal{L}_{\text{conf}} = \sum_{v \in \{1,2\}} \sum_{i \in \mathcal{D}_v} \mathbf{C}_{v,1,i} \ell_{\text{regr}}(v, i) - \alpha \log \mathbf{C}_{v,1,i}, \tag{19}$$

3.1.3 From pointmaps to geometric tasks

The predicted pointmaps are rich enough to support multiple downstream tasks with minimal extra machinery:

- Depth: can be read from the z-coordinate of the pointmap in each view.
- Dense correspondences: can be obtained by nearest-neighbor search in 3D pointmap space, optionally keeping mutual matches for robustness.
- Intrinsics: can be estimated by fitting a focal length under mild assumptions: centered principal point, square pixels.
- Relative pose: can be recovered via alignment between pointmaps, or more robustly via RANSAC PnP [8].

Importantly, these quantities requires only a single feed-forward model rather than prerequisites for reconstruction.

3.1.4 Summary

DUSt3R is the first to push 3D vision from a per-scene geometric-optimization paradigm to a general inference paradigm: by training a generic Transformer at scale with a simple 3D regression objective, it amortizes geometric reasoning into the network, enabling direct operation on uncalibrated, unposed photo collections without re-running a full SfM/MVS optimization for each scene.

3.2 VGGT

While DUSt3R operates primarily on image pairs and typically relies on additional post-processing to fuse many views into a coherent reconstruction, which becomes a bottleneck when the number of images grows and when global consistency is required. VGGT($\underline{\text{V}}$isual $\underline{\text{G}}$eometry $\underline{\text{G}}$rounded

Transformer) takes the next step: it aims to predict all key 3D attributes of a scene from one to hundreds of views in a single feed-forward pass, significantly reducing reliance on geometric optimization while improving scalability and robustness in unconstrained settings.

3.2.1 Formulation

VGGT is a large feed-forward transformer trained on diverse 3D-annotated data to jointly infer multiple, interrelated 3D quantities for an image set. Given N RGB images of the same scene, it directly predicts per-view camera parameters, depth maps, point maps, and dense features for point tracking in one forward pass:

$$f\left((I_i)_{i=1}^{N}\right) = (g_i, D_i, P_i, T_i)_{i=1}^{N}. \tag{20}$$

Here the notions are the rotation $g_i \in \mathbb{R}^9$, the translation $t \in \mathbb{R}^3$, the field-of-view $f \in \mathbb{R}^2$ (assuming the principal point at the image center); the depth $D_i \in \mathbb{R}^{H\times W}$; the dense pointmap $P_i \in \mathbb{R}^{3\times H\times W}$; and the dense tracking features used to recover point tracks T_i. Similar to DUSt3R, VGGT defines pointmaps in the coordinate system of the first camera as canonical space, which acts as the world reference frame, but it does not requires the point map for other representations in DUSt3R, but directly output the regression result by the network.

3.2.2 Architecture

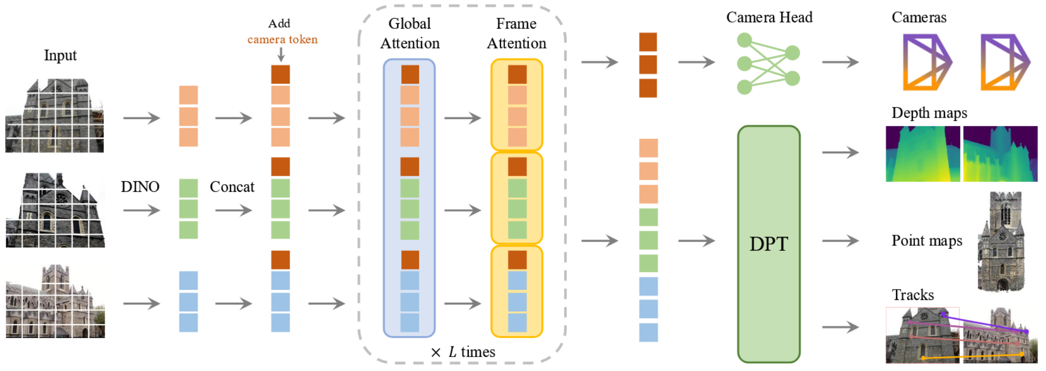

Figure 12: VGGT pipeline for feed-forward multi-view geometry. Given a set of input images, VGGT extracts per-image tokens (e.g., via a ViT/DINO backbone), augments them with a camera token, and performs repeated global and frame-level attention to aggregate multi-view information. Task-specific heads then predict cameras and dense geometric outputs such as depth maps and point maps, as well as tracks for establishing correspondences across views.

