ArcPro: Architectural Programs for Structured 3D Abstraction of Sparse Points (Supplementary Material)

Qirui Huang[1,2], Runze Zhang[1], Kangjun Liu[2], Minglun Gong[3], Hao Zhang[4], Hui Huang[1*]
[1]CSSE, Shenzhen University [2]Pengcheng Laboratory [3]University of Guelph [4]Simon Fraser University

This supplementary material primarily includes implementation details, additional experiments and application results, as well as the motivation behind DSL.

A. Experiments

Network structures. The point cloud encoder is a ResNet-style architecture built with sparse 3D convolutions. It processes voxelized point clouds of size 128^3 and progressively reduces the spatial resolution through five stages, each consisting of residual blocks with sparse convolutions. The network begins with a basic sparse convolutional block and follows a structure where feature dimensions increase across stages: $[64, 128, 192, 384, 512]$. Each stage employs two residual blocks, with downsampling implemented using sparse max pooling. The encoded output is compressed into a $\mathbb{R}^{512}$ feature representation, corresponding to a spatial resolution of 4^3. The program decoder is a classical transformer with 12 layers, each featuring 8 attention heads, a model dimension $d_{\text{model}} = 512$, and a feed-forward dimension $d_{\text{ff}} = 2048$. The encoded point cloud features are incorporated into the program decoder via cross-attention, enabling effective conditional program generation that aligns with the input point cloud.

Training details. The input point cloud is normalized to fit within a unit cube $[-0.5, 0.5]$ by centering and scaling its coordinates based on the data's range. For tokenization, numerical values such as coordinates or height values are discretized within the range $[-1, +1]$. This range is divided into intervals with a resolution of 0.02, resulting in 100 distinct numeric tokens to represent the corresponding discrete values. We use a label smoothing strategy: non-numeric tokens have a ground truth probability of 0.95, with 0.05 distributed evenly among others; numeric tokens have 0.5, with 0.25 assigned to adjacent tokens to reflect their continuous nature for better optimization.

Ablation study. See Table 1. For data augmentation, both *noise scale* and *incomplete ratio* should be moderate: if too weak, they fail to adequately simulate the low-quality nature of real point clouds; if too strong, the problem becomes overly ill-posed, degrading performance and destabilizing training. For *token schema*, we use ⟨/p⟩ and ⟨/h⟩ as end tokens for point coordinates and height values, respectively. While parsing works without them, they improve performance and stabilize training. For *geometry refinement*, omitting it during inference has little impact on the metrics but noticeably degrades visualization due to slight misalignment of points and lines.

Table 1. Ablation study for training configuration.

Noise Scale	Incomplete Ratio	⟨/p⟩, ⟨/h⟩ Token	SfM		Sparse Sampling	
			HD (↓)	LFD (↓)	HD (↓)	LFD (↓)
0			0.0195	4192	0.0291	5387
0.05			0.0177	4338	0.0237	4958
	0%		0.0169	4210	0.0250	5033
	$[50\%, 90\%]$		0.0187	4396	0.0266	5211
		w/o	0.0181	4259	0.0272	5212
0.02	$[10\%, 50\%]$	w/	**0.0154**	**3873**	**0.0219**	**4932**

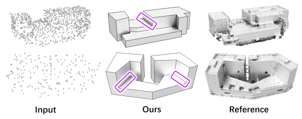

Figure 1. Generalization.

Table 2. Robust to data ratio.

Training data (4-gon : 6-gon)	4-gon		6-gon	
	#n	HD	#n	HD
20% : 80%	4.07	0.0089	5.95	0.0085
50% : 50%	4.03	0.0091	5.94	0.0089
80% : 20%	4.04	0.0087	5.95	0.0090

Generalizability and robustness. Our goal is to learn *conditional* mapping from input point clouds, where domain shifts between synthetic and real data can be mitigated since the input provides a contextual hint during inference. We cannot retrieve the most similar shape from the training set due to online data synthesis, but Figure 1 shows that *our method can infer unsynthesized or unseen shapes*. According to our procedural rules (see Sec 4.2), when $M > 1$, each edge should lie on the extension of a parent edge, but this is not satisfied in Figure 1 above. We also explored robustness against varying data ratios by preparing two test sets (4-gon and 6-gon) and three training sets with different mixing ratios; see Table 2.

*Corresponding author

Additional illustrations. We present examples of procedurally generated synthetic training data, as illustrated in Figure 3. These are generated online during training, including six types of architectural tree models, which are sampled based on specific proportions. More results of our method applied to Structure-from-Motion (SfM) point clouds and sparse sampling point clouds are provided, as shown in Figure 4. Furthermore, we examine the performance of RANSAC plane detection on low-quality point clouds derived from the experimental section of the main paper. As shown in Figure 5, these results reveal RANSAC's struggles with sparse point clouds, causing traditional methods to fail. Finally, we present user study examples comparing our method to alternative approaches. These examples are shown in Figures 9, 10, and 11.

