SIMPLEPROC: FULLY PROCEDURAL SYNTHETIC DATA FROM SIMPLE RULES FOR MULTI-VIEW STEREO

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating procedural synthetic data for multi-view stereo (MVS) usually requires writing complex rules to match the realism of curated datasets. We demonstrate that we can generate effective training data using SimpleProc: a new, fully procedural generator driven by a very small set of rules based on Non-Uniform Rational Basis Splines (NURBS), as well as simple displacement and texture patterns. At a modest scale of 8,000 images, our approach achieves superior results compared to manually curated images at the same scale sourced from games and real-world objects. When scaled to 352,000 images, our approach achieves similar, and in some benchmarks, even better results than the current state-of-the-art trained on over 692,000 manually curated images.

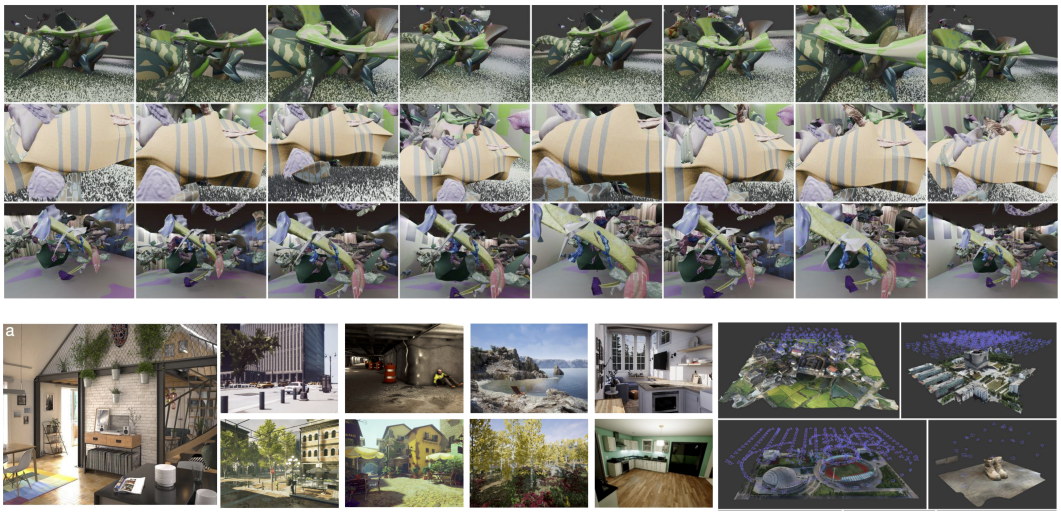

Figure 1: Fully procedural synthetic data from simple rules (top) is as effective as curated data from artists or 3D scans (bottom) for training multi-view stereo models.

1 INTRODUCTION

Synthetic data from computer graphics is widely used in 3D vision. Optical flow models such as WAFT Wang & Deng (2025) and FlowFormer++ Shi et al. (2023) rely on FlyingChairs Dosovitskiy et al. (2015), FlyingThings3D Mayer et al. (2016), Sintel Butler et al. (2012), and Spring Mehl et al. (2023). Multi-view stereo models and monocular depth models such as MVSAnywhere Izquierdo et al. (2025) and Depth Anything V2 Yang et al. (2024) often rely on BlendedMVS Yao et al. (2020), HyperSim Roberts et al. (2021), and other synthetic datasets. Such synthetic data provides accurate geometric ground truth to supervise training.

Generating synthetic data using computer graphics requires a large dataset of 3D assets. A common approach is to use assets created by artists Roberts et al. (2021); Wang et al. (2020) or from video games Hu et al. (2021). Another approach is to scan and reconstruct real-world objects Yao et al.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(2020). However, these methods are labor-intensive, as they require significant human effort for each additional asset.

An alternative approach is procedural generation, which creates data entirely through mathematical algorithms and rules. Procedural dataset generation has several advantages: it offers an infinite variety of randomized outputs, it allows full control of dataset properties, and it can generate large, diverse datasets at minimal cost.

A fundamental challenge in procedural generation is how to design the procedural rules. Given the vast design space of procedural rules, we ask: What kinds of rules are needed? Do we need rules to cover many different categories of objects? What rules are effective for different tasks? These questions remain understudied.

In this paper, we explore the design space of procedural rules for multi-view stereo (MVS), a fundamental task in 3D vision. MVS aims to reconstruct a 3D scene using a collection of images taken from multiple camera views. It has many downstream applications, including autonomous driving, robotics, and augmented reality (AR). MVS is one of the most common configurations for dense 3D reconstruction whenever as few as two camera views are available.

We examine the hypothesis: Does there exist a small set of rules that can generate effective training data? The question is intriguing because the typical training data for multi-view stereo covers diverse realistic scenes. To achieve realism in procedural generation, one would need many rules to cover a wide range of object categories in the real world. On the other hand, there is reason to believe that realism is unnecessary, as a general MVS system should reconstruct *arbitrary* shapes with *arbitrary* materials, not just shapes of particular object categories like cars or laptops.

We are not the first to pose this hypothesis. MegaSynth Jiang et al. (2025) provides large-scale synthetic data with procedural shapes, and Shape Evolution Yang & Deng (2018) jointly generates synthetic shapes while training a deep network. However, they address different questions. First, their tasks differ: MegaSynth focuses on novel view synthesis (and performs significantly worse than our approach when applied to MVS; see Sec. 5.1); and Shape Evolution focuses on the shape-from-shading task. Second, MegaSynth is not fully procedural: it uses a library of non-procedural image textures for materials.

In this paper, we demonstrate that we can generate effective training data for multi-view stereo using a small set of simple rules. Specifically, we introduce SimpleProc, a procedural generator that constructs shapes as Non-Uniform Rational Basis Splines (NURBS) surfaces through a lofting process, derives textures and materials from simple patterns such as Perlin noise, and arranges these shapes at various scales within the scene (see the top row of Fig. 1).

