# Matrix3D: Large Photogrammetry Model All-in-One

Supplementary Material

Here, we present an additional description of the model architecture(Sec. 7), dataset preprocessing(Sec. 8), training details(Sec. 9), and experiments(Sec. 10).

## 7. Model Architecture

For RGB data, we use DINOv2 [70] and Stable Diffusion [82] VAE to extract deep features from pixels before sending them into the modality-specific encoders. The modality-specific encoders are composed of stacked convolution and linear layers following [71] to patchify image-like 2D data into 1D tokens. The patchify scale for RGB, pose, and depth are set to 2, 1, and 4. After the tokens are processed by the transformer, we use similar modality-specific linear layers [71] to unpatchify each modality token back to the original shape according to the corresponding patchify scales. The whole multi-view transformer encoder includes 20 self-attention blocks with a hidden size of 1024, while the decoder includes 40 stacked self-attention and cross-attention blocks with a hidden size of 1408 following HunyuanDiT [51].

For classifier-free guidance (cfg), we empirically found the following settings to perform best: 1.5 for RGB / poses, and 1.0 for depth (w/o cfg).

## 8. Dataset Pre-processing

As illustrated in the main paper, we train Matrix3D on a mixture of six datasets, including Objaverse [19], MVImgNet [132], CO3D-v2 [79], RealEstate10k [141], Hypersim [81], and ARKitScenes [4]. In each training batch, the datasets have a proportion of 4:4:4:4:4:1. Table 8 provides a summary of these datasets used for training, including the size (in terms of scenes and images), type (real or synthetic), scene categories, and supported modalities (RGB, camera poses, and depths). For all datasets, we apply scene normalization and camera normalization. Camera poses are represented as Plücker rays. Note that the depth images provided in each dataset are not always complete. Specifically, CO3D-V2 and ARKitScenes provide incomplete depth images, while for the Objaverse dataset we only have the rendered object foreground depth.

**Normalization.** Due to the highly diverse distributions of existing datasets, including variations in scale and scene type, preprocessing them consistently poses a challenge. To address this, we apply the following normalization.

- Scene Normalization: To normalize the whole scene scale, we adapt our approach depending on the dataset type and available modalities. For object-centric datasets with camera poses provided (i.e., Objaverse [19], MVImgNet [132], and CO3D-v2 [79]), we follow RayDiffusion [137] by setting the intersection point of the input camera rays as the origin of the world coordinates and defining the scene scale accordingly. For scene-type datasets that provide depth information (i.e., Hypersim [81] and ARKitScenes [4]), we use the depth of the first view as a reference, calculating its median value and normalizing it to 1.0. For those datasets without depth data (i.e., RealEstate10k [141]), we determine the scale based on the camera distances to the average positions and set the maximum distance to 1.0.
- Camera Normalization: We perform camera normalization after scene normalization. Specifically, we set the first view's camera as the identity camera with rotation $R = I$ and translation $T = [0, 0, 1]$, while preserving relative transformations between cameras across views.

**Objaverse Rendering.** For the Objaverse dataset, we render all models into RGB and depth images for training. Specifically, each 3D object is first normalized at the world center within a bounding box of $[-1, 1]^3$, and we render the whole scene from 32 random viewpoints. The render camera FoV is set to 50°. The azimuth and elevation angles are randomly sampled in $[0°, 360°]$ and $[-45°, 90°]$. The camera distance to the world center is randomly sampled in [1.1, 1.6], and the height on the z-axis is set in [-0.4, 1.2]. We use a composition of random lighting from area lighting, sun lighting, point lighting, and spot lighting.

## 9. Training Details

Table 7 reports the detailed training hyper-parameter settings of three stages. We didn't apply any data augmentation techniques and center-cropped the input images into a square.

## 10. Experiments

### 10.1. DTU Dataset Split for Depth Evaluation

In Sec. 4.3, we use different evaluation set for monodepth and multi-view depth evaluation. Specifically, we use the IDR [129] subset for monodepth because perfect foreground masks are provided, and follow previous work [111] to use the MVSNet [127] subset for multi-view depth evaluation.

