

# Proc-GS: Procedural Building Generation for City Assembly with 3D Gaussians

## Supplementary Material

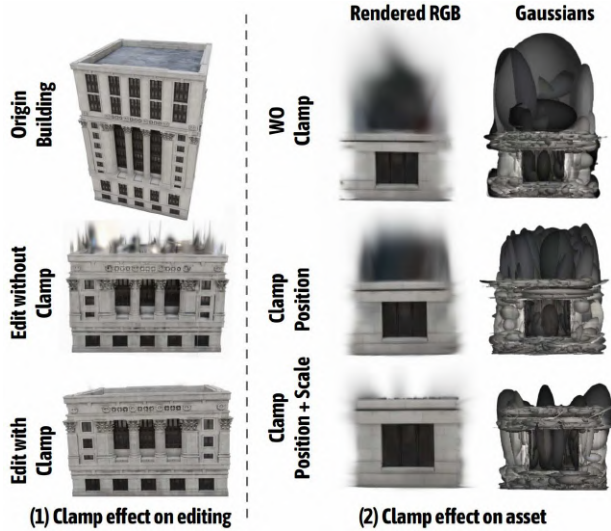


Figure 1. **(1) Clamp effect on editing.** Without clamp, the boundary area of edited scene is intensively corrupted by artifacts, making it impractical to create a new building with these assets; **(2) Clamp effect on asset.** We ablate effects of the clamp operation and demonstrate the effectiveness of both strategies.

## 1. More Dataset Details

Our *MatrixBuilding* Dataset consist of 17 buildings from the City Sample Project [1, 2], as shown in Figure 2 (a), which contains ground-truth procedural code and dense multi-view images. These buildings are created to mimic the building styles of Chicago, San Francisco, and New York. In Table 1, we also provide the number of building base assets and the total count of instantiated assets after assembling complete buildings according to procedural codes. The design of base assets combined with procedural codes significantly reduces the model size. Figure 2 (b) shows the dense camera capture trajectories. The ratio between training view and test view is about five to one.

## 2. Prompt Example

To illustrate the process of obtaining regular procedural code from raw data, we include an example of the prompt used in our framework in Figure 3. The goal is to summarize repetitive and scalable structures within raw data and represent them concisely using regular expressions of procedural code. The raw data represents the configuration of a multi-layered building with modular patterns. For example, a single row might include repetitive modules like *L1.W1*. Learning from one or more pairs of raw data and regular procedural codes, GPT-4o [3] could transform the raw

data into a regularized procedural representation. We transform verbose raw data into a structured, succinct procedural summary that distills the input’s intrinsic regularities while maintaining human interpretability.

## 3. More Qualitative Results

### 3.1. Sparse View

Figure 4 shows the sparse view qualitative results. Unlike 3D-GS [4], which suffers from significant artifacts when reducing training views, Proc-GS exhibits remarkable robustness. The proposed design of shared base assets enables a natural data augmentation mechanism, where base assets are dynamically influenced by all instances throughout the training process. This characteristic significantly enhances our ability to extract base assets from sparse image sets, thereby substantially lowering the overall data collection expenses.

### 3.2. Clamp Strategies

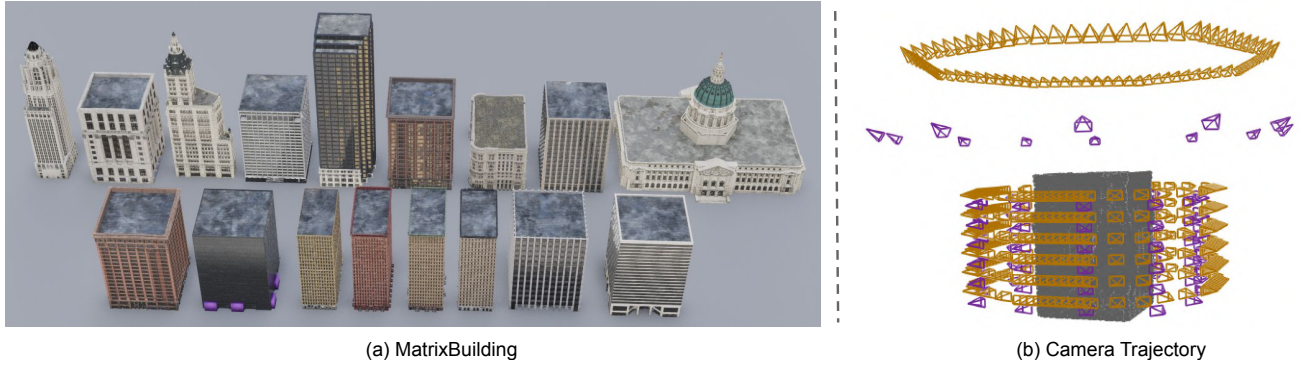
While building components maintain seamless connections, their Gaussian kernels often extend beyond asset boundaries during component decoupling. This overlap complicates asset combination and building editing, as demonstrated in Figure 1 (1). To resolve this issue, we introduced clamp strategies. A qualitative evaluation of our clamp operations in Proc-GS is presented in Figure 1 (2). The implementation of these two clamp strategies successfully achieves cleaner and more precise asset boundaries.