We evaluate our generated data by training MVSAnywhere and benchmarking on the Robust Multi-View Depth (RMVD) benchmark Schröppel et al. (2022). Under a fixed data budget of 8,000 images, our data achieves superior results compared to both MegaSynth and the curated 8-dataset baseline. By scaling the data budget to 352,000 images, we achieved similar, and in some benchmarks, even better results compared to the current state-of-the-art trained on over 692,000 manually curated images.

Our contributions are twofold: (1) we show that state-of-the-art MVS performance can be reached from data based on a small set of simple rules; (2) we provide a procedural generator and a large-scale dataset for MVS.

2 Related Work

2.1 Multi-View Stereo

Multi-View Stereo (MVS) has many important applications including autonomous driving, robotics, and augmented reality (AR), and it is one of the most common settings requiring as few as two camera views. Many approaches focus on optimizing performance in individual benchmarks, e.g., works in the vein of MVSNet Yao et al. (2018) focus on DTU Aanæs et al. (2016) and Tanks & Temples Knapitsch et al. (2017), while those in the vein of PatchMatchNet Wang et al. (2021) focus on the performance on ETH3D Schöps et al. (2017). Subsequently, the Robust Multi-view Depth (RMVD) benchmark Schröppel et al. (2022) was established, which consists of five diverse datasets:

KITTI Geiger et al. (2012), ScanNet Dai et al. (2017), ETH3D, DTU, and Tanks & Temples. To achieve robustness on RMVD, MVSAnywhere Izquierdo et al. (2025) relies on large mixtures of curated training data. Rather than proposing architectural modifications, we choose MVSAnywhere as our baseline model and focus purely on the data side, investigating whether minimalist procedural data can replace manual curation.

2.2 Synthetic Data for MVS

Yao *et al.* introduced BlendedMVS Yao et al. (2020), which has become a standard training recipe for MVS models. BlendedMVS is not purely synthetic as it captures images of real object along manual designed trajectories. Training on it alone is not sufficient for superior performance in RMVD (as proved in the MVSAnywhere paper Izquierdo et al. (2025)). To further scale up the amount of training data, MVSAnywhere uses 8 datasets including synthetic data from games: Hypersim Roberts et al. (2021), TartanAir Wang et al. (2020), BlendedMVS Yao et al. (2020), MatrixCity Li et al. (2023), VKITTI2 Cabon et al. (2020), Dynamic Replica Karaev et al. (2023), MVSSynth Huang et al. (2018), and SAIL-VOS 3D Hu et al. (2021). It takes substantial effort to curate such datasets. In contrast, we scale our data in a different direction: the procedural methods.

2.3 Procedural Data

Procedural data offers the advantage of generating infinite variation through a compact set of rules. For instance, Infinigen Raistrick et al. (2023) and its indoor variant Raistrick et al. (2024) focus on photorealism but are computationally expensive.

Other frameworks, such as Kubric Greff et al. (2022), provide procedural pipelines for generating 3D scenes but rely on existing asset libraries. Similarly, MegaSynth Jiang et al. (2025) provides procedural scenes but uses non-procedural textures.

In addition, MegaSynth focuses on novel view synthesis tasks; it does not ablate the impact of individual procedural rules; and it serves as a complementary part of the training data to the real data, rather than a standalone solution.

In contrast, we aim to demonstrate the effectiveness of minimalist procedural data as the only data source. We design our generator with minimalist principles and detailed ablation experiments.

3 Task and Base Model

We focus on the task of multi-view stereo. Given a set of neighboring images $\{I_i \mid i = 0, 1, \ldots, N\}$ with known camera information, the model outputs a depth map for one of the views called the reference view (I_0). Usually, N is smaller than 10, but large-scale scene reconstruction is achieved by fusing many depth maps. RMVD Schröppel et al. (2022) is a benchmark evaluating the accuracy of the models across five diverse real-world datasets: KITTI Geiger et al. (2012), ScanNet Dai et al. (2017), ETH3D Schöps et al. (2017), DTU Aanæs et al. (2016), and Tanks & Temples Knapitsch et al. (2017).

We conduct our experiments using the current state-of-the-art model on this benchmark, MVSAnywhere Izquierdo et al. (2025) (DoubleTake License). Testing with MVSAnywhere shows that state-of-the-art MVS performance is reachable with simple, fully procedural data. However, our data generator is by no means specialized to MVSAnywhere.

Meanwhile, MVSAnywhere is also a representative case study as many contemporary MVS models share similar frameworks, such as MVSNet Yao et al. (2018), CasMVSNet Gu et al. (2020), CERMVS Ma et al. (2022), and MVSFormer++ Cao et al. (2024):

- Feature Extraction: Deep features are extracted from all the images using a shared encoder.
- Cost-Volume Construction (Global/Local): Depth hypotheses (D in total) are sampled across either the full d_{min} to d_{max} range or a local $d_{\text{current}} \pm \Delta d$ range for each pixel in $H \times W$ space. Feature maps are warped according to the depth, and matching scores are computed to form a cost volume of shape $D \times H \times W$.

- Depth Prediction: Either 3D convolutional regularization or a GRU unit is used to predict the depth probability distribution or an iterative update to the current depth estimate.

In their paper, MVSAnywhere is trained on the curated 8-dataset mixture (692,000 images) listed in Sec. 2.2. We aim to replace these datasets with procedural data generated using the pipeline described below.

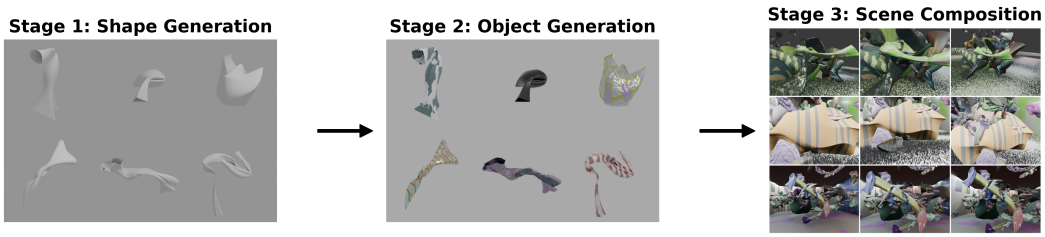

Figure 2: Procedural Data Generation Pipeline. Stage 1 generates NURBS surfaces from the lofting operation. Stage 2 applies displacements, textures, and material properties. Stage 3 arranges cameras, objects, lighting, and optional room boxes.

