Sat2City: 3D City Generation from A Single Satellite Image with Cascaded Latent Diffusion

Supplementary Material

6. More Technical details

6.1. Implementation Details

Our method is primarily implemented based on the sparse 3D structure provided by XCube [39] and the surface fitting network from NKSR [20]. For VAE training, we use 4 Nvidia A800-SXM4-80GB GPUs, while only the diffusion of dense and sparse geometric latent grids is trained across all 4 GPUs; Each layer of the multi-level appearance latent grids is trained on 2 GPUs. The inference process takes approximately **1 minute** on a single A800 GPU, while the appearance-mesh extraction takes around **20 seconds** for a resolution of 512^3. The training objectives for the two VAEs—dense geometry and re-hashed appearance—are defined as $\mathcal{L}_D$ and $\mathcal{L}_S$, respectively:

$$\begin{aligned}
\mathcal{L}_D^{\text{VAE}} &= \mathbb{E}_{\{G,A_N\}}[\mathbb{E}_{X_D\sim q_{\mathcal{E}_d}}[\lambda_0 \text{BCE}(G,\tilde{G})+ \\
&\lambda_1\mathcal{L}_1(A_N,\tilde{A}_N)]+ \\
&\lambda_2\mathbb{KL}(q_{\mathcal{E}_d}(X_D)\parallel p(X_D))], \\
\mathcal{L}_S^{\text{VAE}} &= \mathbb{E}_{\{G,A_N,P_C\}}[\mathbb{E}_{X_S\sim q_{\mathcal{E}_s}}[\lambda_0 \text{BCE}(G,\tilde{G})+ \\
&\lambda_1\mathcal{L}_1(A_N,\tilde{A}_N)+\lambda_3\mathcal{L}_1(P_C,\tilde{P}_C)]+ \\
&\lambda_2\mathbb{KL}(q_{\mathcal{E}_s}(X_S)\parallel p(X_S))],
\end{aligned} \tag{7}$$

where $\text{BCE}(\cdot)$ represents the binary cross-entropy for grid occupancy, and $\mathcal{L}_1$ denotes the L1 loss. Notably, the KL divergence $\mathbb{KL}(\cdot \parallel \cdot)$ is only computed for X_D and X_S, and not for X_{Cn}. Compared to geometric attributes, appearance attributes often exhibit greater variability and diversity. Introducing KL divergence directly in material learning may overly constrain the model, thus limiting the diversity and flexibility of material features. The training procedure for 3D latent diffusion follows the same structure for all bottleneck grids using v-parameterization [42] and the backbone from [12], with its 3D variant implemented by [39]. In practice, we set level $n = 4$, dual training starting epoch as $E = 10$, and the weighting factors are$\lambda_0 = 20, \lambda_1 = 50, \lambda_2 = 0.03, \lambda_3 = 50$.

6.2. Details on Cascaded Latent Diffusion

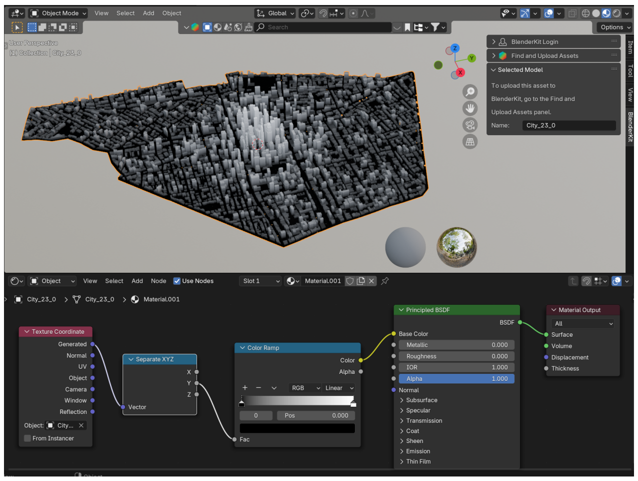
Our method adopts a cascaded latent diffusion pipeline to generate structured 3D cities with hierarchical geometry and appearance refinement. The pipeline follows a three-stage process: (1) dense geometry latent diffusion, (2) sparse geometry latent refinement, and (3) hierarchical appearance decoding:

Dense latent diffusion conditioned on height field. Inference begins by conditioning on a height map-derived point cloud, which serves as the structural prior for generation. A point encoder first maps the point cloud to voxel indices, extracts features via ResNet blocks, and aggregates them using max-pooling. The unordered point cloud features are then transformed into a structured voxel-based conditioning signal for subsequent encoding. The condition encoder further refines this representation by extracting hierarchical geometric features from sparse voxel grids using multi-scale sparse convolutions. Finally, the encoded condition latent grids are projected to match the resolution of the dense feature grid and concatenated with it for joint iterative denoising. The optimized dense latent diffusion is processed through the dense VAE decoder, generating the first-level sparse grid output (*1st grid*).

Sparse latent diffusion condition on decoded dense latent. The sparse latent diffusion serves as a bridge between geometry and appearance representations. It utilizes the *1st grid* to fit a sparse latent volume, facilitating finer-grain surface representation. Crucially, the sparse latent grid decoder records voxel *pruning decisions* at each upsampling step, guiding structured pruning during appearance decoding. This ensures consistency, as the appearance VAE does not explicitly encode geometric distributions.

Re-Hash latent diffusion condition on pruning decision from sparse latent decoder. The *pruning decisions* are subsequently leveraged to guide the upsampling process during appearance decoding, ensuring that the finest-level appearance feature grid maintains a consistent geometric structure with sparse latents at each upsampling step of the appearance decoder. By propagating these decisions, coarser-level appearance features (re-hashed from the finest-level) can dynamically adjust their voxel selection strategy based on the most refined masked voxel decisions. Specifically, at the bottleneck stage, the finest-level appearance feature grid undergoes a *re-hashing* process, generating multiple coarser levels. During each upsampling step, these coarser levels are iteratively refined by adapting to the pruned voxel structure defined by the finest-level grid. Rather than being naively upsampled, each coarser level is re-fitted to the structural surface, ensuring alignment with the progressively pruned geometry at that specific stage of upsampling. This hierarchical conditioning ensures spatial coherence and tightly constrains the appearance latent around the defined surfaces.

Once the geometry structure is established, the multi-level appearance latents are decoded by performing trilinear interpolation over the inferred geometry vertices. The interpolated latent features across all levels are concatenated



Metallic 0.000
Roughness 0.000
IOR 1.000
Alpha 1.000
0 Pos 0.000

Figure 13. Height Map Rendering in Blender.

and processed through an MLP-based output head to generate per-vertex appearance attributes, ensuring spatial consistency across the generated 3D scene.

Diffusion model with parameter Ψ. A widely adopted forward diffusion step (adding noise) is given by:

$$\mathbf{X}_t|\mathbf{X}_{t-1} \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{X}_{t-1}, \beta_t\mathbf{I}), \tag{8}$$

where the noise variance β_t is small, ensuring gradual corruption of data over T diffusion steps. The reverse process attempts to denoise $\mathbf{X}_t$ progressively, ultimately restoring the original data distribution. It is parameterized as:

$$\mathbf{X}_{t-1}|\mathbf{X}_t \sim \mathcal{N}(\boldsymbol{\mu}_\Psi(\mathbf{X}_t, t), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\mathbf{I}), \tag{9}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=0}^{t}\alpha_s$. The mean $\boldsymbol{\mu}_\Psi$ is parameterized by a learnable neural network. In practice, we re-express $\boldsymbol{\mu}_\Psi$ using:

$$\boldsymbol{\mu}_\Psi = \sqrt{\alpha_t}\mathbf{X}_t - \beta_t\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}}\mathbf{v}_\Psi, \tag{10}$$

where the network is trained to predict $\mathbf{v}$ instead of the noise, following the v-parameterization strategy, which has been found to improve optimization stability. Following XCube, the $v_\Psi(\cdot)$ is tailored for our sparse representations based on the one that was originally proposed for dense image space [12].