VGGT's architectural(Fig. 12) first patchifies each image into tokens using a DINOv2[9]-based visual encoder, then appends a **camera token** for camera prediction and several **register tokens** to stabilize and enrich global reasoning. The core backbone is a standard large transformer, but with a key modification called Alternating-Attention (AA): it alternates between frame-wise self-attention (attention within each image separately) and global self-attention (attention across tokens from all images jointly), repeated for L layers (default $L = 24$). Importantly, VGGT uses only self-attention, no explicit cross-attention, aiming for minimal handcrafted 3D inductive bias while still enabling strong multi-view fusion.

With such architectural,VGGT is trained with a multi-task loss combining camera, depth, pointmap, and tracking supervision:

$$\mathcal{L} = \mathcal{L}_{\text{camera}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{pmap}} + \lambda\mathcal{L}_{\text{track}}, \quad \lambda = 0.05, \tag{21}$$

where depth/pointmap losses incorporate predicted uncertainty and also include gradient-based regularization (commonly used in depth estimation). To resolve global similarity ambiguities, the ground-truth data are expressed in the first-camera frame and scale-normalized using the average distance of scene points to the origin; unlike some prior work, VGGT does not necessarily normalize the network predictions during training, but instead trains the model to internalize the chosen canonicalization.

3.2.3 Summary

VGGT is designed to be fast and directly usable: it predicts cameras and geometry for many views in a feed-forward regime. Beyond reconstruction, the pretrained VGGT features are shown to transfer to downstream tasks such as point tracking and feed-forward novel view synthesis.

VGGT is surprisingly 'simple' on the surface: starting from strong visual tokens, it mainly adds camera and register tokens and relies on attention-based aggregation to predict cameras and dense geometry, without explicit geometric modules or iterative solvers. This simplicity suggests that much of multi-view reasoning can be amortized into a generic Transformer when trained at scale, and the main gain may come from the alternating attention patternglobal attention to exchange information across frames, followed by frame-wise attention. This design insight is also reflected in later work such as RayZer.

3.3 Overall

DUSt3R and VGGT mark an important step toward generalizable 3D vision: instead of solving geometry with hand-crafted pipelines or optimizing a separate NeRF or 3DGS for per-scene settings, they amortize multi-view reconstruction into a feed-forward Transformer trained at scale. DUSt3R is flexible and effective in the pairwise, unconstrained setting but often needs an additional global alignment stage to scale to many views, whereas VGGT performs direct multi-view aggregation and jointly predicts cameras and dense geometry in one pass, offering better scalability at the cost of a heavier multi-view model.

Despite these advances, both methods still primarily rely on large amounts of labeled or pseudo-labeled geometric supervision (e.g., posed multi-view data with depth/points/poses) during training, indicating that the current generation of general 3D backbones remains largely within a supervised learning regime.

4 Novel View Synthesis with Minimal 3D Inductive Bias

While DUSt3R and VGGT demonstrate that geometry can be predicted in a generalizable, feed-forward manner, accurate 3D structure alone does not automatically translate into photorealistic rendering. In practice, novel view synthesis must model appearance effects that are only weakly constrained by geometryocclusions, view-dependent reflectance, specularities, and lighting changesespecially under wide-baseline extrapolation. Moreover, enforcing strong explicit 3D inductive bias such as strict geometric consistency and carefully engineered rendering pipelines can improve controllability but often reintroduces sensitivity to camera accuracy and limits scalability.

4.1 LVSM

This motivates a complementary line of work: LVSM (Large View Synthesis with Minimal 3D Inductive Bias). Instead of committing to a specific geometric representation and optimizing it per scene, these methods aim to learn transferable priors for rendering directly from large-scale multi-view data, using architectures that can generalize across scenes and viewpoints.