4 Data Generation Pipeline

Overview. Our data generation pipeline (Fig. 2) is based on Blender ble and utility functions from Infinigen Raistrick et al. (2023). It consists of multiple stages: Stage 1 - shape generation; Stage 2 - materials and mesh displacement; and Stage 3 - camera placement, object arrangement, lighting, and rendering.

4.1 Shapes

All of our shapes are generated by **lofting** a **profile** curve along a **stem** curve. The profile curve is a closed curve, and the stem curve is an open curve. The lofting process creates a NURBS surface by sampling different profile curves as cross-sections uniformly along the stem curve. A random scaling factor is also applied to the profiles, and the profiles are interpolated smoothly.

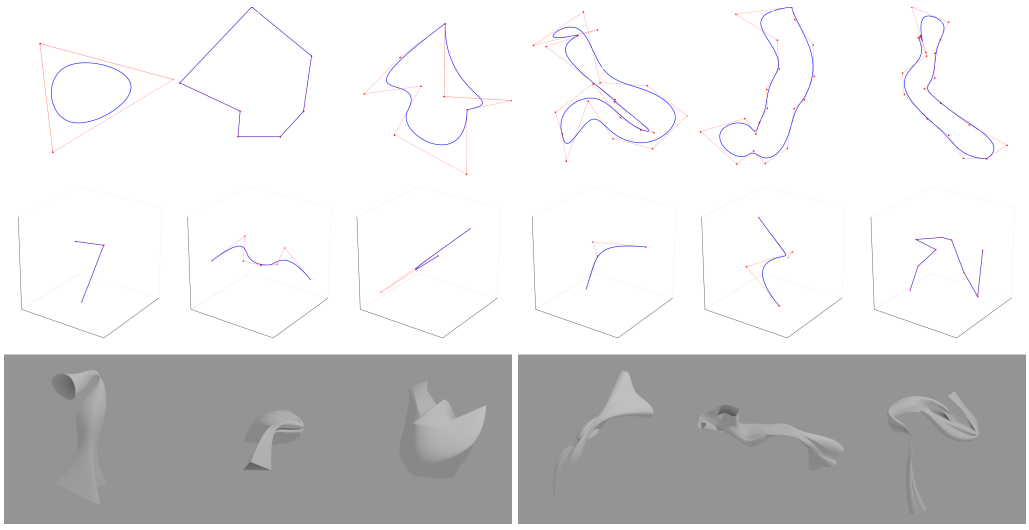

Figure 3: Shape generation pipeline. Top: Profile curves (Starfish and Reptile styles) with control points in red (only one of the profiles is shown for each shape). Middle: Stem curves from 3D random walks. Bottom: Resulting lofted NURBS surfaces showing diverse smooth and sharp features.

Both the profile and the stem are NURBS curves with a degree between 1 and 3, defined by a set of control points and a knot vector. The stem curve has control points generated from a 3D random walk. The profile curve follows one of two styles:

- Starfish style: We generate a set of control points uniformly in a circle, and then perturb each point with independent Gaussian noise in both radial and tangential directions. (Fig. 3, top-left).
- Reptile style: We generate a sequence of points from a 2D random walk and turn it into an open NURBS curve. Then we convert the open curve into a closed profile by offsetting it with a constant radius and fitting a closed NURBS curve to it. (Fig. 3, top-right).

The bottom of Fig. 3 shows several examples of the generated shapes through this lofting operation. Both smooth and sharp features are present in the shapes.

4.2 Displacement

We augment the base NURBS shapes with *micro-geometry*, meaning small-magnitude displacement of their vertices. This uses a combination of the two mechanisms below:

- Geometry Nodes: for each shape, we use Blender's brick texture, wave texture, or noise texture with random scale and magnitude along the normal direction. To alleviate low-poly artifacts, we use a relatively high resolution in the previous shape generation step followed by the "subdivide" operator.
- Displacement Socket in Shaders: sometimes, even the "subdivide" operator is too expensive to give sufficient geometry detail. Therefore, we use one of the three noise types in the displacement socket of the object's shader as well. This technique efficiently creates detail at very small scales.

The top row of Fig. 4 shows several example displacements applied to the shapes in Fig. 3.

4.3 Texture and Material Properties

Similar to the displacement, we use brick, wave, or Perlin noise patterns for the texture. For the wave and noise patterns, we convert their continuous values into two discrete regions using a threshold, and assign a distinct color to each region. Such textures are shown in the middle row of Fig. 4. We use Boolean operations to combine two such textures for each object (Fig. 4, bottom row); for example, the gray color in the leftmost shape is a boolean XOR of a Perlin noise pattern and a wave pattern.

The color values in the textures are uniformly sampled in HSV space. Note that the rendered images have different HSV distribution depending on other factors, including lighting, rendering engines, and post-processing.

Other material properties are randomly sampled from either a default value or a specified range. For example, roughness is set to 0.2 with 50% probability and uniformly sampled from $[0.2, 1.0]$ otherwise; metallic is set to 0 with 50% probability and sampled from $[0, 0.8]$ otherwise.

4.4 Camera Placement

To make the objects more efficiently used, we place cameras before adding objects into the scene. The position of the camera is sampled by randomly selecting the azimuth angle, elevation angle, and radius within predefined ranges in spherical coordinates (an elevation $\phi \in [-5°, 30°]$ and an azimuth θ spanning $45°$). Each camera is oriented to look at the scene center, with the z-axis defined as the up direction, and a small perturbation is applied to the final rotation. We place a total of eight cameras in each scene to maximize the diversity of the dataset. The horizontal field-of-view of each camera is randomly sampled within the range $[31°, 97°]$, and the aspect ratio is set to $3 : 4$ (height:width).

