Deep Learning for Computer Vision

HW4 Report

b12901069 段奕鳴

Problem 1: Can Generative Video Models Help Pose Estimation?

Q1.1 Explain the DUSt3R and VGGT models and analyze their key differences.

DUSt3R (Dense and Unconstrained Stereo 3D Reconstruction)

DUSt3R pioneers a paradigm shift in 3D computer vision, moving from traditional iterative optimization pipelines like Structure-from-Motion (SfM) and Multi-View Stereo (MVS) to end-to-end, feed-forward deep learning models for joint camera pose and dense geometry estimation from unconstrained image collections.

Core Task: DUSt3R is initially designed to solve the pairwise (stereo/two-view) 3D reconstruction problem without needing prior information about camera calibration or viewpoint poses ("unconstrained"). It can also handle a single input image for monocular reconstruction.

Architecture: DUSt3R uses a transformer encoder and decoder architecture, often leveraging a pre-trained Vision Transformer (ViT) encoder in a Siamese manner with shared weights.

Output Representation: The core output is a set of corresponding pointmaps (dense 2D $\leftrightarrow$ 3D mappings), along with associated confidence maps. The pointmaps are expressed in a common coordinate frame (the coordinate frame of the first image).

Multi-View Scaling: For more than two images, DUSt3R uses a separate global alignment post-processing step to align the pairwise predicted pointmaps into a common reference frame. This step typically requires iterative post-optimization.

Downstream Tasks: From the pointmaps, it can seamlessly recover various geometric quantities like depth estimation (the z-coordinate of the pointmap), pixel correspondences (point matching via Nearest Neighbor search in 3D space), and camera parameters (intrinsics and relative pose via optimization/Procrustes alignment).

VGGT (Visual Geometry Grounded Transformer)

VGGT is a subsequent, large-scale model that builds upon the feed-forward concept, aiming to address the multi-view limitation of DUSt3R.

Core Task: VGGT is designed to perform 3D reconstruction for a general number of views in a single forward pass.

Architecture: A large transformer (~1.2 B parameters) with minimal 3D inductive biases. Its key architectural feature is alternating-attention layers, which switch between frame-wise self-attention (within a single image's tokens) and global self-attention (across all frames' tokens). It uses a DINO backbone for initial tokenization.