B. Applications

Multi-view aerial images. We extend our framework to process raw SfM point clouds from multi-view aerial images, bypassing building segmentation. This introduces noise like ground points and outliers. To mitigate this, we augment data by expanding a building's footprint's convex hull or bounding box to simulate ground and adding noise to represent trees, cars, and other elements. This allows us to more effectively process unsegmented SfM point clouds. Compared to traditional MVS methods, ArcPro significantly improves inference speed while producing lightweight, textured 3D abstractions, as shown in Figure 6. ArcPro takes 0.034s on an RTX 4090 GPU, compared to 739s for the traditional MVS pipeline (using the commercial software ContextCapture), achieving a 10,000x reduction in time and data size (#V for vertices, #F for triangular faces). This allows faster processing, lower rendering costs, and more efficient data transfer and storage, which is critical for spatial computing applications.

Natural language retrieval. Our method encodes architectural structures as programs, leveraging their linguistic properties for natural language-driven analysis and retrieval using large language models prompt by DSL definition. ArcPro transforms a database of 3D architectural models into corresponding programmatic representations, establishing connections between programs and 3D models. For example, as shown in Figure 7, given a query like *"two-layer buildings where the second layer is higher than the first,"* a language model such as ChatGPT will generate Python code for an `IsMatching(program)` function based on the DSL definition, implementing the logic to verify each program. The function returns `True` for programs meeting the criteria and `False` otherwise, enabling the retrieval of relevant 3D architectural models effectively.

LiDAR point clouds. We also explore the performance of ArcPro on LiDAR point cloud input, using data from the DublinCity dataset [1], as shown in Figure 8. Even without incorporating specific design in data augmentation to simulate the characteristics of LiDAR point clouds, our method is still able to achieve reasonable performance.

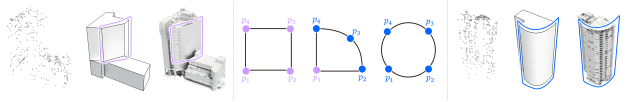
Figure 2. The examples of non-planar surfaces in the current and extended ArcPro framework.

Non-planar surfaces. As our work primarily focuses on recovering planar surfaces, curved surfaces, such as the one shown in Fig. 2 left, need to be approximated by polygonal contours. Extending our framework to handle non-planar contours is quite straightforward. By distinguishing curve points from corner points at the token level (marked in purple or blue), the geometry compiler can fit curve segments as parametric curves. We synthesize corresponding training data to obtain preliminary results shown in Fig. 2 right, highlighting the potential of our program framework.

C. Motivation of DSL

We design our DSL to align with architectural priors and be syntactically compatible with traditional procedural building generation (PBG), instead of using a more general shape language. This approach offers these advantages:

- *More compact representation* with a more efficient solution space. For example, unlike sketch-and-extrude, which requires six DoF (origin and orientation), our approach employs a parent layer index to simultaneously specify the 3D coordinate frame and layer hierarchy.
- *Leveraging mature PBG research* for large-scale training data synthesis, where architectural priors can be injected.
- *Extensibility* to accommodate new statements that support additional architectural features, such as roof structures from OpenStreetMap (OSM) or for-loops for repetitive elements like windows in façade modeling.
- *Explicit encoding of building properties*, such as hierarchical relationships in `CreateLayer` statements, facilitating language-based retrieval and analysis.
- *Editability* through parametric modeling and the relation of geometric elements align with architectural features.

References

[1] SM Zolanvari, Susana Ruano, Aakanksha Rana, Alan Cummins, Rogerio Eduardo Da Silva, Morteza Rahbar, and Aljosa Smolic. Dublincity: Annotated lidar point cloud and its applications. *arXiv preprint arXiv:1909.03613*, 2019. 2