Most high-quality novel view synthesis(NVS) systems rely on explicit 3D representations and rendering pipelines such as NeRF's volume rendering or 3DGS' splatting, which impose strong 3D inductive bias and often require per-scene optimization. LVSM argues that high-fidelity view synthesis can instead be learned as a data-driven mapping with minimal explicit 3D bias: avoid hand-crafted 3D scene representations and let a large Transformer learn multi-view fusion directly from large-scale posed data.

4.1.1 Plücker rays

Plücker rays are the keys to LVSM's success. A 3D viewing ray can be represented in multiple equivalent forms. Besides the common origin + direction parameterization $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, a particularly convenient, coordinate-consistent representation is given by the Plücker coordinates: a directed line in $\mathbb{R}^3$ is encoded by a 6D vector

$$\boldsymbol{\ell} = (\mathbf{d}, \mathbf{m}) \in \mathbb{R}^6, \tag{22}$$

where $\mathbf{d} \in \mathbb{R}^3$ is the line normalized direction, and $\mathbf{m} \in \mathbb{R}^3$ is the moment defined as

$$\mathbf{m} = \mathbf{p} \times \mathbf{d}, \tag{23}$$

with $\mathbf{p}$ being any point on the line. The moment is invariant to the choice of $\mathbf{p}$ along the line: if $\mathbf{p}' = \mathbf{p} + t\mathbf{d}$, then $\mathbf{p}' \times \mathbf{d} = \mathbf{p} \times \mathbf{d}$ since $\mathbf{d} \times \mathbf{d} = \mathbf{0}$. Therefore, $(\mathbf{d}, \mathbf{m})$ uniquely specifies a line up to scale, subject to the Plücker constraint $\mathbf{d}^\top \mathbf{m} = 0$.

For a camera with extrinsics (R, t) and intrinsics K, a pixel $\tilde{\mathbf{u}} = (u, v, 1)^\top$ defines a ray. First obtain a direction in the camera frame, $\mathbf{d}_c \propto K^{-1}\tilde{\mathbf{u}}$, and transform it to the world frame:

$$\mathbf{d} = R^\top \mathbf{d}_c \tag{24}$$

The camera center in world coordinates is $\mathbf{C} = -R^\top t$, which lies on the ray. The corresponding Plücker moment is

$$\mathbf{m} = \mathbf{C} \times \mathbf{d}. \tag{25}$$

Thus, each pixel can be mapped to a 6D Plücker embedding $(\mathbf{d}, \mathbf{m})$, producing a dense ray map that encodes both camera pose and pixel direction.

Plücker coordinates provide a translation-aware line representation: while the direction $\mathbf{d}$ captures viewing orientation, the moment $\mathbf{m}$ encodes the ray's position relative to the origin. This makes $(\mathbf{d}, \mathbf{m})$ well-suited as a geometric conditioning signal for Transformers, since the model can reason about multi-view relationships through ray geometry without explicitly constructing 3D volumes, cost volumes, or performing analytical rendering.

In LVSM (and later RayZer-style models), Plücker ray maps serve as a minimal yet expressive way to inject camera geometry into a purely token-based, feed-forward view synthesis pipeline.