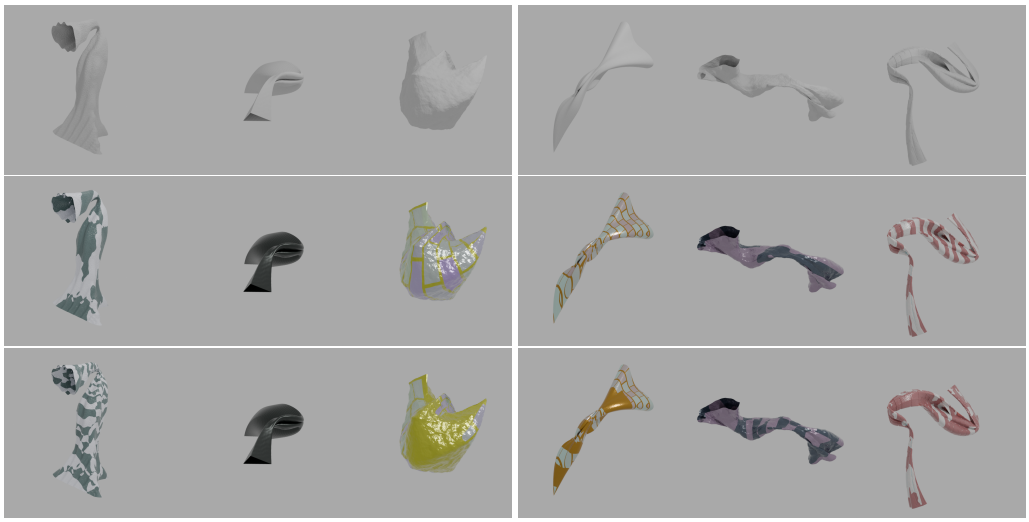

Figure 4: Shapes with Displacement and Textures. The top row shows displacements applied to base shapes. The middle row shows textures from brick, wave, or Perlin noise pattern. The bottom row shows complex textures created by boolean operations.

4.5 Object Arrangement, Lighting and Rendering

We place n_0 large objects and n_1 small objects in the scene. One of the large objects is always in the center as the main object. For the other large objects, we ensure they are visible in at least half of the cameras.

After the large objects are placed, we place small objects in a mixture of two ways:

- Uniform: We compute the bounding box of all the large objects and place a small object uniformly within it.
- Clustered: We sample a large object and sample a surface point on it to place a small object. This mimics realistic placement and contact between objects.

Optionally, we add an enclosing room box with a 50% probability to simulate an indoor environment with large planar surfaces. We apply textures (the same as Sec. 4.3) to the room box, but not displacement. We also optionally scatter small objects on a ground plane, again with a 50% probability, to mimic an outdoor grass field.

Area lights are placed on a plane above the scene. When a room box is present, lights are placed within the room. For rendering efficiency, we render using the EEVEE engine in Blender.

Please see the Fig. 6 in the Appendix for 10 random examples of our generated scenes.

5 Experiments

Scene Generation We render our dataset on a cluster. Scene generation takes 490 s per scene on average on a single CPU core with 64 GB RAM, and rendering takes 262 s per image on an NVIDIA RTX 3090. Generation is fully parallel across scenes and the render jobs use various NVIDIA GPU models (including RTX 2080Ti and RTX 3090).

Training and Evaluation We train the MVSAnywhere model Izquierdo et al. (2025) and evaluate the model following RMVD benchmark Schröppel et al. (2022). We split each dataset equally into **a validation set** and **a test set**. We optimize our data generator using the validation set and we report the score on the test set. For ScanNet, we use the frame selection from the MVSAnywhere paper. We also undistort both the images and the depth maps in ETH3D dataset as pointed out in that paper. The training and evaluation were performed using NVIDIA RTX 3090 and RTX A6000 GPUs.

We run each training process **3 times** and report the mean performance for the relative error (rel) and threshold accuracy (τ) across each benchmark and the aggregate score; we also report the **standard deviation** of the performance on the aggregate score.

5.1 FIXED-BUDGET COMPARISON

To isolate data quality from dataset size, we fix the budget to be 8,000 images. We maximize the total number of scenes as this provides the most diversity. A minimum of 8 images per scene is required for MVSAnywhere to train in a multi-view setting. We select such a subset of 1000 scenes and 8 images per scene from (a) training set of MVSAnywhere (8-dataset) and (b) MegaSynth Jiang et al. (2025). We generate an equivalent amount of 8,000 images with SimpleProc, which takes approximately 580 GPU-hours.

When we sample images from the 8-datasets mixture, we preserve the original dataset proportions to ensure the best performance under the current budget. MegaSynth does not contain covisibility information for pair selection in multiview stereo. Therefore, we select the images based on the following rule: given one camera view and an inter-camera angle threshold, we identify other views within this threshold and select the closest 7 views as its neighbor views. We randomly pick 1000 such tuples in 1000 scenes. We optimize the threshold on the validation split and report the results with the best threshold on the test split.

We performed three training runs of 200,000 steps each with a batch size of 3 and show the results in Table 1.

Test Split	KITTI		ScanNet		ETH3D		DTU		T&T		Average	
	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
8-dataset	4.56	55.68	**4.38**	**56.83**	4.32	80.54	1.37	93.39	2.96	79.54	3.52 ($\pm$ 0.03)	73.20 ($\pm$ 0.11)
MegaSynth	4.69	54.09	5.97	49.62	4.71	79.70	3.05	94.00	3.65	78.26	4.41 ($\pm$ 0.42)	71.13 ($\pm$ 0.24)
Ours	**3.41**	**65.64**	5.19	53.29	**3.98**	**85.57**	**0.84**	**96.05**	**2.93**	**85.12**	**3.27** ($\pm$ 0.06)	**77.13** ($\pm$ 0.20)

Table 1: Performance of models trained on a fixed budget of 8,000 images. Our procedural data demonstrates superior performance across nearly all benchmarks, with the exception of ScanNet.

Our model achieves significantly better results on all the benchmarks except for ScanNet, resulting in a much higher average score, showing the data efficiency of our approach.

Note that ScanNet is different from other datasets in two aspects: first, it contains noisier ground truth due to sensor limitations; second, many keyframe regions lack covisibility in neighboring views, which forces the model to rely on monocular priors rather than geometric 3D correspondences, favoring models trained on the 8-dataset mix due to its similar indoor priors.