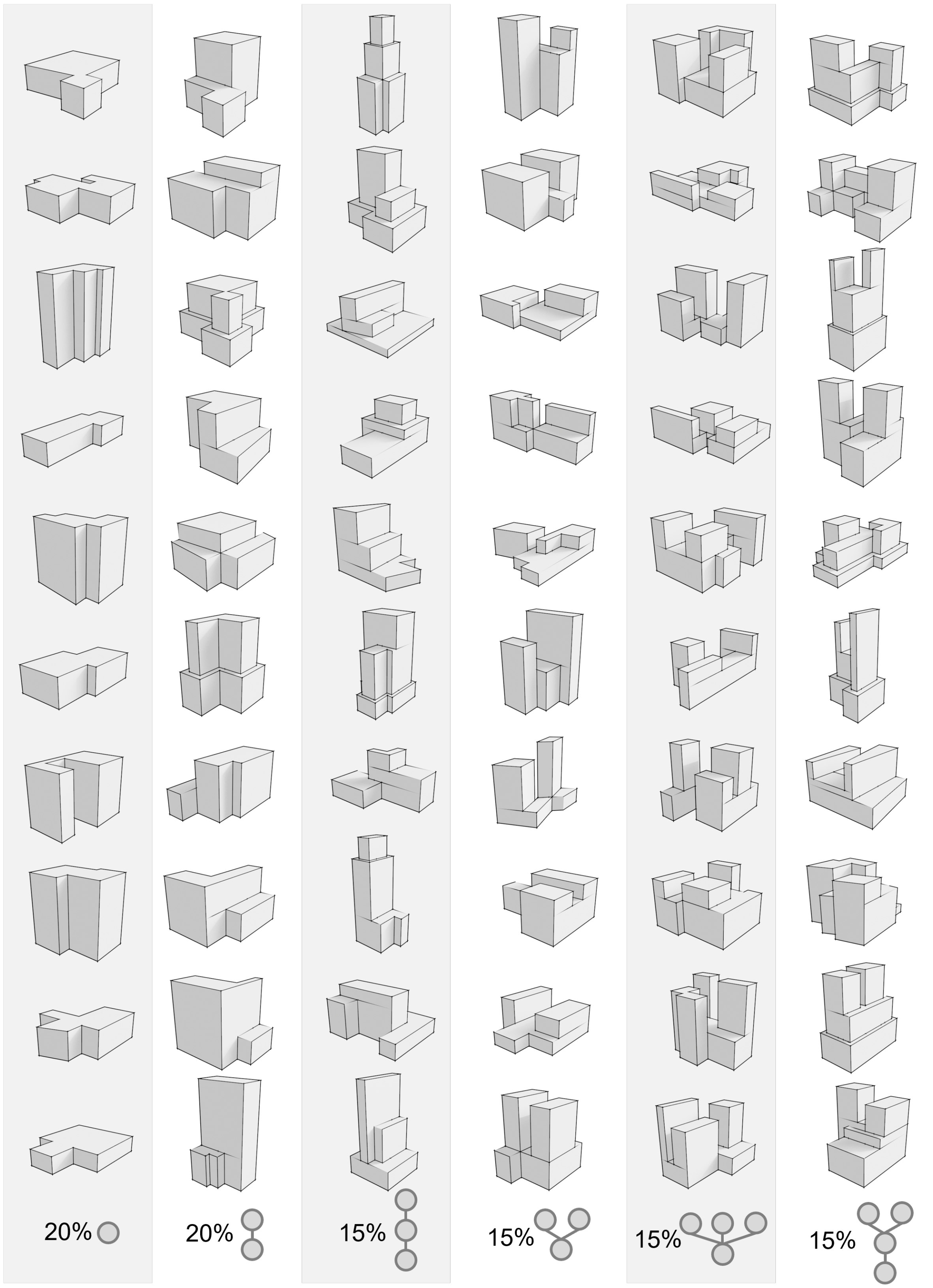

Figure 3. Synthetic training data by procedural generation. The bottom row shows six architecture tree modes and their sampling ratios.