4.1.2 Architecture

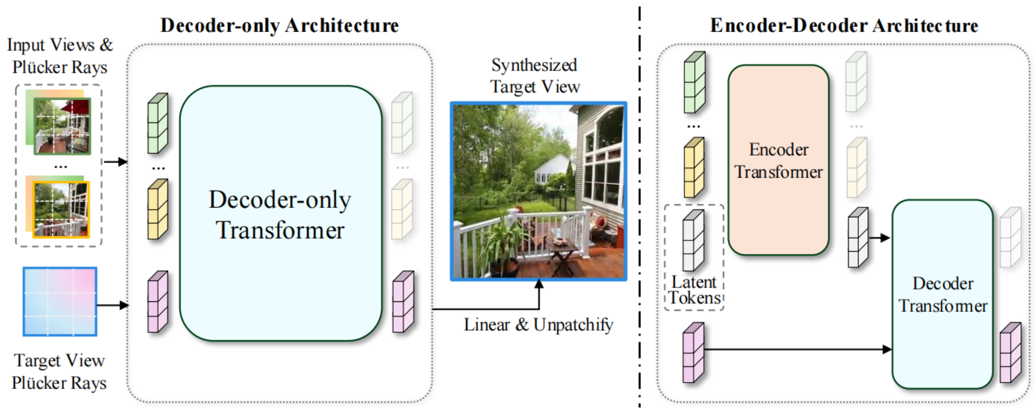

Figure 13: LVSM pipeline and architectural variants. LVSM conditions a Transformer on input views together with Plücker-ray embeddings, and synthesizes a target view by decoding target-ray tokens. The figure contrasts a decoder-only design (directly mapping input-view and target-ray tokens to the rendered image) with an encoder–decoder variant that compresses multi-view information into latent tokens before decoding.

As the LVSM's architecture shows in Fig. 13, LVSM encodes camera geometry through per-pixel Plücker rays. For each input view, it concatenates RGB patches with the corresponding ray-embedding patches to form input tokens; for the target view, it forms query tokens from target-ray embeddings only. A Transformer maps the multi-view input tokens to target tokens, which are linearly decoded and unpatchified to produce the target image. This design keeps the model geometry-aware through rays, while avoiding explicit 3D volumes or meshes or analytical rendering equations, leading to the 'minimal 3D inductive bias'.

4.1.3 Overall

LVSM is trained end-to-end with standard image reconstruction losses. LVSM demonstrates that large Transformers with ray conditioning can achieve strong photorealistic NVS without committing to a specific explicit 3D representation, but it still relies on posed multi-view supervision and can struggle under extreme extrapolation or out-of-distribution camera or image settings.

4.2 RayZer

Unfortunately, despite their strong generalization at test time, recent "general" 3D models such as DUSt3R, VGGT, and LVSM are still largely trained in a supervised manner, relying heavily on large-scale labeled multi-view data such as the depth or geometry annotations or posed image sets. This dependence is becoming a practical bottleneck: high-quality 3D supervision is expensive to obtain, and the available labeled datasets are being rapidly exhausted, which can slow further progress. Moreover, a non-trivial portion of the training signal in current pipelines is derived from SfM systems such as COLMAP, providing only approximate camera calibration and poses. These pseudo-labels can be noisy or biased, and imperfect geometry or camera supervision may even be detrimental by encouraging the model to fit inconsistent labels rather than learn robust geometric priors. And currently the general foundation models are trained almost on all the labeled dataset that is available.

These limitations motivate a shift toward weaker and unsupervised learning objectives, where the model can infer the missing labels such as cameras, correspondences, and latent 3D structure from raw image collections. Enabling models to self-discover reliable geometric supervision from data is therefore crucial for scaling beyond curated 3D datasets and for improving robustness in truly unconstrained settings.