6.3. Details on Dataset Height Map Construction

We employ a procedural shading technique using a height-dependent grayscale gradient (Figure 17). The shading process is implemented within the Blender shading node, leveraging a combination of procedural texture coordinates, channel separation, and color mapping.

Coordinate-Based Height Extraction We begin by extracting the generated texture coordinates of the city mesh using the Texture Coordinate node. This provides a spatially varying reference frame that adapts dynamically to the geometry. The coordinate vector is subsequently decomposed into its components (x, y, z) through a Separate XYZ node, isolating the vertical height z for use in the shading computation.

Height-Driven Color Mapping The extracted height information is passed through a Color Ramp node, which maps height values to a grayscale color gradient. This mapping is crucial for simulating height-based shading variations, where lower buildings are rendered darker, and higher structures receive lighter intensities, mimicking ambient occlusion and environmental exposure effects.

Physically-Based Material Composition The resulting color gradient is then fed into the Base Color input of a Principled BSDF shader. This shader governs the material's light interaction properties, maintaining a physically consistent representation of urban structures. For enhanced realism, the roughness is set to 1.0 (diffuse reflection), and metallic properties are disabled to simulate non-reflective building surfaces.

7. Discussion

Limitations. The evaluation of our approach on real-world datasets remains pending, as obtaining high-resolution colorized point clouds precisely aligned with remote sensing elevation maps necessitates considerable resource allocation and may encounter limitations in data availability and dissemination.

Spoiler alert. We developed an automated pipeline leveraging Google Earth Engine and the Google Maps Platform to retrieve (i) high-resolution satellite imagery and (ii) co-registered photorealistic 3D Tiles for identical geographic bounding boxes. Figure 14 shows several such samples, and we have already begun constructing a large-scale dataset.

We evaluated the data processing workflow—comprising API-based data acquisition, point cloud conversion, and height map inference—on several commercially available desktop PCs. The complete pipeline requires approximately 10 minutes per scene, indicating that large-scale data collection is practically achievable within a standard academic laboratory environment.

As an initial step toward evaluating Sat2City's generalizability, we fine-tuned Depth Anything [62] on a dataset comprising real-world height maps and corresponding satellite imagery [57]. We inferred the scenes shown in Figure 14. Quantitatively, we computed the Chamfer Distance between point clouds derived from height maps and their colorized ground truth, yielding 0.2909 for synthetic data (Sat2City dataset) and 0.0977 for real-world data (Google Earth), both averaged over seven randomly

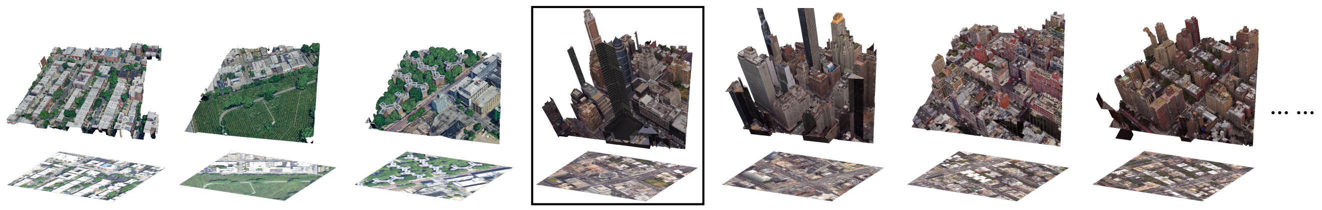


Figure 14. Real-world dataset includes diverse morphologies.