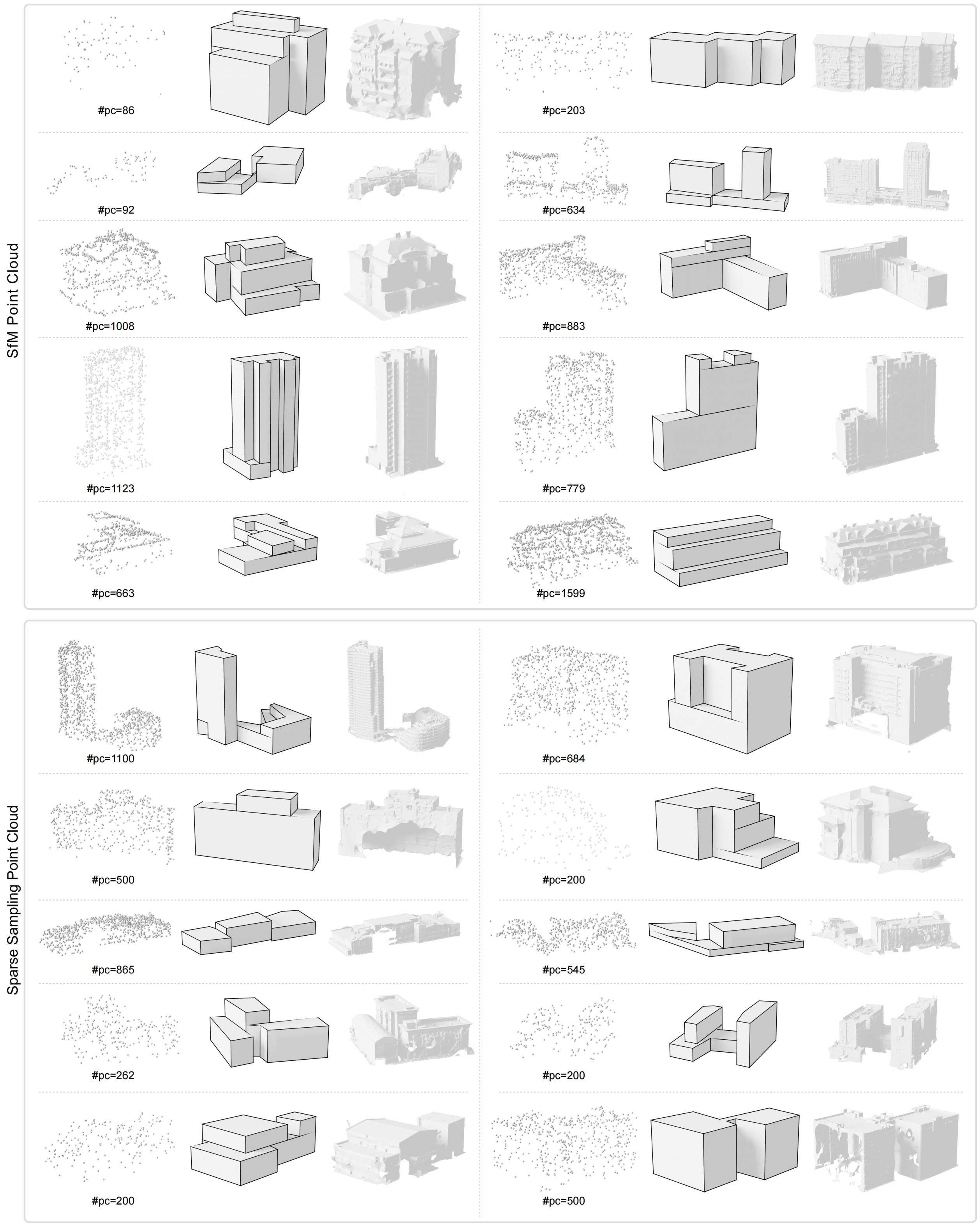

Figure 4. More results of our method applied to Structure-from-Motion (SfM) point clouds and sparse sampling point clouds. Our method can recover structured 3D abstractions from low-quality architectural point clouds that are non-uniform, incomplete, and noisy.

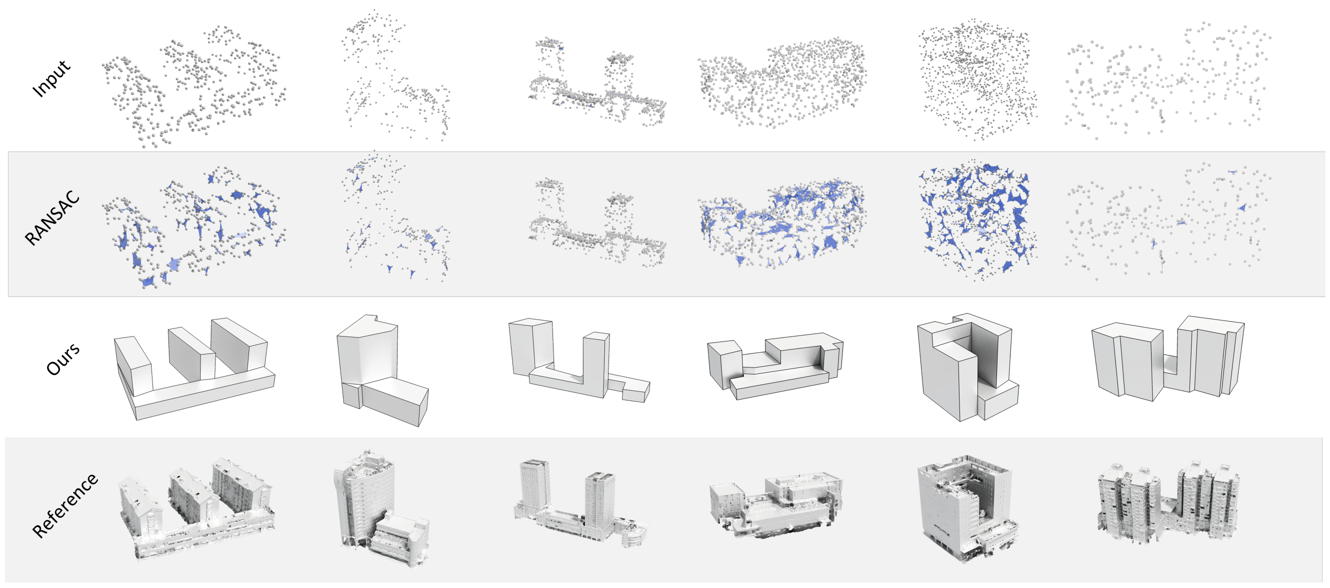

Figure 5. RANSAC plane detection results on the input from Figure 5 in the main paper. The results demonstrate that RANSAC struggles with diverse and low-quality architecture point clouds, leading to the failure of traditional methods that rely on RANSAC.