4.2.1 Architecture

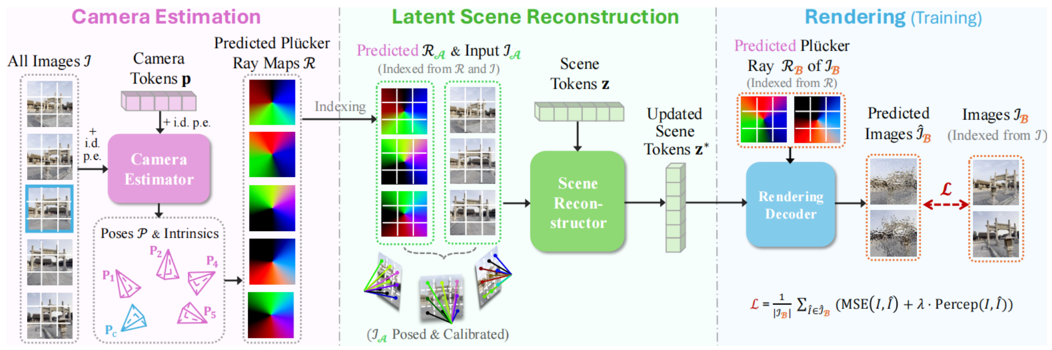
Camera Estimation
Latent Scene Reconstruction
Rendering (Training)
All Images $\mathcal{I}$
Camera Tokens $\mathbf{p}$
Predicted Plücker Ray Maps $\mathcal{R}$
+ i.d. p.e.
+ i.d. p.e.
Camera Estimator
Poses $\mathcal{P}$ & Intrinsics
P_1
P_2
P_4
P_5
P_c
Indexing
Predicted $\mathcal{R}_{\mathcal{A}}$ & Input $\mathcal{I}_{\mathcal{A}}$
(Indexed from $\mathcal{R}$ and $\mathcal{I}$)
($\mathcal{I}_{\mathcal{A}}$ Posed & Calibrated)
Scene Tokens $\mathbf{z}$
Scene Recon-structor
Updated Scene Tokens $\mathbf{z}^*$
Predicted Plücker Ray $\mathcal{R}_{\mathcal{B}}$ of $\mathcal{I}_{\mathcal{B}}$
(Indexed from $\mathcal{R}$)
Rendering Decoder
Predicted Images $\hat{\mathcal{I}}_{\mathcal{B}}$
Images $\mathcal{I}_{\mathcal{B}}$
(Indexed from $\mathcal{I}$)
$\mathcal{L}$
$\mathcal{L} = \frac{1}{|\mathcal{I}_{\mathcal{B}}|} \sum_{\hat{I} \in \hat{\mathcal{I}}_{\mathcal{B}}} (\mathrm{MSE}(I, \hat{I}) + \lambda \cdot \mathrm{Percep}(I, \hat{I}))$

Figure 14: RayZer training pipeline with camera estimation and ray-conditioned rendering. RayZer first predicts camera poses/intrinsics and per-image Plücker ray maps, reconstructs latent scene tokens from posed input views, and renders target views by decoding along predicted target rays. The model is trained end-to-end with image reconstruction losses between rendered and ground-truth target views.

As the RayZer's architecture shows in Fig. 14, given an unposed set of images from the same scene, RayZer first predicts camera parameters for each image, converts them into dense per-pixel Plücker ray maps, and then learns to render a target view conditioned on the target rays and a latent scene representation. Training uses only image reconstruction losses between the rendered view and the ground-truth target image, encouraging the model to discover a camera scene decomposition that is useful for synthesis.

RayZer follows a cascaded design:

- a camera estimaton predicts per-view extrinsics, which relative to a reference view, and a simplified intrinsics model
- the predicted cameras define Plücker ray maps that serve as pixel-aligned geometric conditioning
- a latent scene reconstructor aggregates multi-view information into a compact set of scene tokens
- a rendering decoder maps target-ray tokens and scene tokens to target RGB patches, which are then unpatchified to obtain the final image. Notably, the only explicit geometric prior injected into the network is the ray structure derived from the predicted cameras

4.2.2 Unsupervised Learning Details

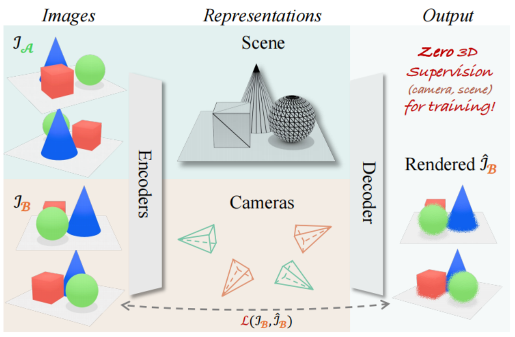
Images
Representations
Output
$\mathcal{I}_{\mathcal{A}}$
Scene
Zero 3D Supervision (camera, scene) for training!
Encoders
Decoder
Rendered $\hat{\mathcal{I}}_{\mathcal{B}}$
$\mathcal{I}_{\mathcal{B}}$
Cameras
$\mathcal{L}(\mathcal{I}_{\mathcal{B}}, \hat{\mathcal{I}}_{\mathcal{B}})$

Figure 15: RayZer high-level formulation: zero 3D supervision for novel view synthesis. Given a set of images, RayZer encodes them into a latent scene representation and jointly reasons about camera geometry, then decodes a target view without requiring explicit 3D labels (e.g., camera poses or scene geometry) during training.