### 10.2. Point Cloud Fusion

In Sec. 4.3, we back-project multi-view depth maps to point cloud. In practice, we additionally conduct geometric consistency filtering and fusion to clean the point cloud. The

| Hyper-parameters | Ablation | Stage 1 | Stage 2 | Stage 3 |
|---|---|---|---|---|
| Optimizer | AdamW [62] | AdamW [62] | AdamW [62] | AdamW [62] |
| Learning rate | 1e-4 | 1e-4 | 1e-5 | 1e-5 |
| Learning rate scheduler | Constant | Constant | Constant | Constant |
| Weight decay | 0.05 | 0.05 | 0.05 | 0.05 |
| Adam $\beta$ | (0.9, 0.95) | (0.9, 0.95) | (0.9, 0.95) | (0.9, 0.95) |
| Max view num | 4 | 4 | 8 | 8 |
| Batch size | 512 | 1024 | 1024 | 256 |
| Steps | 100k | 200k | 30k | 30k |
| Warmup steps | 4k | 4k | 1k | 1k |
| Initialization | HunyuanDiT[51] | HunyuanDiT[51] | Stage 1 | Stage 2 |
| Attention blocks (encoder) | 20 | 20 | 20 | 20 |
| Attention blocks (decoder) | 40 | 40 | 40 | 40 |
| Image resolutions | $256\times256$ | $256\times256$ | $256\times256$ | $512\times512$ |
| Raymap resolutions | $16\times16$ | $16\times16$ | $16\times16$ | $32\times32$ |
| Depth resolutions | $64\times64$ | $64\times64$ | $64\times64$ | $128\times128$ |
| Datasets | Object-centric | Object-centric | All | All |

Table 7. Detailed hyper-parameters.

| Dataset | Size | | Type | | Support Modalities | | |
|---|---|---|---|---|---|---|---|
| | Scenes | Images | Real/Synthetic | Scene Type | RGB | Poses | Depths |
| Objaverse [19] | 800K | 25M | Synthetic | Object-centric | ✓ | ✓ | Foreground only |
| MVImageNet [132] | 220K | 6.5M | Real | Object-centric | ✓ | ✓ | × |
| CO3Dv2 [79] | 19K | 1.5M | Real | Object-centric | ✓ | ✓ | Incomplete |
| RealEstate10K [141] | 10K | 10M | Real | Indoor/Outdoor Scene | ✓ | ✓ | × |
| Hypersim [81] | 461 | 77K | Synthetic | Indoor Scene | ✓ | ✓ | Complete |
| ARKitScenes [4] | 5K | 450K | Real | Indoor Scene | ✓ | × | Incomplete |

Table 8. Dataset details.

| Methods | GT Pose | GT Range | GT Int. | Align | DTU | | ETH3D | | T&T | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | rel ↓ | $\tau$ ↑ | rel ↓ | $\tau$ ↑ | rel ↓ | $\tau$ ↑ |
| DeepV2D | × | × | ✓ | med | 7.7 | 33.0 | 11.8 | 29.3 | 8.9 | 46.4 |
| DUSt3R | × | × | × | med | 2.76 | 77.32 | **4.71** | **61.74** | **5.54** | **56.38** |
| Ours | × | × | ✓ | med | **1.85** | **85.46** | 7.83 | 38.80 | 6.16 | 49.43 |

Table 9. Unposed MVD evaluation on DTU, ETH3D, and T&T.

filtering follows [127, 134], consisting of a combination of the following operations.

- Geometric filtering. We project the pixels from the reference view to source views, find the pixel at the projection location, and project it back to reference view. Then we check the difference of the original position and the re-projected position, as well as their depths.
- Geometry fusion. We project all pixels from source views to the reference views, and each pixel in the reference view may receive multiple values. We then change the original depth result to the average or the median of all the gathered values.

## 10.3. Ablation Study on Multi-task Training

In this section we compare the model trained by masked learning and the task-specific models including NVS, pose estimation and depth estimation. The latter ones have the same network architecture as the stage 1 model, but the input/output configuration of the training samples is set to only one task. All 4 models are trained from HunyuanDiT [51] initialization with halved batch size and total steps due to limited time and compute resources. The evaluation metrics for each task is the same as the main paper.