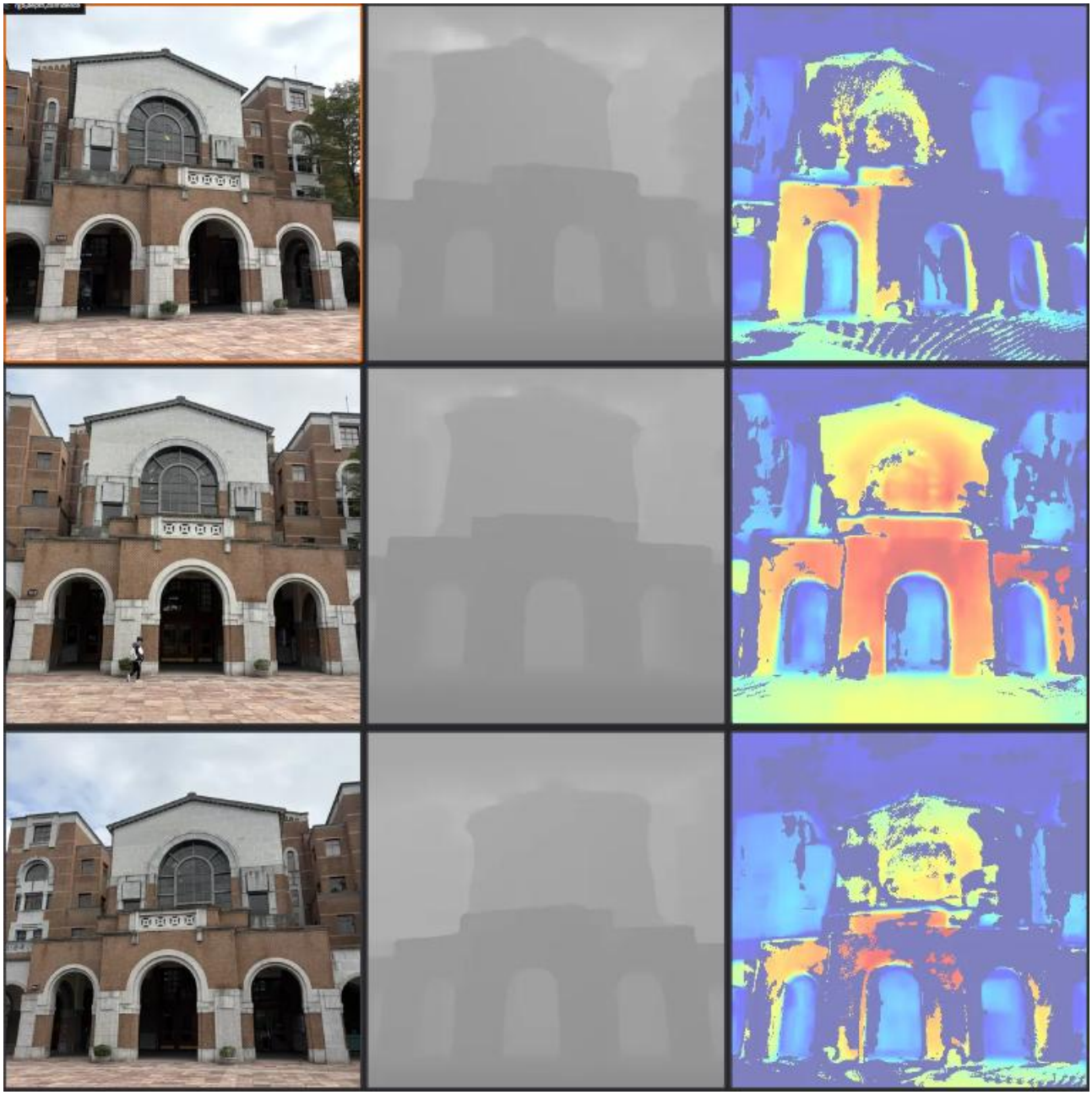
Output Representation: In a single forward pass, VGGT directly predicts a comprehensive set of 3D attributes for all N input frames, including camara parameters, depth maps, point maps in the coordinate system of the first camera, 3D point tracks/dense features.

Multi-View Scaling: VGGT performs true end-to-end multi-view prediction in a single forward pass, effectively eliminating the need for DUSt3R's costly global alignment post-processing for multi-view scenarios.