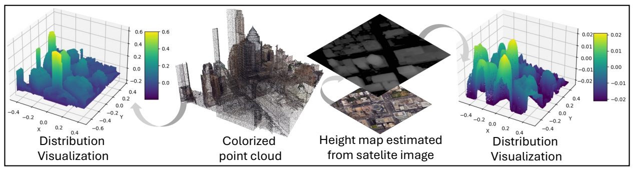

Distribution Visualization
Colorized point cloud
Height map estimated from satelite image
Distribution Visualization

Figure 15. Raw data example for one scene in Figure 14.

selected scenes. Qualitatively, Figure 15 (real) illustrates that the height map distribution in the real-world dataset exhibits greater overall consistency with the colorized point clouds compared to its synthetic counterpart shown in Figure 3 (synthetic). These results suggest that real-world data is unlikely to introduce generalization problems and may even improve the quality of generation.

Future works. Although contributing an additional real-world dataset is beyond the scope of this submission, as concurrent works (SynCity [13] and NuiScene [25]) likewise validate only on synthetic data, we plan to explore it in future work fully. Nevertheless, our findings underscore both the necessity and the promise of building real-world datasets to push this research direction forward. We also hope our work will inspire a series of follow-up studies in the remote sensing community to explore the model's limitations and possible improvements on such a real-world dataset—examining, for example, its performance in various urban environments (*e.g.*, diverse terrains, or mixed vegetation and building layouts), at different remote sensing resolutions.

8. More About Evaluations

8.1. User Study Settings

Table 4 presents our user study design, where participants were instructed to rate images based on the provided questions. The evaluation follows a 10-point scale, where 1 indicates "Very Poor" and 10 represents "Excellent." For each sample, participants answered two questions—either texture-based or geometry-based—ensuring a comprehensive assessment of both perceptual quality and structural fidelity.

Metric	User Study Question
TPQ	How would you rate how the scene looks overall? Think about the details, the textures, and how realistic it seems to you.
TSC	How well do you think the shapes and structure of the scene are represented? Does the geometry look accurate and complete?
GPQ	How would you rate how the scene looks overall? Think about the details, the textures, and how realistic it seems to you.
GSC	How well do you think the shapes and structure of the scene are represented? Does the geometry look accurate and complete?

Table 4. User study metrics and evaluation questions.

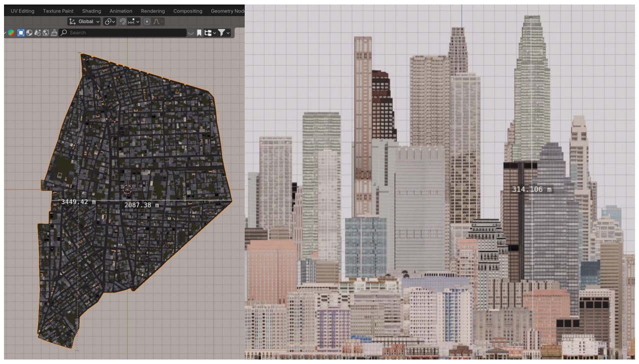


Figure 16. Orthographic camera view for synthetic satellite imagery and building distribution.

8.2. Texture Evaluation.

Our evaluation of the generated textures is limited to qualitative visual analysis, primarily because this work does not involve any neural rendering components that produce 2D images. Moreover, standard 2D generation metrics such as FID or KID are not applicable for assessing cross-modal outputs—namely, the input colorized point cloud and the resulting textured mesh.