Quantitative reuslts are shown in Table 10. The model with masked learning strategy (*Multi-task*) surpasses the task-specific model in the NVS task, but fails for pose estimation and depth estimation. One possible reason is that the model capacity is shared by different tasks. Another reason related to practice is that the models for ablation studies do not fully converge. According to the evaluation curve with respect to training steps in Figure 8, the model with halved batch size at 100k steps has similar performance as the full model at 60k-70k steps which still has large room of improvement. Given that our model is initialized by an RGB diffusion model, the functionality of outputing ray maps and depth maps may need longer time to converge, and thus task-specific models achieves better results within limited training time. Although not a fair comparison, note that all models for ablation study are weaker than the stage 1 and the stage 3 model. Also, in Sec. 4.5 we show that the model trained by masked learning can support flexible input and boost the performance by utilizing additional input.

## 10.4. 3D Reconstruction

**Camera trajectory generation.** We build different camera trajectories for generating novel views depending on different reconstruction tasks. For monocular image input, we create an orbital trajectory and sample 80 cameras evenly. All cameras are set as look-at to the world center. For sparse-view image input, we fitted a spline trajectory from the input poses, and scaled up the trajectories two times, resulting in $240 (= 80 \times 3)$ views.

**3DGS optimization.** The proposed 3DGS optimization system in built upon the open-source pipeline [97] with several modifications. For each optimization step, we optimize Gaussian points on mini-batch images instead of single images. Besides of original L1 loss and SSIM loss, we adopt additional losses to improve the reconstruction robustness, including LPIPS loss $L_{\text{LPIPS}}$ [139], mask loss $L_{\text{mask}}$, accumulation regularization $L_{\text{accum}}$, absolute depth loss $L_{\text{depth}}$, and relative depth ranking loss $L_{\text{rel-depth}}$ [106]. The accumulation regularization is designed to constrain the alpha values of Gaussian points to be either fully opaque or completely transparent, aiming to reduce floaters in the scene. It is composed of a binary cross-entropy loss and entropy loss:

$$L_{\text{accum}} = BCE(\alpha, 0.5) - \alpha \log(\alpha) + (1 - \alpha) \log(1 - \alpha),$$

where $\alpha$ denotes the accumulation values.

For monocular 3D reconstruction, the value for each loss is set to $w_{\text{L1}} = 1.0, w_{\text{SSIM}} = 0.2, w_{\text{LPIPS}} = 10.0, w_{\text{mask}} = 5.0, w_{\text{accum}} = 5.0$. Depth loss is not applied in the optimization. The mini-batch size for each step is set to 10. For the input view, the weight of L1 loss is specifically set to 10.0 for high significance.

For sparse-view 3D reconstruction, the weight values are set to $w_{\text{L1}} = 1.0, w_{\text{SSIM}} = 0.2, w_{\text{LPIPS}} = 10.0, w_{\text{mask}} = 5.0, w_{\text{accum}} = 0.5, w_{\text{depth}} = 10.0, w_{\text{rel-depth}} = 20.0$. The mini-batch size for each step is set to $5$, and the L1 loss weight of input views is set to 20.

We use the back-projected point cloud as Gaussian point initialization. Similar to CAT3D [29], we conduct in total of 1200 and 3000 optimization steps for two tasks, respectively. We apply the scale regularization [118] to constrain the extreme Gaussian scales.

## 10.5. Limitation

During experiments, we found that our model performs well on object-centric and indoor scenes but degrades in outdoor environments, primarily due to the lack of large-scale outdoor training data—our dataset consists of objects and limited indoor scenes. The lack of high-quality outdoor data is a common issue in the community, and similar problem has been noticed in other models.

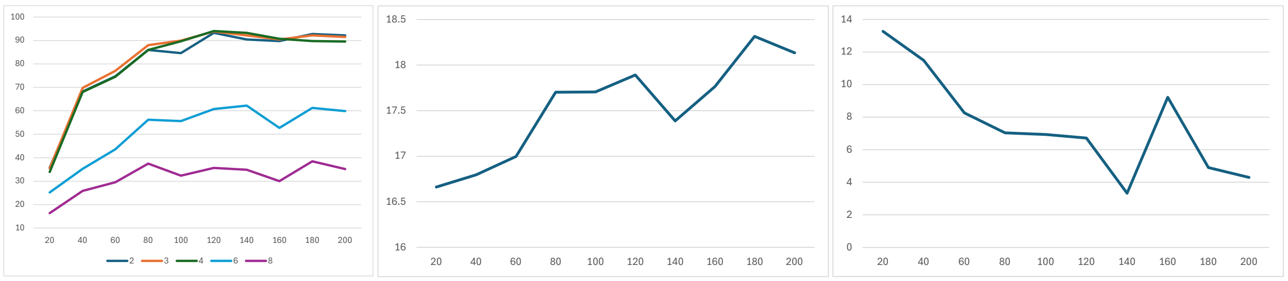
Tab. 9 demonstrates unposed depth prediction results on ETH3D and T&T. Our model performs worse than DUSt3R (trained on outdoor datasets), but still surpass DeepV2D.