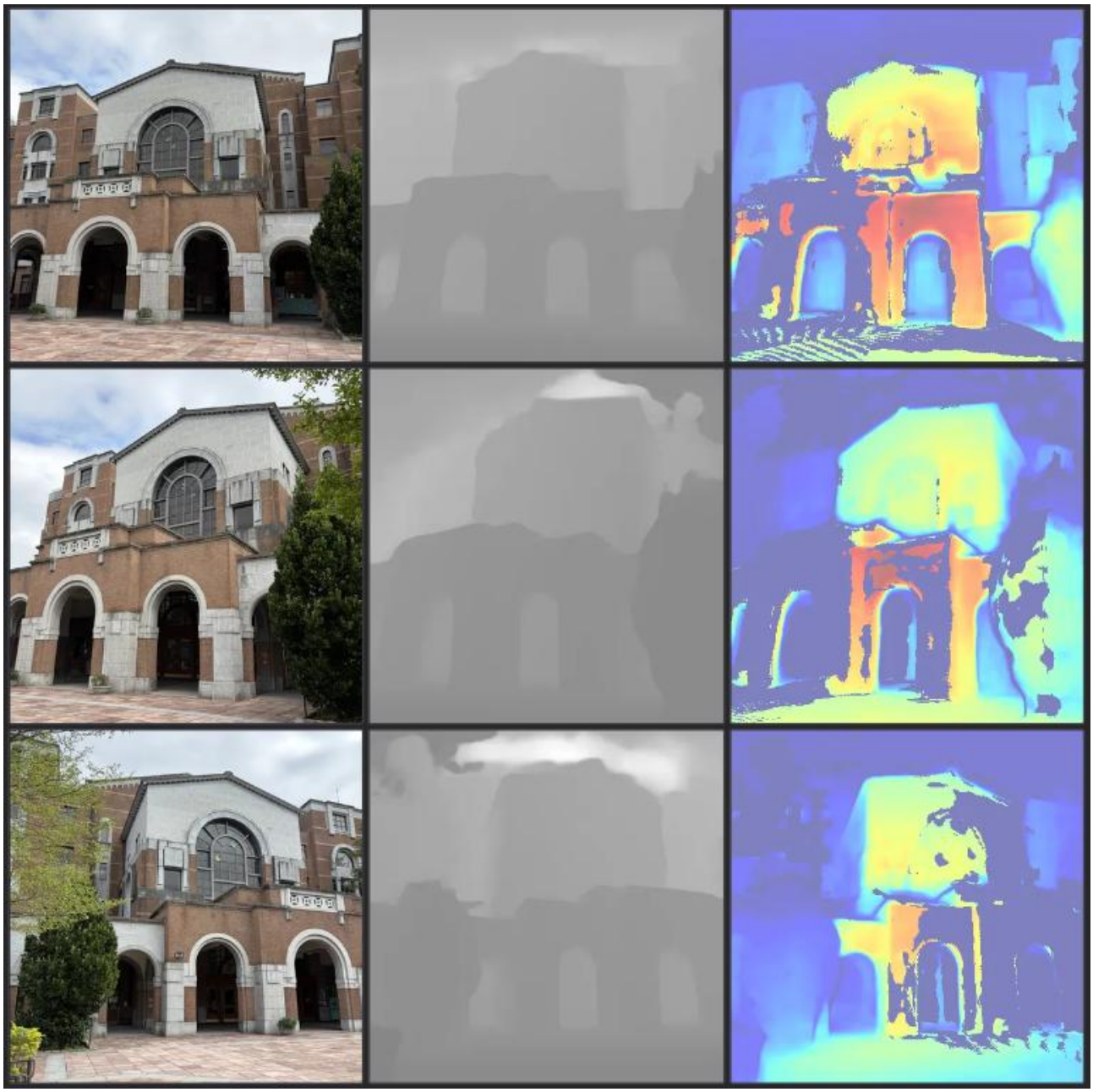
Comparison

	DUSt3R	VGGT
Primary Scope	Pairwise (2 views)	Multi-view (1 to hundreds images)
Multi-View Process	Pairwise Inference + Post-processing	Single Forward Pass (eliminates costly global alignment step)
Core Architecture	Siamese Transformer using Cross-Attention for two-view information sharing	Large Transformer using Alternating-Attention layers
Key Output	Pointmaps (dense 2D $\leftrightarrow$ 3D mappings)	Jointly predicts Camera, Depth, Pointmaps, and Tracks
Speed	Slow post-optimization required for multi-view	Very fast feed-forward prediction