5.2 UNLIMITED-BUDGET COMPARISON

Next, we compare performance of our data against the original MVSAnywhere training data. We rendered 352,000 images in total using our procedural generator, which takes about 25,600 GPU-hours, including variant configurations during generator development. We performed three training runs of 1,600,000 steps each with a batch size of 6 and show the results in Table 2. In addition to the standard 5-benchmark average, we compute an average excluding ScanNet to evaluate performance on more reliable and typical benchmarks (for the reasons discussed in Sec. 5.1). We report results for both the test split and the entire dataset for reference, though the test split is the primary rigorous metric.

We excluded MegaSynth from this experiment as it has very high storage cost for full-scale training, and showed lower performance in the previous experiment. For the 8-dataset configuration, we use the off-the-shelf model (trained on over 692,000 images) rather than retraining (no standard deviation is reported for this data point).

Interestingly, our advantage is more pronounced on the test set, confirming that the generator does not overfit the validation split. The test set results show that we achieved comparable or sometimes even better scores across all benchmarks except ScanNet. Notably, this approach outperforms the "8-dataset" baseline on the KITTI τ metric, both ETH3D metrics, and the DTU rel metric. Excluding

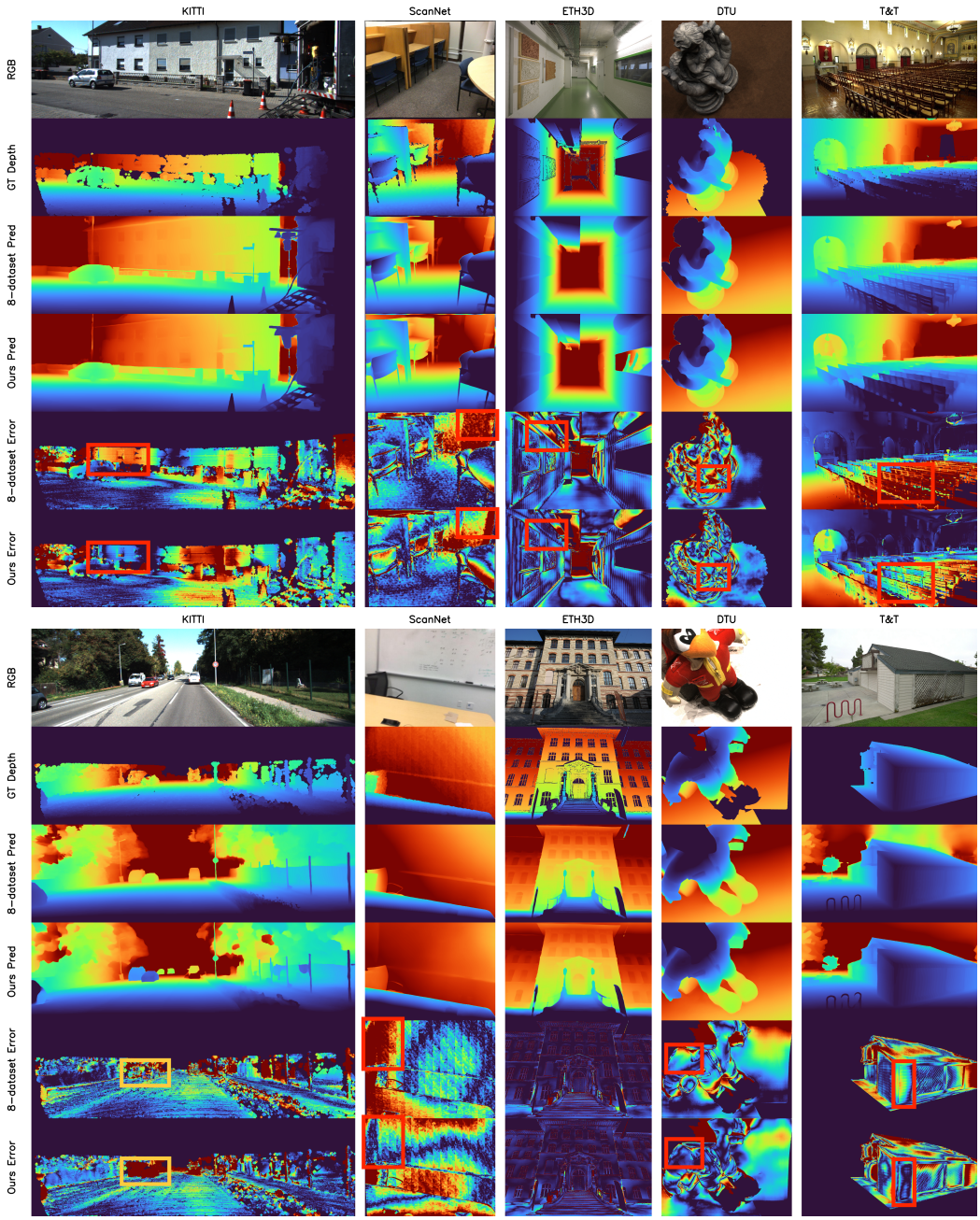

Figure 5: Unlimited-Budget qualitative comparison across five benchmarks. Our procedural training data yields competitive depth estimation results compared to the 8-dataset baseline.

ScanNet, our model achieves a 0.4% improvement on both τ and rel. Including ScanNet, our model is only 2.2% worse on τ.