### 3.3. Building Editing

In Figure 5, we provide three building editing results from the real-world scene. We further showcase an intriguing building editing demo, where by manipulating variance assets, we precisely spelled out our method’s name on the building facade, thereby illustrating the remarkable controllability of our approach.

## References

- [1] <https://www.unrealengine.com/marketplace/product/city-sample>. 1
- [2] <https://www.unrealengine.com/>. 1
- [3] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1, 2
- [4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023. 1, 3



Building	CHB	CHD	CHE	CHF	CHG	CHH	CHI	CHJ	NYAA	NYAB	NYAE	NYAF	NYG	SFA	SFB	SFD	SFE
# Base Assets	90	30	90	32	24	19	43	8	17	24	25	37	56	54	81	12	20
# Total Assets	1559	345	1645	1170	617	1585	697	2409	1869	1920	1929	2831	438	295	821	729	1405



User

```

Procedural Code
L1_C1,(L1_W1)*,L1_W2,(L1_W1)*
L2_C1,(L2_W1)*,L2_W2,(L2_W1)*
L3_C1,(L3_W1)*,L3_W2,(L3_W1)*
L4_C1,(L4_W1)*,L4_W2,(L4_W1)*
L5_C1,L5_W1,(L5_W2,L5_W3)*,L5_W1,L5_W2
L1,(L2)*,L3,(L4)*,L5

```

Figure 3. **An example of the prompt used to obtain regular procedural code.** GPT-4o [3] takes the raw data, one or more examples as well as descriptions of procedural code as input and summarizes the regular procedural code as output.

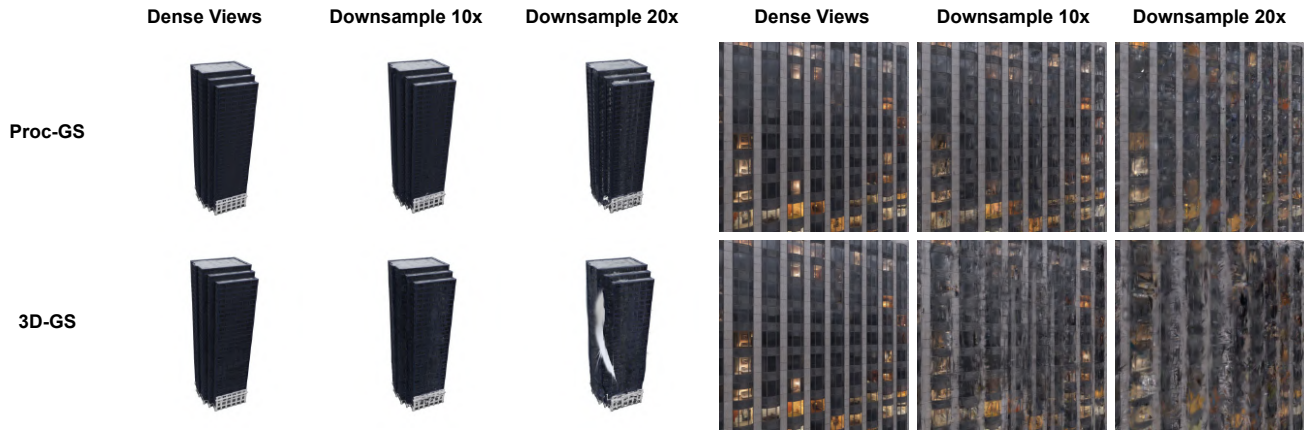


Figure 4. **Sparse view qualitative results.** Proc-GS demonstrates significant robustness when reduces the number of training views, in contrast to 3D-GS [4], which exhibits a pronounced susceptibility to numerous artifacts under similar conditions. This difference is attributed to the superior data efficiency inherent in our procedural code design, affirming its effectiveness in optimizing performance even with limited data inputs.