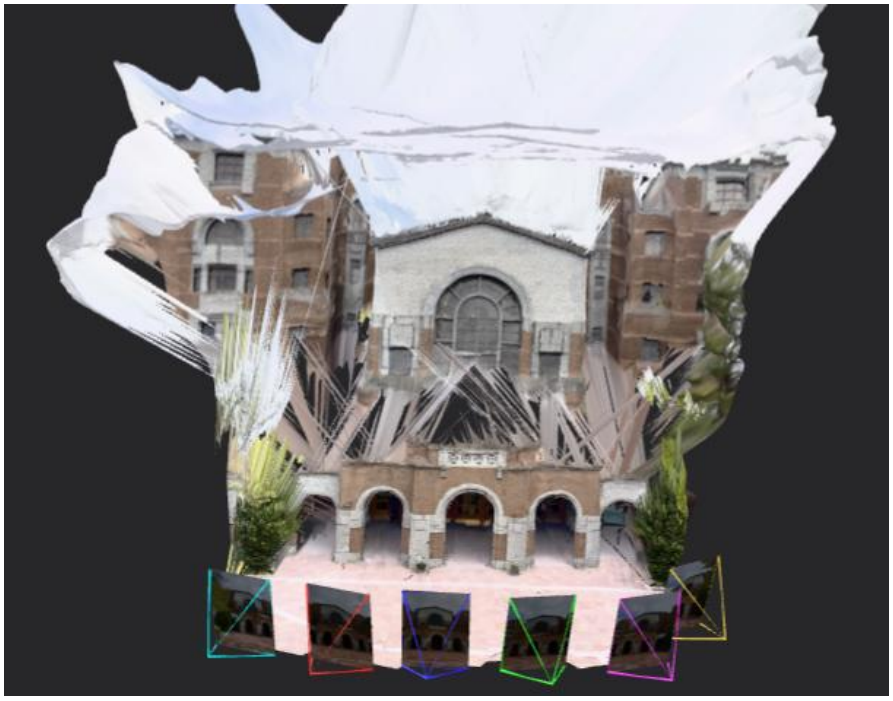
Q1.2 Run a demo of DUSt3R using your own image sequence and present the results.



rgb, depth, confidence for image 1-3



rgb, depth, confidence for image 4-6



3d model

Settings for this demo run:

schedule: linear

num_iterations: 5000

scenegraph: complete (all possible image pairs)

min_conf_thr: 1

Q2.1 Quantitative Evaluation on Public Dataset

Original wide-baseline pairs

```
================================================================================
Evaluation Summary (Mode: R)
------------------------------------------------------------
Total samples processed: 40

Rotation Errors - 40 valid samples:
  Mean (MRE):   86.864°
    Acc < 5°:   0.00%
    Acc < 15°:   20.00%
    Acc < 30°:   25.00%

AUC (R) @ Threshold:
  AUC @ 30°:    0.1558
  AUC @ 15°:    0.0767
  AUC @ 5°:    0.0000
  AUC @ 3°:    0.0000
================================================================================
```

Interpolated sequences

```
================================================================================
Evaluation Summary (Mode: R)
------------------------------------------------------------
Total samples processed: 40

Rotation Errors - 40 valid samples:
  Mean (MRE):   75.324°
    Acc < 5°:   2.50%
    Acc < 15°:   20.00%
    Acc < 30°:   45.00%

AUC (R) @ Threshold:
  AUC @ 30°:    0.2225
  AUC @ 15°:    0.0933
  AUC @ 5°:    0.0100
  AUC @ 3°:    0.0000
================================================================================
```

Q2.2 Case Study: idx 3781 PnP Solver Failure

Add try…except handling in dust3r/dust3r/cloud_opt/init_im_poses.py to catch the cv2.error

```
274         try:
275             success, R, T, inliers = cv2.solvePnPRansac(pts3d[msk], pixels[msk], K, None, iterationsCount=niter_PnP, reprojectionError=5, flags=cv2.SOLVEPNP_SQPNP)
276 
277         except cv2.error as e:
278             print(f"WARN: PnP failed for focal {focal} due to degenerate points: {e}")
279             success = False
```

The idx 3781 of public dataset will cause the PnP solver to crash.

```
WARN: PnP failed for focal 1014.2542114257812 due to degenerate points: OpenCV(4.12.0) /io/opencv/modules/calib3d/src/sqpnp.cpp:
238: error: (-215:Assertion failed) point_coordinate_variance >= POINT_VARIANCE_THRESHOLD in function 'computeOmega'
```

Present the MRE for both the original pair and the interpolated sequence relative to the ground truth.