RayZer removes the need for any 3D labels, that is no goundtruth poses, intrinsics, depth or geometry by training purely through view reconstruction. As shown in Fig. 15, given an unposed image set from the same scene, it splits the images into two subsets: $\mathcal{J}_A$ as the context and $\mathcal{J}_B$ as the targets. The encoder uses $\mathcal{J}_A$ to build a latent scene representation, while simultaneously predicting camera parameters for all views. These predicted cameras are converted into dense per-pixel Plücker ray maps, which serve as the only explicit geometric conditioning. A rendering decoder then takes the scene latent together with the target-ray embeddings of $\mathcal{J}_B$ and synthesizes $\hat{\mathcal{J}}_B$. Training minimizes a purely image-based loss

$$\mathcal{L}(\mathcal{J}_B, \hat{\mathcal{J}}_B) = \frac{1}{|\mathcal{J}_B|} \sum_{I \in \mathcal{J}_B} \Big(\mathrm{MSE}(I, \hat{I}) + \lambda \mathrm{Perceptual}(I, \hat{I}) \Big), \tag{26}$$

and gradients propagate through the entire pipeline, forcing the model to self-discover cameras and a 3D-aware scene latent that are jointly consistent for rendering. This directly addresses the scalability bottleneck of supervision-heavy pipelines and avoids being constrained by noisy SfM pseudo-labels.

4.2.3 Summary

RayZer provides a concrete path beyond pipelines without labeling: it shows that large view synthesis models can learn 3D-aware rendering directly from raw image collections by jointly learning cameras, rays, and a latent scene representation under self-supervision. This directly addresses the limitations of supervised general models and motivates further research on camera-free, label-free multi-view learning at scale.

Conceptually, RayZer can be viewed as a synthesis of the key ideas from the previous models: it adopts a self-supervised without labeling learning objective to remove dependence on 3D annotations, while reusing a VGGT-style token-based multi-view aggregator (i.e. add camera and register tokens with attention) to fuse information across views. At the same time, it inherits LVSM's Plücker-ray conditioning as a minimal yet effective way to inject camera geometry into a Transformer-based renderer. In this sense, RayZer combines "unsupervised learning + token-based multi-view reasoning + ray-based geometric conditioning" into a unified framework for scalable novel view synthesis.

4.3 Overall

LVSM and RayZer illustrate a complementary route to general 3D vision that prioritizes photorealistic novel view synthesis with minimal explicit 3D inductive bias. Instead of committing to a handcrafted 3D representation such as volumes or Gaussian primitives and an analytical renderer, both methods formulate rendering as a learned, token-based mapping conditioned on ray geometry, where Plücker-ray embeddings provide a lightweight but expressive way to encode camera information. LVSM shows that, given posed multi-view supervision, a large Transformer can directly learn high-quality view synthesis in a feed-forward manner. RayZer pushes this paradigm further by removing 3D labels altogether and jointly learning cameras, a latent scene representation, and a ray-conditioned renderer under self-supervision. Together, these works suggest that ray-conditioned Transformers can serve as scalable, reusable backbones for view synthesis, while highlighting an emerging trade-off between minimizing explicit geometric constraints and maintaining strong consistency and robustness under challenging viewpoint extrapolation.

5 Conclusion

Across the four papers reviewed, we can see a clear and consistent message that general 3D vision is less about inventing a new scene representation, and more about amortizing multi-view reasoning into large pre-trained Transformers, while carefully choosing what geometric structure to expose to the model. The resulting design space can be cleanly summarized along three axes: supervision: what labels we assume, representation: what intermediate variables we predict, and the degree of explicit 3D inductive bias: how much hard geometry we require.

NeRF and 3DGS show that ray-consistency with differentiable rendering is sufficient for high quality, but their per-scene optimization makes scalability the bottleneck. In contrast, DUSt3R and VGGT treat geometry as a feed-forward prediction problem: once trained, they run on new scenes without iterative solvers. The key conceptual shift is not just speed, it changes what the model learns. A model trained at scale can internalize a strong prior about geometry, so the network becomes a forward pass rather than a hand-designed pipeline.