## 10.6. More visualization

Here we present more visualization results about unposed sparse-view 3D reconstruction (Fig. 9) and multi-view depth predictions (Fig. 10).

| Methods | RRA @ 15° ↑ | | | CA @ 0.1↑ | | | Methods | PSNR↑ | SSIM↑ | LPIPS↓ | Methods | rel↓ | $\tau$ ↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 2 | 3 | 4 | | | | | | | |
| Pose only | **89.2** | **86.7** | **85.8** | 100.0 | **83.1** | **77.0** | NVS only | 16.30 | 0.77 | 0.30 | Depth only | **9.07** | **26.21** |
| Multi-task | 81.1 | 77.8 | 75.3 | 100.0 | 75.8 | 64.5 | Multi-task | **17.21** | **0.79** | **0.25** | Multi-task | 10.76 | 18.52 |
| Stage 1 | 92.2 | 91.5 | 89.6 | 100.0 | 87.8 | 80.8 | Stage 1 | 18.13 | 0.81 | 0.19 | Stage 1 | 4.30 | 49.81 |
| Stage 3 | 95.6 | 96.0 | 96.3 | 100.0 | 93.5 | 91.7 | Stage 3 | 18.87 | 0.85 | 0.21 | Stage 3 | 1.83 | 85.45 |

Table 10. Ablation study on multi-task training and task-specific training. Besides different training target, the ablation models have halved batch size and total steps. The multi-task model achieves better results in NVS task but fails for pose estimation and depth estimation. One possible reason is that the multi-task model converges slower than the task-specific models. Please refer to Sec. 10.3 for more analysis.



a) Pose (RRA@15° ↑) b) NVS (PSNR↑) c) Depth (rel↓)

Figure 8. Evaluation results of stage 1 model for a) pose estimation, b) NVS and c) Depth estimation with respect to training step. For pose estimation we report the results for multiple view numbers. Note that the stage 1 model is only trained with view number $\leq 4$.