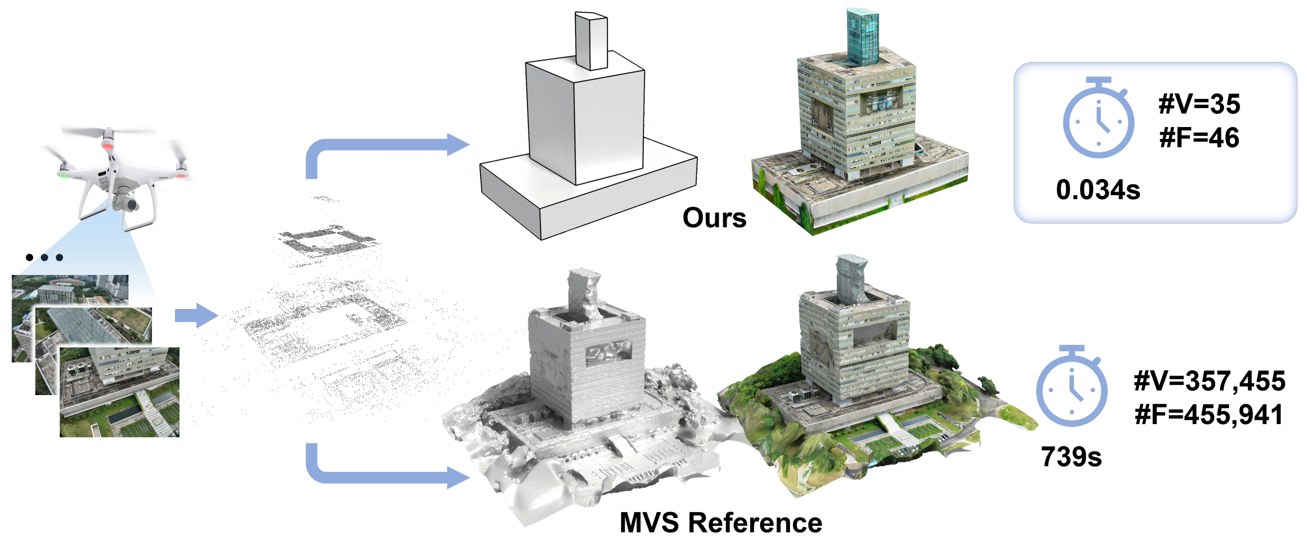

Figure 6. The result processes raw SfM point clouds from multi-view aerial images, bypassing building segmentation. Compared to traditional MVS methods, ArcPro significantly enhances inference speed while generating lightweight, textured 3D abstractions.

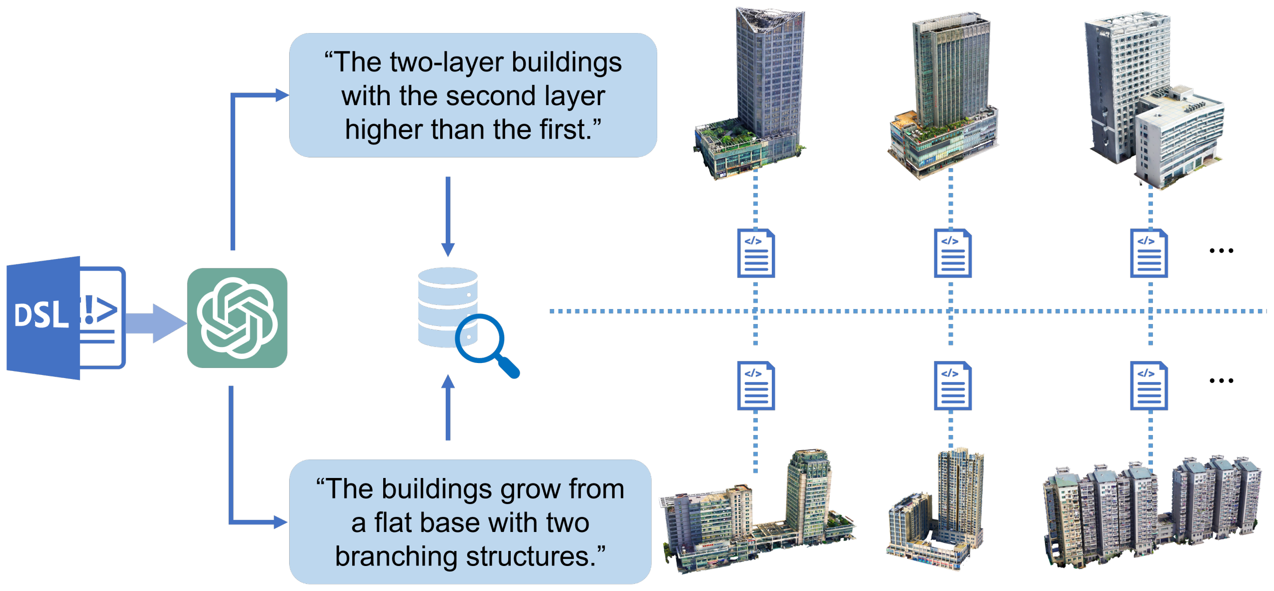
DSL
"The two-layer buildings with the second layer higher than the first."
"The buildings grow from a flat base with two branching structures."
</>

Figure 7. Architecture geometry structure analysis and natural language retrieval. Prompting ChatGPT with DSL definitions to convert geometric structure queries into Python code `IsMatching(program)` to vertify each program for retrieving matching programs.