DUSt3R's pointmap is a great example that the intermediate variables is important: it is dense enough to support multiple downstream tasks (depth, matching, pose), yet simple enough to be regressed directly. VGGT appears even more brutally simple: add camera and register tokens, then repeatedly mix information with attention, and the model learns to output cameras and dense geometry. This suggests that the bottleneck has shifted from designing geometry modules to designing the information flow, i.e. how tokens communicate across views. In particular, alternating between global attention (cross-view exchange) and frame attention (per-view consolidation) looks like a learned analogue of match-and-fuse, which plausibly explains why such a seemingly plain architecture can work so well on VGGT.

For novel view synthesis, geometry alone is not enough: wide-baseline extrapolation and real-world appearance require a strong prior beyond metric reconstruction. LVSM's key insight is that one can push photorealistic synthesis with minimal explicit 3D bias by conditioning a large Transformer on Plücker rays, i.e., providing a lightweight geometric interface without committing to a handcrafted 3D representation on LVSM. RayZer takes this further and, in my view, can be summarized as: self-supervised learning + VGGT-style token aggregation + LVSM-style Plücker-ray conditioning on RayZer. This combination is compelling because it keeps the geometry injection minimal, i.e. the rays, while letting the model learn the rest such as scene latent, rendering from data.

A practical constraint emerges clearly: DUSt3R, VGGT, LVSM largely rely on supervised training with posed multi-view data. This creates two limitations. First, truly high-quality 3D labels do not scale indefinitely, so progress can become data-limited. Second, when supervision comes from SfM pipelines such as using COLMAP's predicted poses, the labels may be noisy or biased; fitting them too faithfully can mislead training and reduce robustness. RayZer directly attacks this problem by learning cameras and scene representations from image reconstruction alone, which I consider an essential direction if we want 3D foundation models to scale beyond curated 3D datasets on RayZer.

Minimizing explicit 3D inductive bias improves flexibility, but it also weakens guarantees. A recurring trade-off is that strict projective ormetric consistency versus photorealism and generalization. Geometry-regression models may output shapes that are usable but not perfectly consistent under a single camera model; synthesis-first models may render convincingly but drift in geometry, especially under extreme viewpoint extrapolation or limited context.

I see some concrete directions that follow naturally from the surveyed works: a better self-supervision signals matters: move beyond pure pixel or perceptual losses by incorporating multi-view cycle constraints, correspondence consistency, or uncertainty-aware training so the model can reject noisy pseudo-labels. The network could unify backbone for reconstruction and synthesis: use a shared multi-view aggregator that outputs both a geometry-centric state for controllability and an appearance-centric state for photorealism, with rays as the common interface. And self-supervised learning can be used to improve robustness: treat camera estimation not as a preprocessing step but as a learned, uncertainty-aware latent variable, so that models remain stable when real-world metadata is missing or inaccurate.

Overall, DUSt3R and VGGT show that generalizable geometric reconstruction is feasible with feed-forward Transformers trained at scale, while LVSM and RayZer demonstrate that photorealistic synthesis can be achieved with minimal explicit 3D inductive bias by conditioning on rays. Taken together, these works suggest a promising recipe for 3D foundation models: token-based multi-view aggregation, ray-based geometric interfaces, and increasingly self-supervised learning to break the dependence on fragile or expensive 3D labels on all these introduced works. The following table is a summary of the four representative methods:

Method	Task	**Supervision / camera info**	**Representation**
DUSt3R [1]	Recon.	3D regression; Paired image relative camera labels	Pointmap (pixel-aligned 3D) + confidence
VGGT [2]	Recon.	Multi-task supervised (cameras + dense geometry + tracks)	Multi-view tokens + dense heads (depth/pointmap)
LVSM [3]	NVS	Posed multi-view (pose-conditioned training/inference)	Plücker-ray-conditioned Transformer
RayZer [4]	NVS	Self-supervised; cameras are self-estimated (no 3D labels)	Token aggregation + Plücker rays + latent scene tokens

Table 1: Compact comparison along supervision, representation.

References

[1] 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.

[2] 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.

[3] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

[4] Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, et al. Rayzer: A self-supervised large view synthesis model. *arXiv preprint arXiv:2505.00702*, 2025.

[5] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[7] Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. *Advances in Neural Information Processing Systems*, 35:3502–3516, 2022.

[8] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] 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.