	Original pair	Interpolated sequence
MRE	82.886°	33.459°

Use visual evidence (both the DUSt3R-generated point cloud and the visual effect of the interpolated sequence itself) to analyze why the PnP solver failed for this sequence.

The PnP (Perspective-n-Point) solver fails because the 3D points generated from the interpolated sequence are co-linear.

PnP algorithms calculate camera poses by matching 2D image points to 3D world points. To find a stable solution, the 3D points must have sufficient spread in 3D space structure.

Mathematically, if all the 3D points lie on a single 2D plane (co-planer) or a single 1D line (co-linear), the system of equations used to solve for the pose becomes degenerate. These are either infinite solutions or the solver becomes numerically unstable.

The interpolated sequences lack consistent 3D parallax. Instead of simulating a true 3D camera movement around the object, the video model seems to be simply morphed the pixels in a linear way, like a 2D zoom and pan.

Q3.1 Compare the overall performance difference from your two runs

	Original Pairs	Interpolated Sequences
MRE	86.864°	75.324°
Acc $< 5°$	0.00 %	2.50 %
Acc $< 15°$	20.00 %	20.00 %
Acc $< 30°$	25.00 %	45.00 %
AUC @ 30°	0.1558	0.2225
AUC @ 15°	0.0767	0.0933
AUC @ 5°	0.0000	0.0100
AUC @ 3°	0.0000	0.0000

The quantitative results show a significant improvement in rotation error and accuracy when using the interpolated sequences. For the interpolated sequences (video input), the MRE has 8.69° reduction.

Q3.2 Analyze the quantitative results and explain your high-level hypotheses (i.e. Why, in general, does this performance gap exist?)

The overall performance gap exists because the video generative model effectively transforms a **geometrically ill-posed problem** into a **more robust, solvable short-baseline problem.**

The high error observed for the original pairs is primarily due to the severe difficulty of establishing **geometric correspondence across large camera motion**. The large change in viewpoint, such as the 80° to 90° yaw difference in this dataset, introduces significant scale, perspective distortion, and illumination changes between the two images. This visual disparity makes it extremely difficult for DUSt3R's feature extraction and matching modules to establish reliable 2D correspondences, consequently leading to a poor initial pose estimate and failure during global alignment.

The successful mechanism of interpolation lies in the video generative model's ability to **leverage powerful spatial and temporal priors** to hallucinate a geometrically plausible visual path between the two end-point images. This conversion breaks the challenging

wide-baseline jump into a sequence of many small, manageable **short-baseline steps**. This technique simplifies the task, enabling **robust correspondence** between temporally adjacent frames. Additionally, the sequence provides **dense contextual information** about the 3D geometry and camera trajectory, which stabilizes the global alignment process and allows the model to converge to a much more accurate pose estimate, resulting in the observed performance gain.

In summary, the high-level hypothesis is that the video generative model acts as a powerful **feature stabilizer**, ensuring a continuous, high-quality correspondence path where standard feature matching would otherwise fail catastrophically due to large baseline disparity. This change in the problem domain is the core reason for the substantial performance improvement.

Problem 2: Sparse-View 3D Gaussian Splatting

Q1.

(1) Try to explain the difficulty of sparse-view 3DGS in your own words.

The difficulty of sparse-view 3D Gaussian Splatting (3DGS) are primarily from the lack of sufficient geometric constraints. 3DGS, like NeRF, relies on optimizing 3D primitives (the Gaussians) to accurately render novel views. In a sparse-view scenario, the input images are few and often widely separated. This means there are large unobserved gaps between the input views. The optimization process struggles to correctly infer the precise 3D location, size, and color properties of Gaussians that lie in the unobserved regions, leading to floaters (artifact Gaussians in the air), blurry reconstructions, or poor fidelity when synthesizing new views far from the training data. The model suffers from ambiguity because it has too little information to distinguish correct geometry from plausible artifacts.