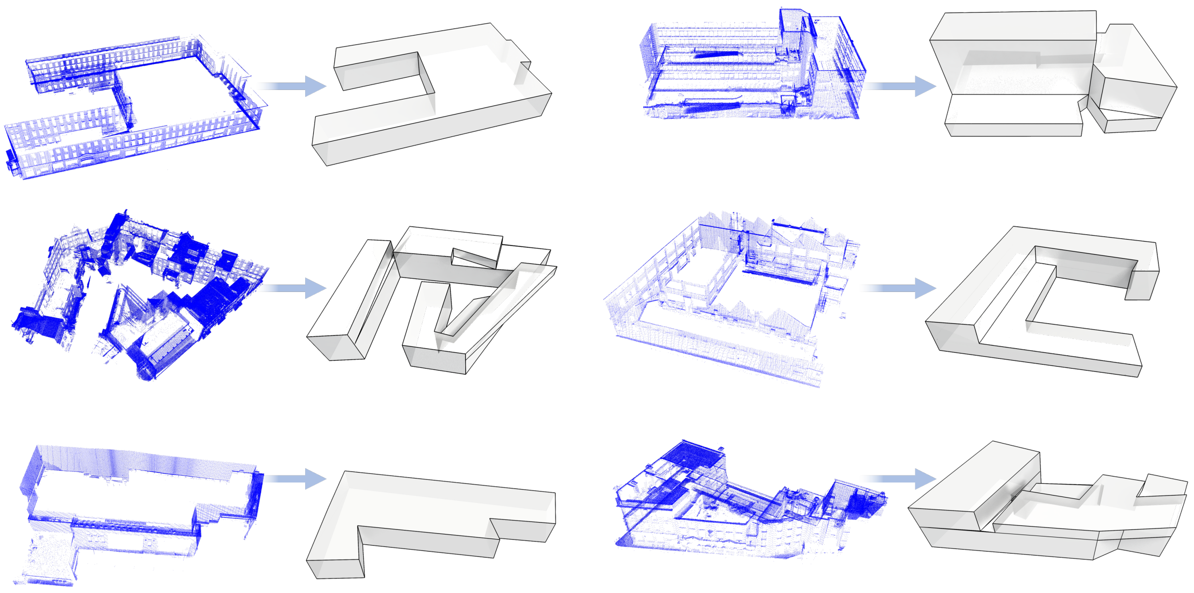

Figure 8. Results with LiDAR point clouds. Without specialized data augmentation, our method achieved reasonable performance.