8.3. More Dataset & Generation Snapshots

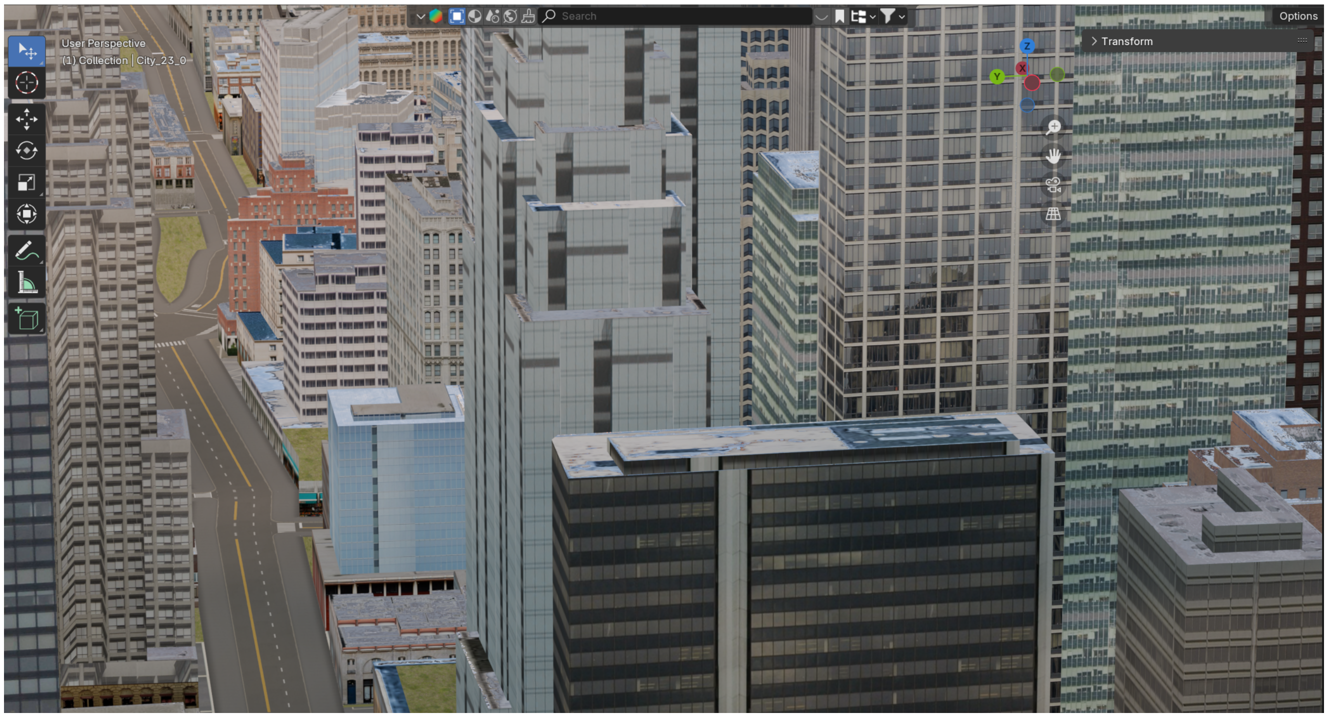


Figure 17. Exploring the original artist-designed mesh model.