In Figure 5, we show two random samples in each of the 5 benchmarks. We show the RGB image of the reference view (7 neighbor views are omitted), the ground-truth depth, prediction of the model trained from 8-datasets and prediction from the model trained on our data. Each example shows competitive results, with clear regions where our model performs better (in red boxes). The circle/stripe artifacts in the error map of ScanNet likely suggest RGB-depth desynchronization and inaccurate calibration in the ground-truth. In KITTI (yellow boxes), moving cars violate the static multi-view assumption. Our model treats the car as static and was misled. This may mean that

Test Split	KITTI		ScanNet		ETH3D		DTU		T&T		**Average**		**Avg ex-S**	
	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
8-dataset	**2.98**	71.02	**3.70**	**64.04**	3.52	88.58	0.86	**96.90**	**2.21**	**88.49**	**2.66**	**81.80**	2.39	86.25
Ours	3.00	**71.60**	5.06	53.47	**3.35**	**90.36**	**0.75**	96.58	2.43	87.92	2.92 (±0.02)	79.99 (±0.06)	**2.38** (±0.03)	**86.62** (±0.20)

Both splits	KITTI		ScanNet		ETH3D		DTU		T&T		**Average**		**Avg ex-S**	
	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
8-dataset	**3.24**	**68.80**	**3.69**	**65.09**	**3.32**	**88.75**	1.27	**94.98**	**2.13**	**90.48**	**2.73**	**81.62**	**2.49**	**85.75**
Ours	3.39	67.84	4.95	55.49	3.43	88.67	**1.09**	94.20	2.42	89.86	3.06 (±0.03)	79.21 (±0.17)	2.58 (±0.04)	85.14 (±0.20)

Table 2: Comparing the 8-dataset baseline with Ours under an unlimited budget. Excluding ScanNet (Avg ex-S), our method achieves superior average performance on the Test Split.

changing only the training data of MVSAnywhere results in a greater focus on 3D correspondences rather than memorizing domain-specific priors, improving generalization ability.

Both of these quantitative and qualitative results demonstrate that procedural training data with a small set of rules is effective for multi-view stereo.

6 Ablation Study

We ablate the design of object shapes, room boxes, materials, displacements, object sizes, lighting, and camera settings. The experimental setup, selection criteria, and full results are presented in Appendix A. Selected results are shown below:

- **Shape Type**: Instead of our proposed lofting-based NURBS shapes, we evaluated basic primitives, i.e., a mixture of cubes, spheres, and cylinders. We also tried a mixture of NURBS shapes and primitives. The proposed NURBS shapes achieve the best accuracy.
- **Number of Objects**: We evaluated the effect of the number of large objects (scaling small objects proportionally). The results show that increasing object count improves the performance, but we cap at 8 large objects to maintain practical efficiency.
- **Cameras**: Our default camera setting uses randomized ranges for FoV, camera-to-center distance, and inter-camera azimuth. We compared this against fixed settings for FoV and distance, as well as smaller or larger azimuth ranges. The results show that our default random range performs best.

6.1 Limitation and Future Work

The results, especially on ScanNet, could potentially be improved if we incorporate a small amount of real-world indoor data as priors. Furthermore, SimpleProc currently does not model dynamic objects, sensor noise, lens effects, or complex materials such as strongly specular and transparent surfaces. We leave these as future work.

7 Conclusion

In this paper, we built SimpleProc, a procedural data generation system based on NURBS and simple texture patterns and demonstrated that we can generate effective training data for multi-view stereo with fully procedural data from a small set of optimized rules.

AI USE STATEMENT

In this work, we used generative AI tools for creating software code and giving suggestions on improving the readability of the paper. We have not used generative AI tools for any other tasks. We have reviewed all AI-assisted work. Specifically, LLM-generated code was verified and tested for correctness, and editing suggestions were reviewed for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our complete anonymous source code and samples of our data are provided as supplementary material.

REFERENCES

Blender. `https://blender.org`.

Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *Int. J. Comput. Vis.*, 120(2):153–168, 2016.

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid (eds.), *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, volume 7577 of *Lecture Notes in Computer Science*, pp. 611–625. Springer, 2012.

Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *CoRR*, abs/2001.10773, 2020.

Chenjie Cao, Xinlin Ren, and Yanwei Fu. Mvsformer++: Revealing the devil in transformer's details for multi-view stereo. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2432–2443. IEEE Computer Society, 2017.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2758–2766. IEEE Computer Society, 2015.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 3354–3361. IEEE Computer Society, 2012.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Derek Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, A. Cengiz Öztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 3739–3751. IEEE, 2022.

Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 2492–2501. Computer Vision Foundation / IEEE, 2020.

Yuan-Ting Hu, Jiahong Wang, Raymond A. Yeh, and Alexander G. Schwing. SAIL-VOS 3d: A synthetic dataset and baselines for object detection and 3d mesh reconstruction from video data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 1418–1428. Computer Vision Foundation / IEEE, 2021.

Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2821–2830. Computer Vision Foundation / IEEE Computer Society, 2018.

Sergio Izquierdo, Mohamed Sayed, Michael Firman, Guillermo Garcia-Hernando, Daniyar Turmukhambetov, Javier Civera, Oisin Mac Aodha, Gabriel J. Brostow, and Jamie Watson. Mvsanywhere: Zero-shot multi-view stereo. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 11493–11504. Computer Vision Foundation / IEEE, 2025.

Hanwen Jiang, Zexiang Xu, Desai Xie, Ziwen Chen, Haian Jin, Fujun Luan, Zhixin Shu, Kai Zhang, Sai Bi, Xin Sun, Jiuxiang Gu, Qixing Huang, Georgios Pavlakos, and Hao Tan. Megasynth: Scaling up 3d scene reconstruction with synthesized data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 16441–16452. Computer Vision Foundation / IEEE, 2025.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Dynamicstereo: Consistent dynamic depth from stereo videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 13229–13239. IEEE, 2023.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):78:1–78:13, 2017.

Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 3182–3192. IEEE, 2023.

Zeyu Ma, Zachary Teed, and Jia Deng. Multiview stereo with cascaded epipolar RAFT. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXI*, volume 13691 of *Lecture Notes in Computer Science*, pp. 734–750. Springer, 2022.

Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4040–4048. IEEE Computer Society, 2016.

Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 4981–4991. IEEE, 2023.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 12630–12641. IEEE, 2023.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 21783–21794. IEEE, 2024.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Ángel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 10892–10902. IEEE, 2021.

Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2538–2547. IEEE Computer Society, 2017.

Philipp Schröppel, Jan Bechtold, Artemij Amiranashvili, and Thomas Brox. A benchmark and a baseline for robust multi-view depth estimation. In *International Conference on 3D Vision, 3DV 2022, Prague, Czech Republic, September 12-16, 2022*, pp. 637–645. IEEE, 2022.

Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 1599–1610. IEEE, 2023.

Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 14194–14203. Computer Vision Foundation / IEEE, 2021.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian A. Scherer. Tartanair: A dataset to push the limits of visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pp. 4909–4916. IEEE, 2020.

Yihan Wang and Jia Deng. WAFT: warping-alone field transforms for optical flow. *CoRR*, abs/2506.21526, 2025.

Dawei Yang and Jia Deng. Shape from shading through shape evolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 3781–3790. Computer Vision Foundation / IEEE Computer Society, 2018.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pp. 785–801. Springer, 2018.

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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 1787–1796. Computer Vision Foundation / IEEE, 2020.

A Ablation Study

We ablate the design of object shapes, room boxes, materials, displacements, object sizes, lighting, and camera settings. The results are presented in Tables 3 and 4.

Experimental Setup In each experimental block, only one feature is modified while others are kept constant within that block (including the random seed). While these fixed parameters may differ between blocks, we assume that the observed trends are independent of these variations. We observed that the advantage might shift between configurations as the dataset size grows. Therefore, we chose 1,000 scenes in total to ensure relatively stable results.

We perform three training runs of 200,000 steps each with a batch size of 3. We report the mean performance across all benchmarks and for the aggregate score, including the standard deviation of the aggregate score. We focus exclusively on the τ metric, as it is more robust.

Selection Criteria While individual ablation runs exhibit some noise, we follow the principle that multiple incremental improvements can accumulate in the final result. Therefore, in general, we select the setting with the best aggregate score regardless of the standard deviation. For some cost-sensitive parameters, e.g., the number of large objects, we cap parameter values to maintain data generation efficiency.

Individual Analysis

Validation Split	KITTI $\tau \uparrow$	ScanNet $\tau \uparrow$	ETH3D $\tau \uparrow$	DTU $\tau \uparrow$	T&T $\tau \uparrow$	Average $\tau \uparrow$
Shape Type						
Primitives	55.79	51.56	77.24	84.22	90.25	71.81 ($\pm$ 0.86)
Primitives + NURBS	60.12	53.54	81.55	88.37	92.32	75.18 ($\pm$ 0.54)
NURBS	**61.53**	**55.76**	**83.64**	**89.61**	**93.28**	**76.77** ($\pm$ 0.21)
Displacement						
No	61.37	**56.17**	80.24	86.32	91.81	75.18 ($\pm$ 0.16)
Yes	**62.62**	56.15	**81.21**	**89.79**	**92.20**	**76.40** ($\pm$ 0.50)
Materials						
Uniform Color	60.03	55.62	76.54	85.89	91.13	73.84 ($\pm$ 0.46)
w/ Noise Texture	**62.29**	**55.90**	78.91	87.85	**92.57**	75.51 ($\pm$ 0.04)
w/ Noise Texture + Boolean	62.07	55.87	**80.21**	**88.43**	92.34	**75.78** ($\pm$ 0.37)
Number of Large Objects						
1	57.15	52.11	70.08	83.29	90.55	70.64 ($\pm$ 1.20)
2	59.01	52.51	76.22	83.68	92.05	72.69 ($\pm$ 0.67)
8	**62.72**	**55.84**	**79.82**	**87.99**	**93.05**	**75.88** ($\pm$ 0.23)
Small Objects						
None	60.71	55.82	81.43	86.49	91.68	75.23 ($\pm$ 0.57)
Uniform Placement	61.12	**56.63**	81.22	**90.23**	91.46	76.13 ($\pm$ 0.16)
Clustered Placement	**62.08**	55.42	80.33	88.84	**92.51**	75.84 ($\pm$ 0.13)
50-50 Clustered and Uniform	61.73	55.68	**81.97**	89.71	92.06	**76.23** ($\pm$ 0.28)
Small Objects: Number and Size						
160; Smaller	60.86	56.40	81.92	90.21	91.59	76.20 ($\pm$ 0.32)
160; Larger	**61.49**	56.23	**83.23**	90.03	91.98	76.59 ($\pm$ 0.42)
320; Smaller	61.12	56.63	81.22	90.23	91.46	76.13 ($\pm$ 0.16)
320; Larger	61.47	**57.05**	83.19	**90.94**	**92.25**	**76.98** ($\pm$ 0.13)
Room Box						
w/o	60.52	53.03	75.52	**88.94**	92.47	74.10 ($\pm$ 0.40)
w/	**62.12**	**55.47**	**78.40**	88.83	**92.59**	**75.48** ($\pm$ 0.32)
Scattered Tiny Objects						
w/o	61.20	54.24	**80.68**	86.60	92.31	75.01 ($\pm$ 0.38)
w/	**62.12**	**55.47**	78.40	**88.83**	**92.59**	**75.48** ($\pm$ 0.32)

Table 3: Ablation study on procedural rule design choices for object geometry, surface materials, and scene arrangement.

Validation Split	KITTI $\tau \uparrow$	ScanNet $\tau \uparrow$	ETH3D $\tau \uparrow$	DTU $\tau \uparrow$	T&T $\tau \uparrow$	Average $\tau \uparrow$
Number of Lights						
5 - 10	62.17	55.57	81.52	87.83	92.52	75.92 ($\pm$ 0.19)
5 - 20	62.44	55.96	**81.70**	87.83	92.10	76.01 ($\pm$ 0.17)
5 - 40	**62.49**	55.23	81.37	86.65	92.27	75.60 ($\pm$ 0.51)
5 - 80	62.48	55.86	80.92	89.34	92.71	76.26 ($\pm$ 0.10)
40	**62.49**	56.07	80.14	88.25	92.71	75.93 ($\pm$ 0.16)
80	62.13	**57.06**	80.21	**90.59**	**93.36**	**76.67** ($\pm$ 0.35)
Camera Settings						
Constant Small FoV	57.04	50.52	66.22	83.33	89.75	69.37 ($\pm$ 0.30)
Constant Medium FoV	60.44	53.19	74.96	88.99	92.91	74.10 ($\pm$ 0.12)
Constant Large FoV	57.86	51.24	74.32	83.28	92.13	71.77 ($\pm$ 0.49)
w/o Distance Change	61.66	53.03	76.50	87.50	92.68	74.27 ($\pm$ 0.33)
Larger Azimuth Range	59.60	**56.26**	78.70	80.07	90.89	73.10 ($\pm$ 0.79)
Smaller Azimuth Range	62.45	54.93	79.62	**89.16**	92.09	75.65 ($\pm$ 0.15)
Default	**62.72**	55.84	**79.82**	87.99	**93.05**	**75.88** ($\pm$ 0.23)

Table 4: Ablation study on lighting and camera parameters.