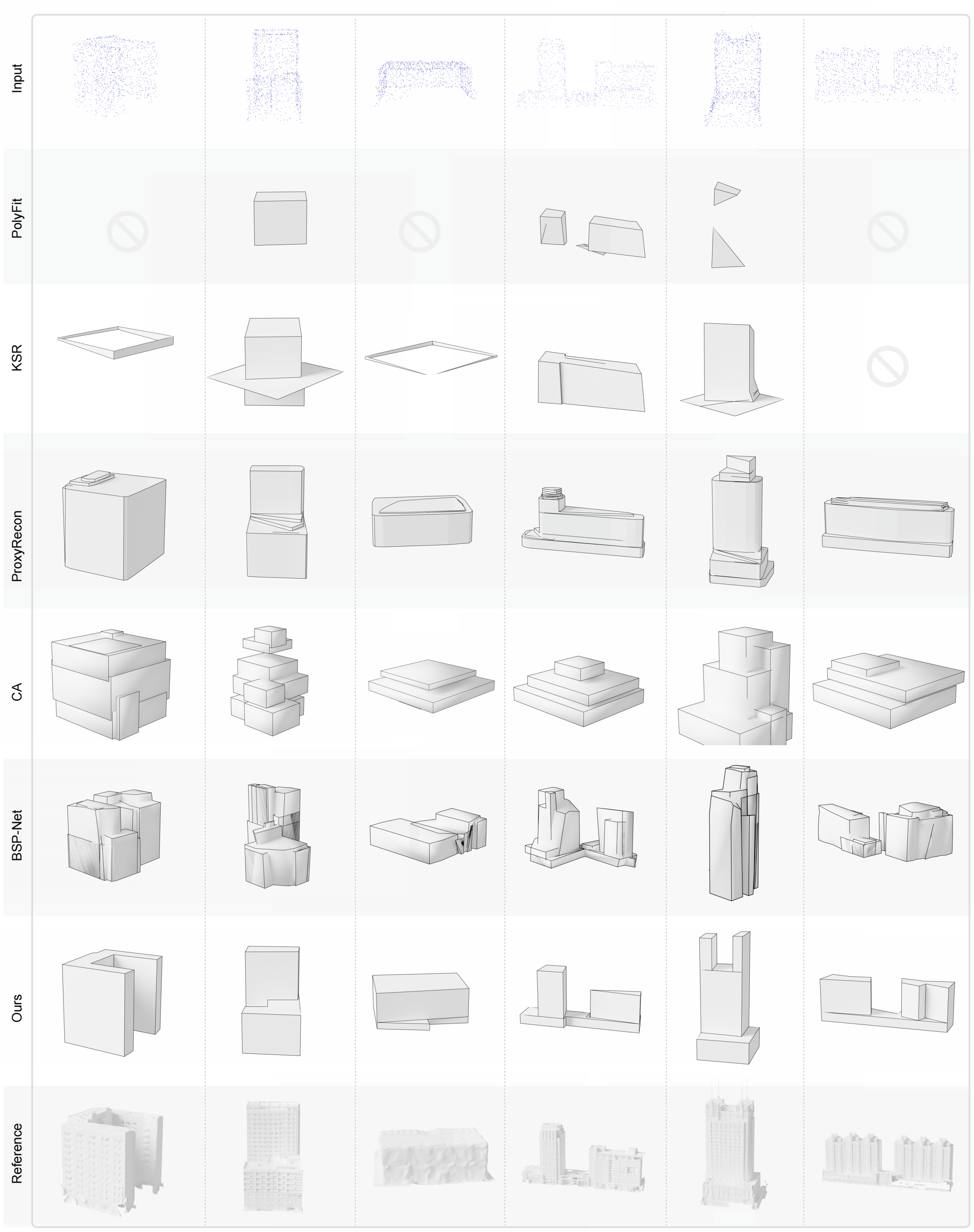

Figure 9. The user study examples comparing our method to other methods.