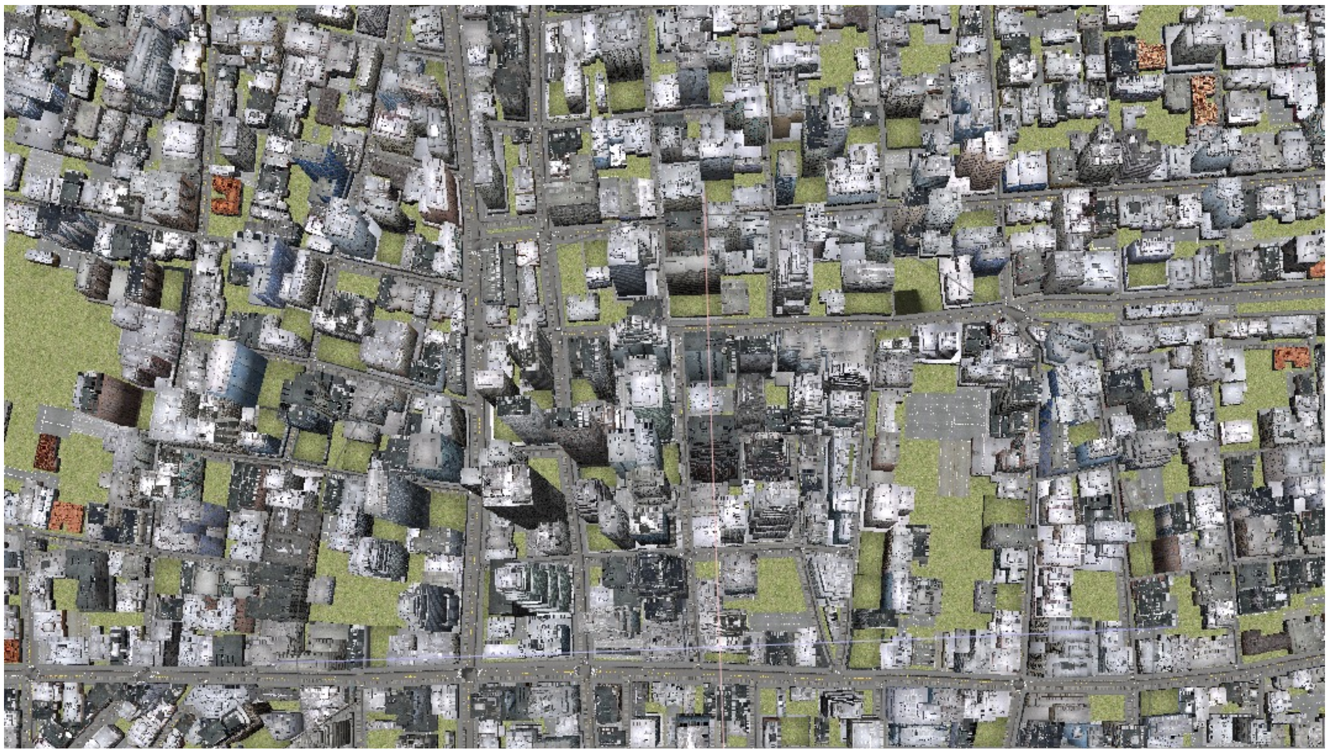


Figure 18. More views on the entire sampled point cloud.