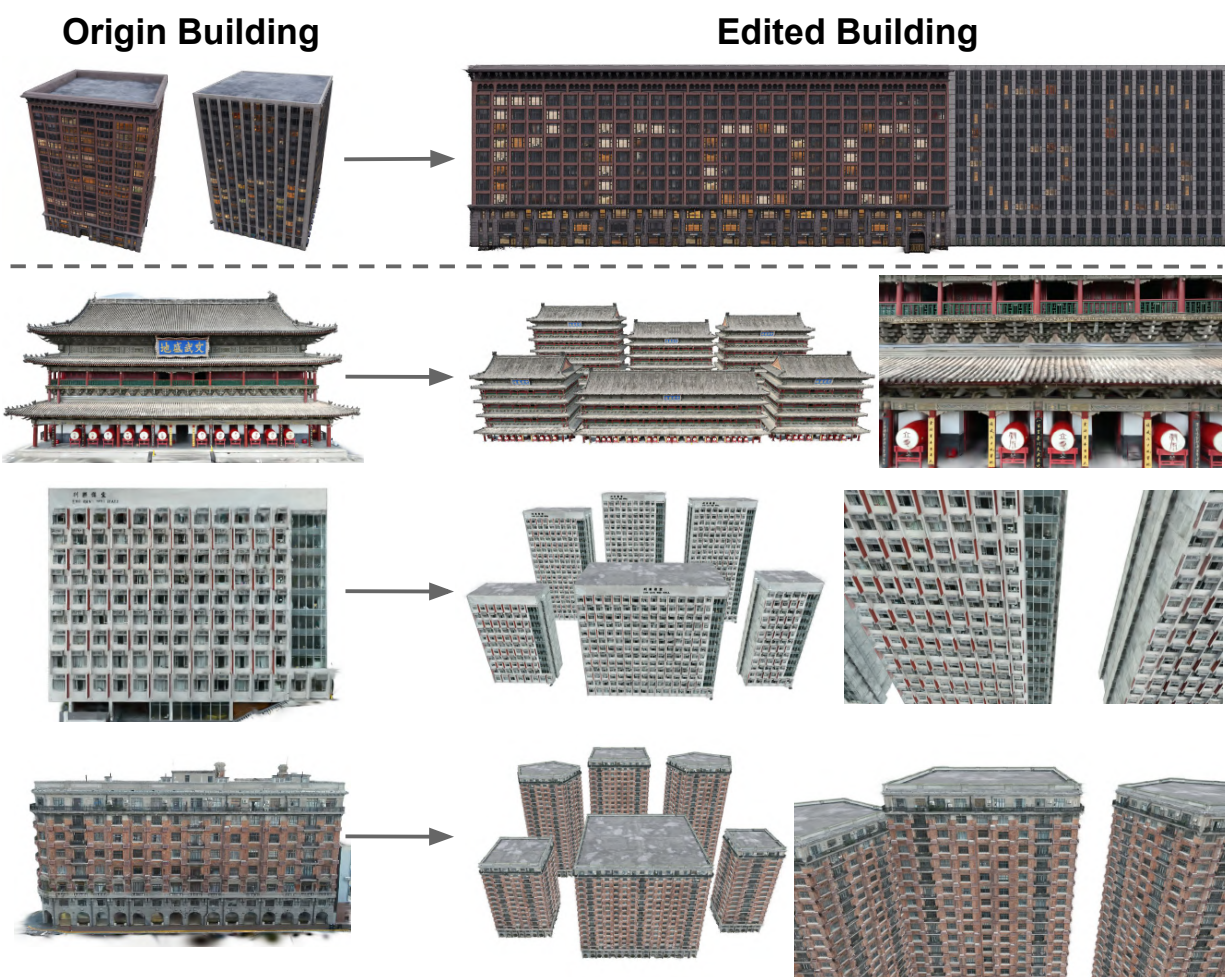


Figure 5. **Building Editing.** (1) The upper part shows synthetic data results, where we arranged variance assets to spell our method’s name, emphasizing its high controllability. (2) The lower part presents three editing results from the real-world scene.