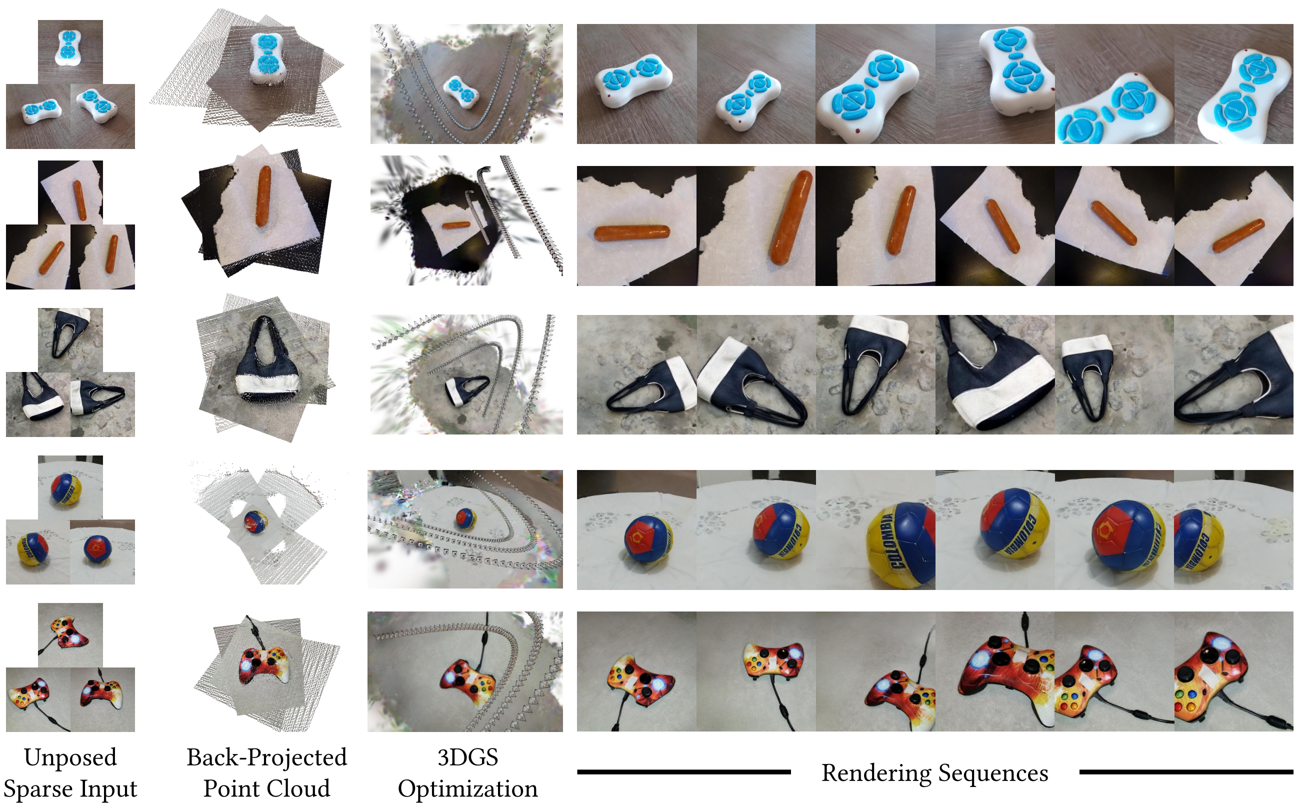


Figure 9. More unposed sparse-view 3D reconstruction results.

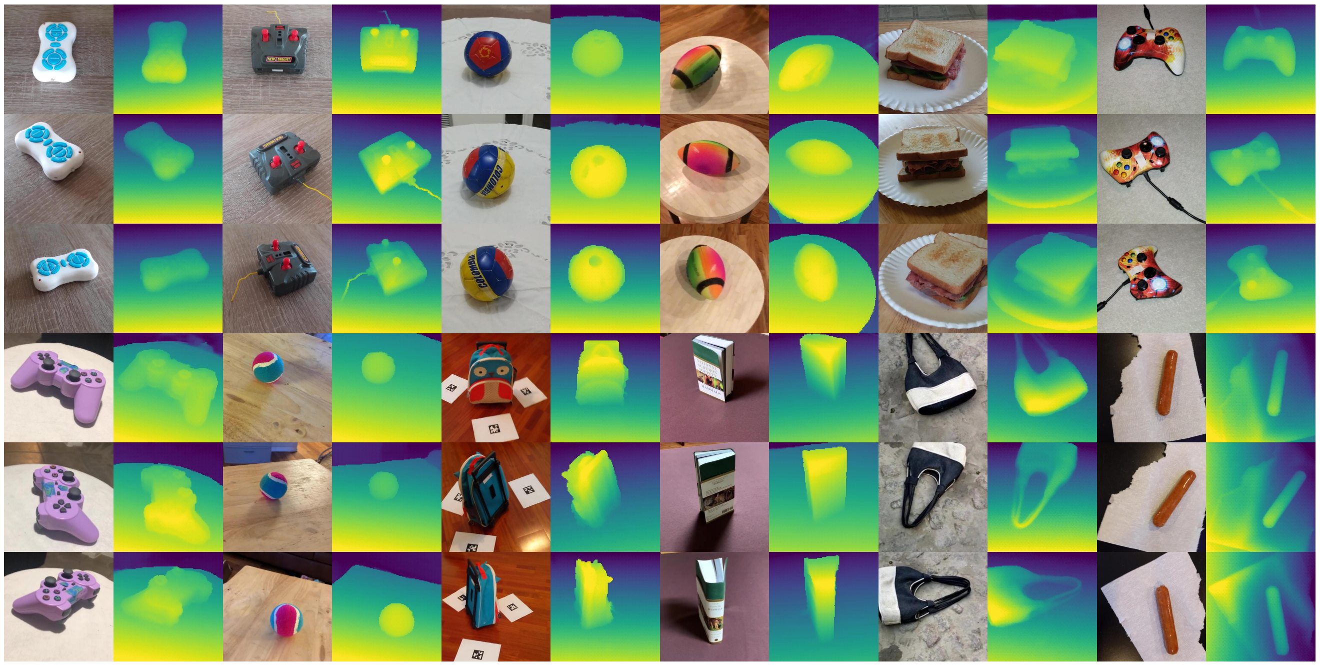


Figure 10. Visualization of multi-view depth prediction results.