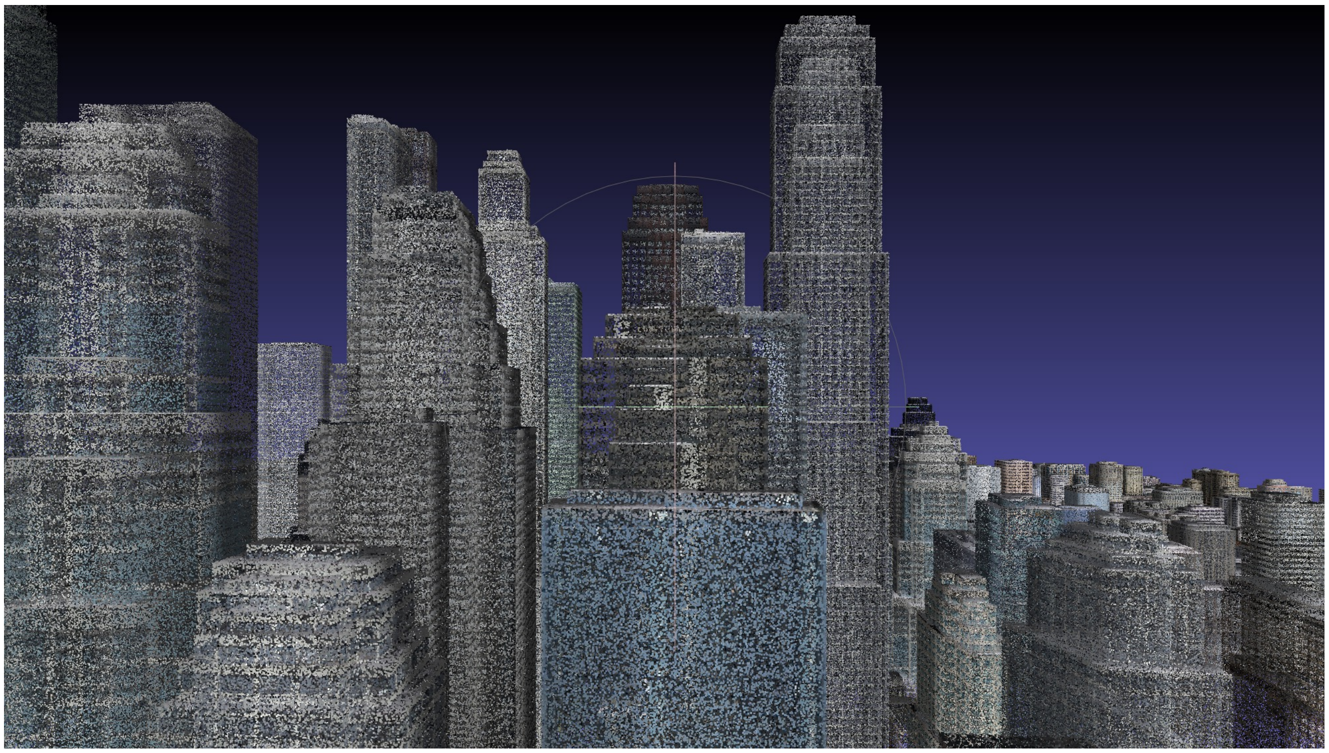
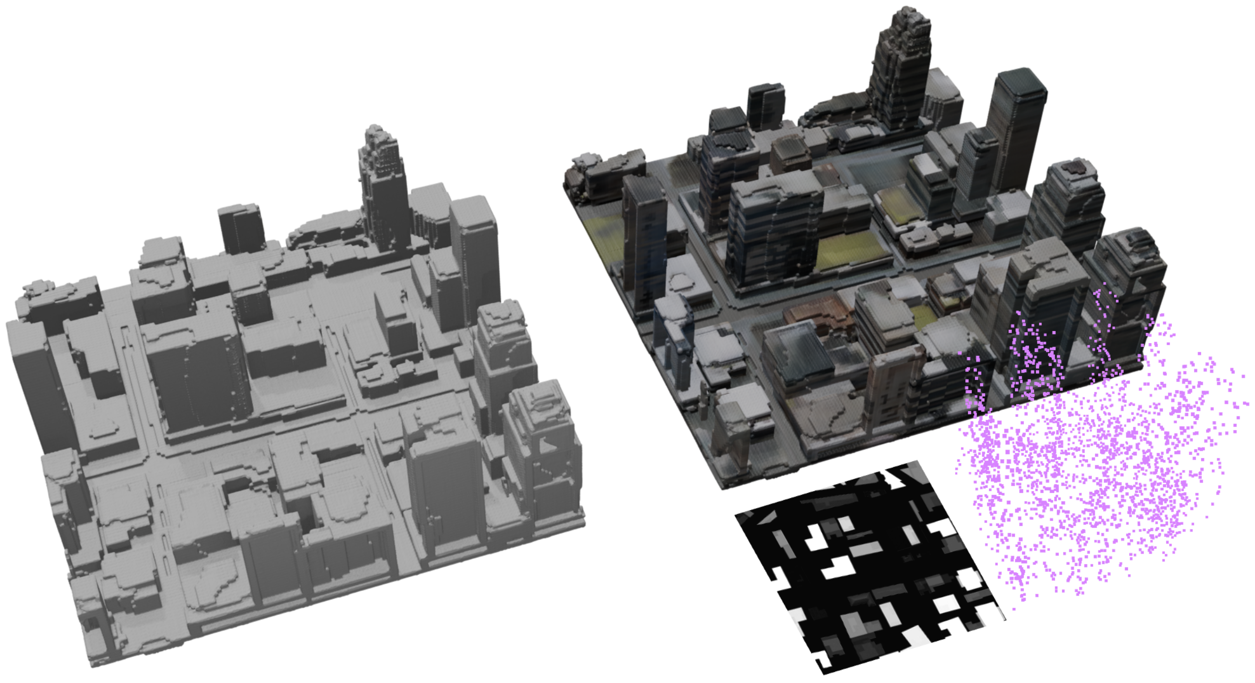


Figure 19. More views on zoomed-in sampled point cloud.