- **Shape Type**: Instead of our proposed lofting-based NURBS shapes, we evaluated basic primitives, i.e., a mixture of cubes, spheres, and cylinders. We also tried a mixture of NURBS shapes and primitives. The proposed NURBS shapes achieve the best accuracy.
- **Displacements**: We compared the configuration with and without our noise-based displacement. The results show that the proposed displacement constantly improves the performance.
- **Materials**: We compared three configurations: (a) uniform (single color) for each object, (b) with noise texture but without the proposed Boolean modification, and (c) with noise texture and the boolean modification. Configuration (c) achieves the best score.
- **Number of Objects**: We evaluated the effect of the number of large objects (scaling small objects proportionally). The results show that increasing object count improves the performance, but we cap at 8 large objects to maintain practical efficiency.
- **Small Objects**: We compared several configurations: None, Uniform, Clustered (around large objects), and Hybrid (Uniform + Clustered). The results show that the Hybrid configuration performs best. We also evaluated the impact of number and size of the small objects. The results show that increasing both the count and scale of small objects improves the performance.
- **Room Box**: We compared the configurations with and without the room box. The room box improves the performance on all the benchmarks, especially for ETH3D.
- **Scattered Tiny Objects**: We compared the configurations with and without the scattered tiny objects. These objects improve the performance on all the benchmarks except ETH3D, likely because the ground planes in ETH3D are mostly flat.
- **Lighting**: We evaluated the impact of the number of lights, including random configurations. The results show that increasing the number of lights improves the performance, but we cap at 80 lights in total.
- **Cameras**: Our default camera setting uses randomized ranges for FoV, camera-to-center distance, and inter-camera azimuth. We compared this against fixed settings for FoV and distance, as well as smaller or larger azimuth ranges. The results show that our default random range performs best.

B Random Data Samples

Fig. 6 shows 10 random examples of our generated scenes. Each row contains 8 cameras in the same scene with suitable covisibility. We can see ground-scattered objects in rows 2, 3, etc, and room boxes in rows 4, 5, etc.

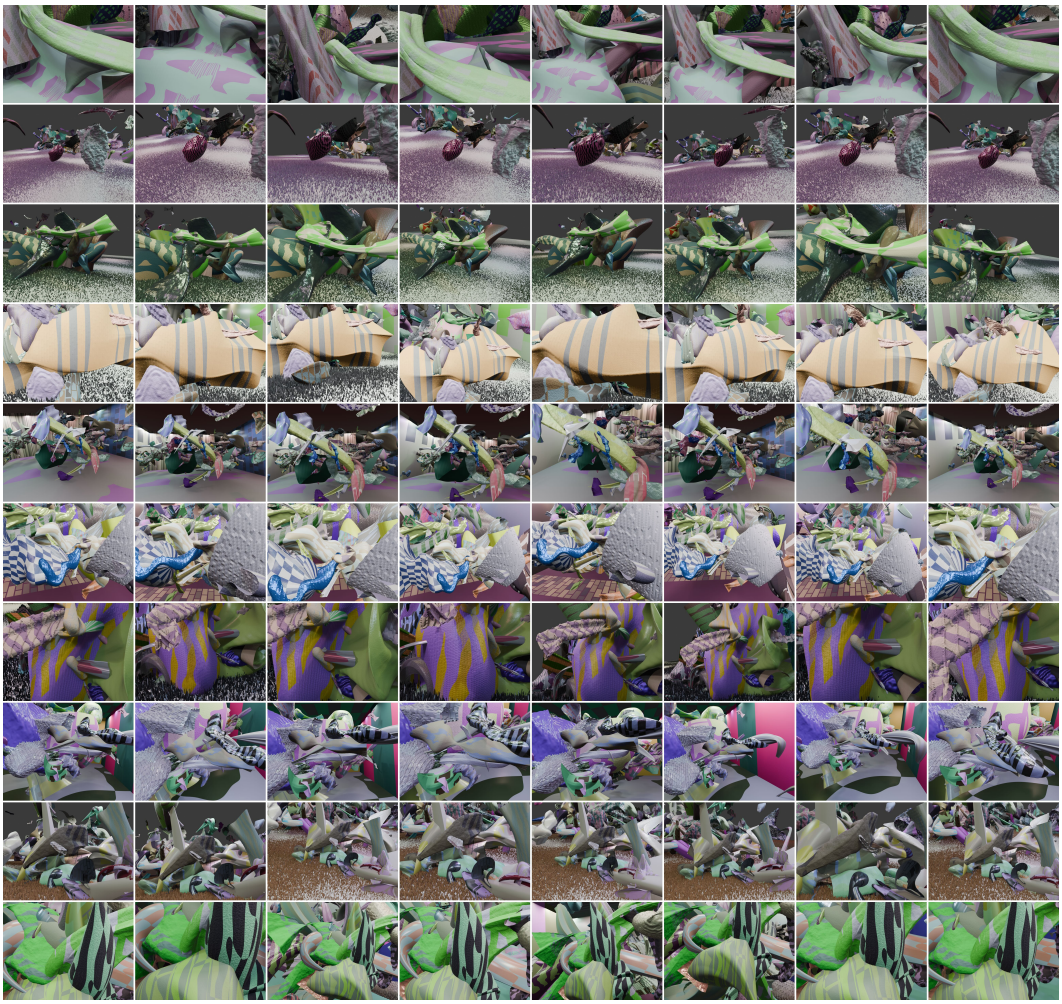

Figure 6: Gallery of 10 random examples of our generated scenes