(2) Compare 3D Gaussian Splatting with NeRF (pros & cons)

	3D Gaussian Splatting	NeRF
Representation	Explicit scene representation using thousands/millions of individual 3D Gaussians	Implicit scene representation encoded within the weights of a multilayer perceptron (MLP)
Rendering Speed	Fast because a highly parallelized rasterization pipeline is used, real-time rendering	Slow because it requires ray-marching hundreds of points per pixel, difficult for real-time use
Training Speed	Significant faster training due to the differentiable rasterization process	Slow due to the extensive ray-marching required
Memory	High memory consumption as it stores millions of explicit Gaussians and their properties	Lower memory consumption for the model itself because it only stores network weights
Geometric Fidelity	May sometimes produce visual artifacts or requires complex regularization like densification and pruning	Very fine detail and smooth surfaces can be achieved, often resulting in high-quality depth maps

(3) Which part of 3D Gaussian Splatting is the most important you think? Why?

The most important part of 3D Gaussian Splatting is the differentiable Gaussian rasterization pipeline. This allows 3DGS to render novel views at real-time frame rates, bypassing the slow ray-marching process used by NeRF.

The rasterization process is designed to be fully differentiable. This is crucial because it allows the gradients to efficiently flow back to the parameters of every single Gaussian (3D position, size, orientation, and color). This enables the rapid, data-driven optimization of the Gaussian properties, which is the key to the fast training speed.

Q2.

Give novel view camera pose, your 3D gaussians should be able to render novel view images. Please evaluate your generated images and ground truth images with the following three metrics (mentioned in the InstantSplat paper). Try to use at least three different hyperparameters settings and discuss/analyze the results.

	Settings	PSNR ↑	SSIM ↑	LPIPS ↓
1	1000 iters, position lr=1.6e-4, disable densification	15.75	0.5481	0.3780
2	1000 iters, position lr=3.2e-4, disable densification	17.77	0.5742	0.3801
3	2000 iters, position lr=1.6e-4, disable densification	15.78	0.5445	0.3870
4	1000 iters, position lr=1.6e-4, densification interval=100	15.58	0.5519	0.3791
5	1000 iters, position lr=1.6e-4, densification interval=50	15.53	0.5511	0.3781

Discussions:

The most effective modification was doubling the position learning rate in Setting 2, which yielded the highest improvements in both PSNR (+2.02 dB) and SSIM (+0.026). This suggests that while the initialization provides a strong geometric foundation, the points are not perfectly aligned with the ground truth scale and poses initially. The standard learning rate appears too conservative to correct these misalignments within a short training window, whereas the higher rate allowed the Gaussians to migrate more aggressively to their correct positions. However, the perceptual metric (LPIPS) degraded slightly, implying that this aggressive movement might have introduced some high-frequency noise that is penalized by deep features despite the better pixel-level alignment.

Conversely, simply extending the training duration to 2000 iterations in Setting 3 offered negligible gains in PSNR and SSIM while worsening the LPIPS score. This likely indicates that the model has begun to overfit the sparse training data. With only few views available, an extended optimization process allows the model to "memorize" the training images by creating artifacts or "floaters" in empty space. These artifacts minimize the reconstruction error from the known angles but look incorrect or unnatural from novel test angles, leading to the observed drop in perceptual quality.

Finally, attempts to enable densification in Settings 4 and 5 resulted in lower performance compared to the baseline, possibly indicating the risks of modifying geometry in sparse settings. Without sufficient multi-view constraints, the densification process tends to add points in incorrect locations, such as void spaces, rather than enhancing actual detail. The results suggest that for this specific task, trusting the fixed topology of the high-quality initialization might be a better strategy than learning new geometry from scratch.

Visualization of image 129896000

Setting 1

Setting 2

Setting 3

Setting 4

Setting 5