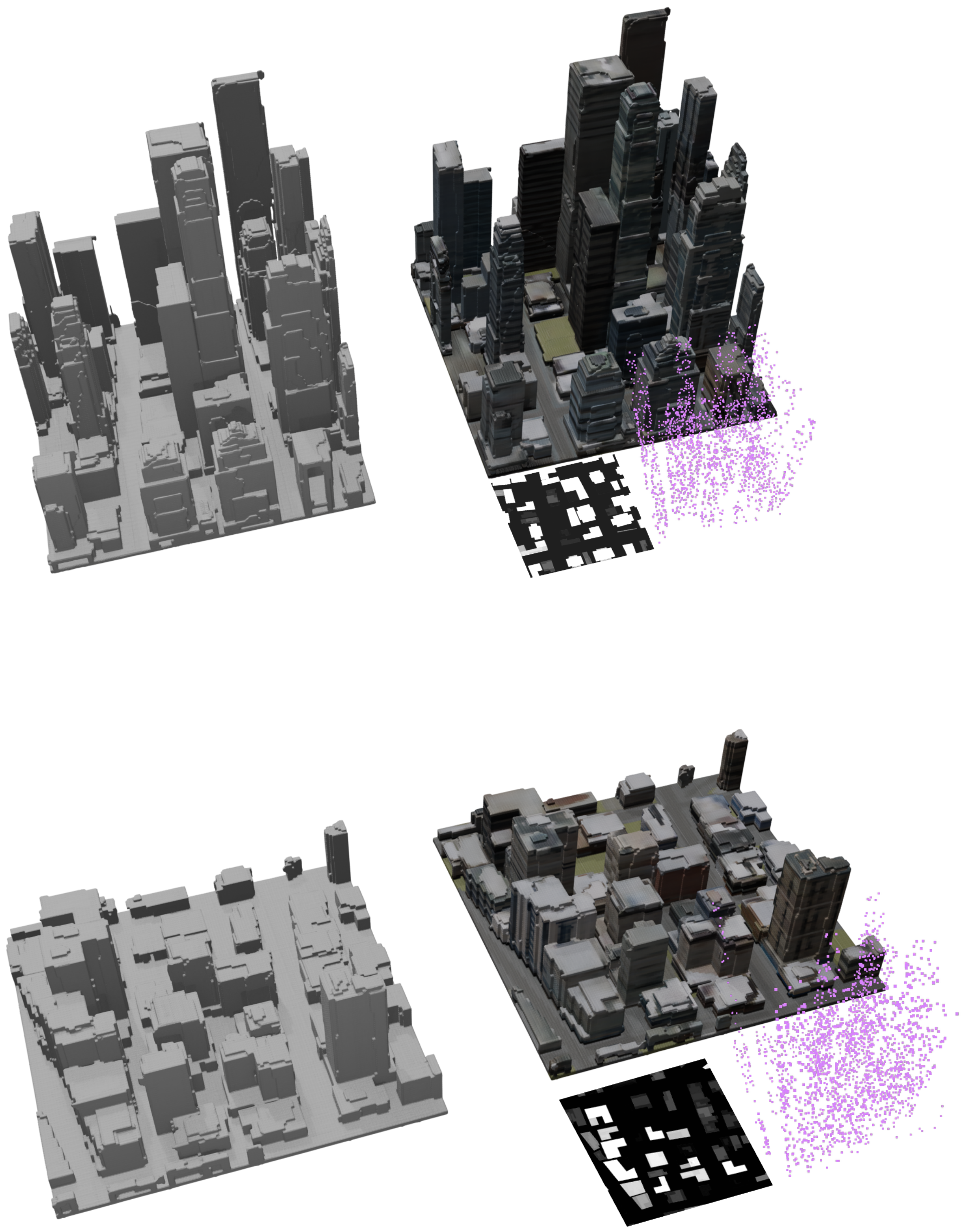


Figure 20. More results.