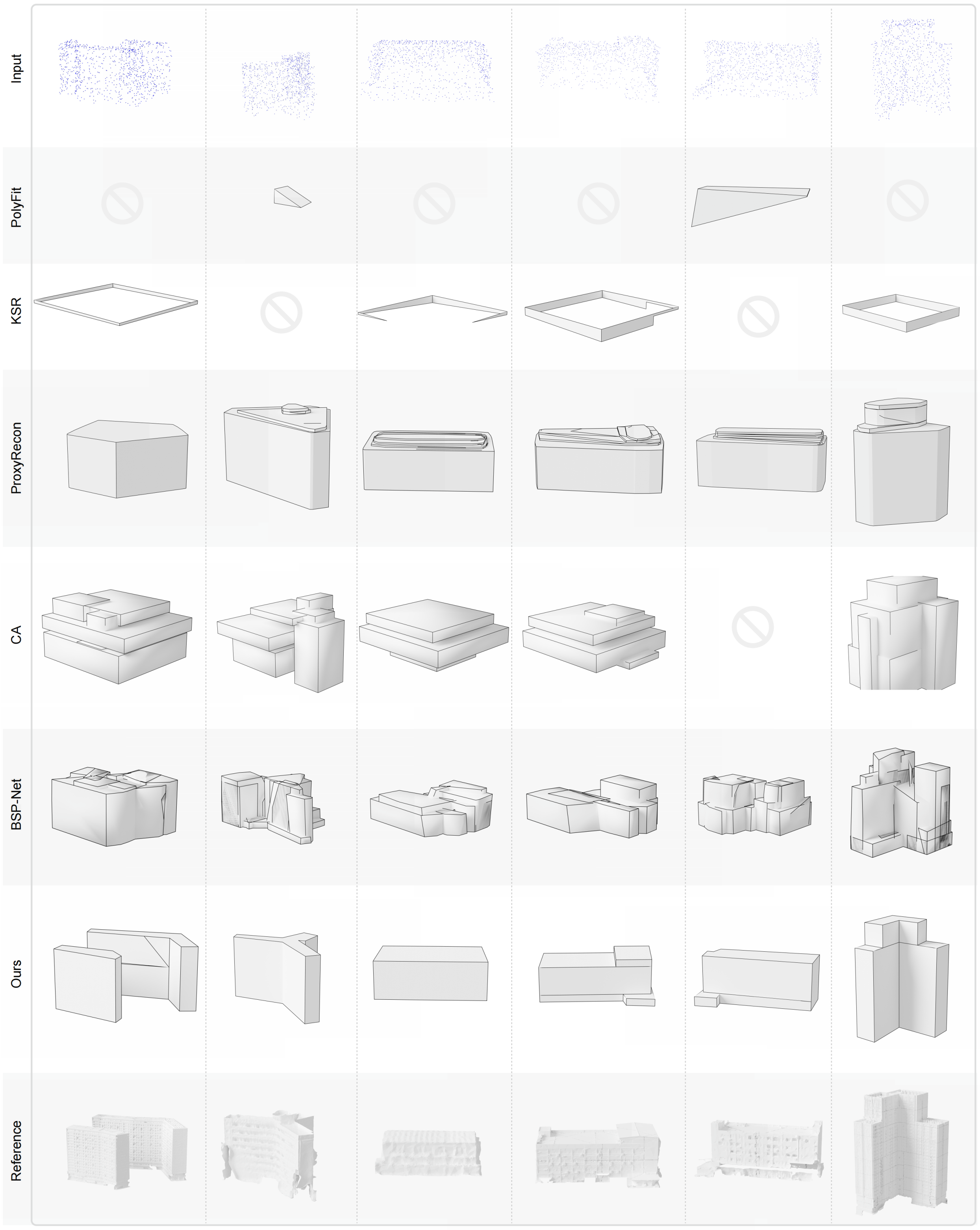

Figure 10. The user study examples comparing our method to other methods.

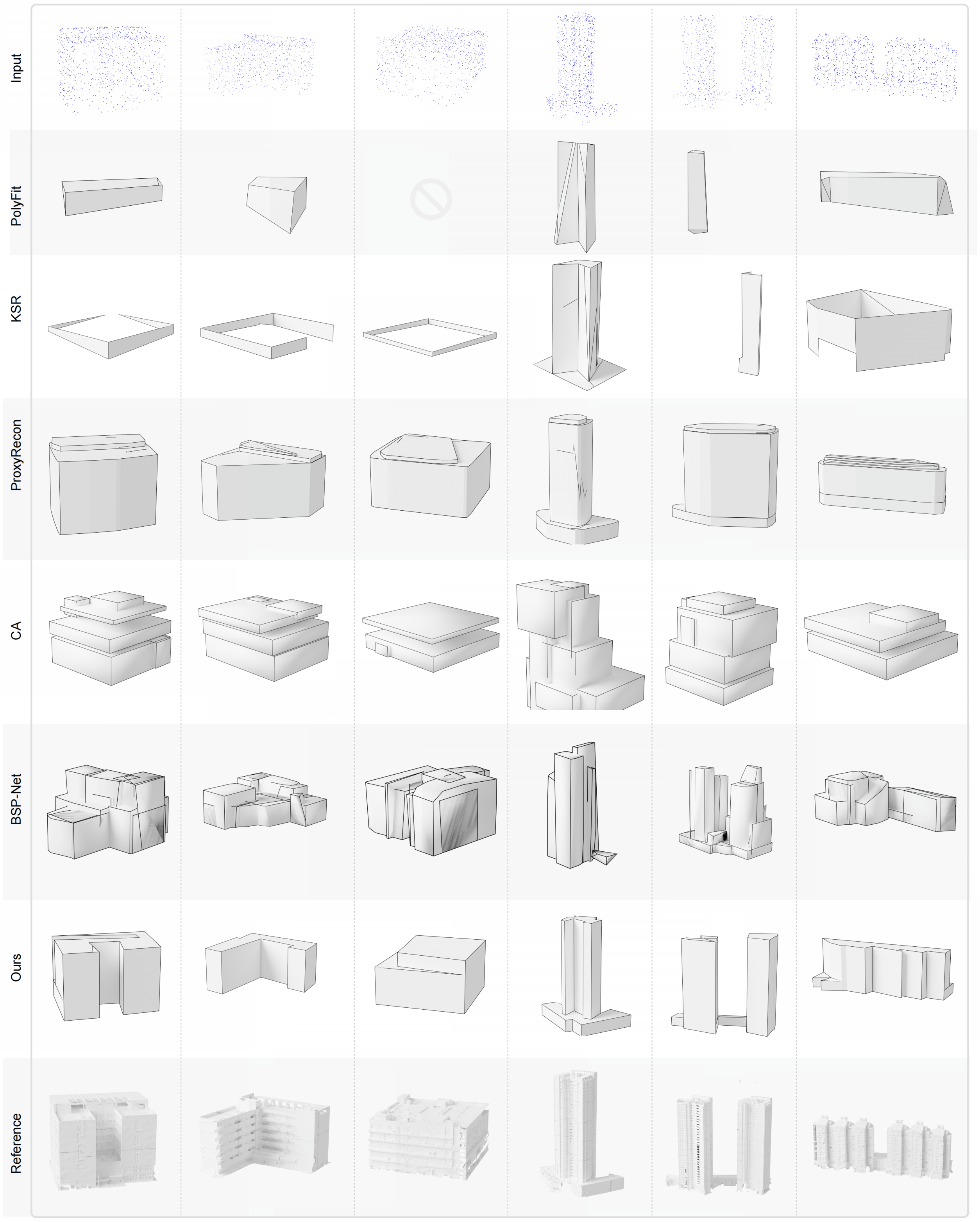

Figure 11. The user study examples comparing our method to other methods.