Supplementary Material: Concise Plane Arrangements for Low-Poly Surface and Volume Modelling

Raphael Sulzer and Florent Lafarge

Centre Inria d'Université Côte d'Azur
firstname.lastname@inria.fr

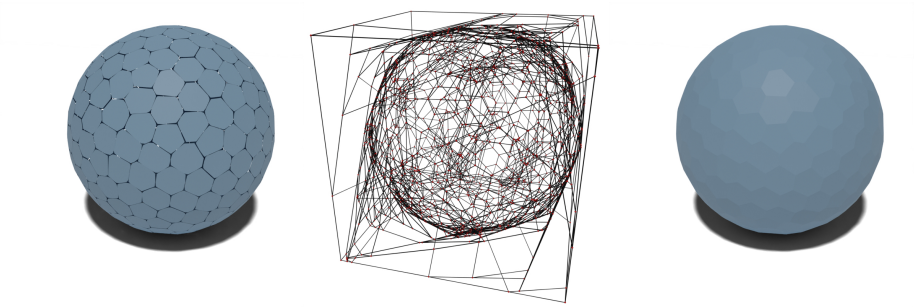
In this supplementary document, we first discuss the limitations of our method in Section 1. In Section 2, we provide implementation details about the evaluation metrics, our algorithm and the baseline methods. Finally, we provide an additional comparison with Points2Poly [3], additional applications and experimental results in Section 3.

1 Limitations

No Optimality Guarantee Our algorithm does not offer the guarantee to return the optimal solution in terms of minimal number of cells or splitting operations. Reaching optimality is very challenging here, because global optimization techniques are typically not designed to operate on the configuration space of tree data structures built sequentially via costly geometric operations. Monte Carlo Tree Search can support the exploration of a BSP-tree space, but in practice it increases processing time by at least two orders of magnitude without guaranteeing optimality or better arrangements than ours.

Dependency on Planar Shape Detection. Our modelling pipeline depends on the quality of the detected planar shapes from point clouds. The detection of planar shapes is robust to noise, outliers and point density, but only weakly to missing data. Planar shapes cannot be detected for unseen parts of objects or scenes during the acquisition, *e.g.* due to occlusion. One possible solution is to generate artificial planar shapes to complete missing data. However, such a completion strategy relies upon the insertion of strong shape priors, *e.g.* the generation of orthogonal planar shapes on borders of input points [2], which reduces the flexibility and applicability of the method. Another solution is to rely on point consolidation or neural surface reconstruction methods for preprocessing the input to recover parts of missing data.

Fully Convex Objects. The perfomance of the construction algorithm is reduced when reconstructing fully convex objects such as a sphere (see Fig. 1) or cylinder. This is because sorting condition (ii) is never valid if all input planar shapes lie on the convex hull of the object. However, very few real-world objects exhibit a fully convex shape. Moreover, the best reconstruction technique to apply to fully convex shapes is a simple computation of the convex hull.



Planar shapes | Plane arrangement | Reconstructed surface

Fig. 1: Reconstruction of a Sphere. Our reconstruction of the sphere as a single convex cell from 280 input planar shapes.

Lack of Semantics. Our method is purely geometric and does not use semantic information. Models from the same object category are not guaranteed to be reconstructed with the same level of abstraction. However, our method for producing simplified volume meshes often leads to reconstructions with one convex cell for each semantic instance (see *e.g.* parts of the chairs in the teaser, or the pillows in Fig. 4).

2 Implementation Details

2.1 Evaluation Metrics

To compute Chamfer distance (CD), Hausdorff distance (HD) and normal consistency (NC) between a ground-truth surface mesh $\mathcal{M}^g$ and a reconstructed surface mesh $\mathcal{M}^r$ we sample a set of points S^r and S^g on the facets of the ground-truth mesh and the reconstructed mesh, respectively, with $|S^r| = |S^g| = 100,000$.

Chamfer Distance. We approximate the Chamfer distance between $\mathcal{M}^g$ and $\mathcal{M}^r$ as follows:

$$\mathrm{CD}(\mathcal{M}^g, \mathcal{M}^r) = \frac{1}{2|S^g|} \sum_{x \in S^g} \min_{y \in S^r} ||x - y||_2 + \frac{1}{2|S^r|} \sum_{y \in S^r} \min_{x \in S^g} ||y - x||_2 \ . \quad (1)$$

Hausdorff Distance. We approximate the Hausdorff distance between $\mathcal{M}^g$ and $\mathcal{M}^r$ as follows:

$$\mathrm{HD}(\mathcal{M}^g, \mathcal{M}^r) = \max\{ \max_{x \in S^g} \min_{y \in S^r} ||x - y||_2, \max_{y \in S^r} \min_{x \in S^g} ||y - x||_2 \ \}. \quad (2)$$

Normal Consistency. Let $n(x)$ be the unit normal of a point x on a mesh, and $\langle\cdot,\cdot\rangle$ the Euclidean scalar product in $\mathbb{R}^3$. We estimated the normal consistency as follows:

$$\begin{aligned}\mathrm{NC}(\mathcal{M}^g,\mathcal{M}^r) = \frac{1}{2|S^g|}\sum_{x\in S^g}\left\langle n(x), n\left(\arg\min_{y\in S^r}||x-y||_2\right)\right\rangle \\ +\frac{1}{2|S^r|}\sum_{y\in S^r}\left\langle n(y), n\left(\arg\min_{x\in S^g}||y-x||_2\right)\right\rangle\ .\end{aligned} \tag{3}$$

2.2 Our Algorithm

Our algorithm for constructing a concise plane arrangement is shown in Alg. 1. The algorithm is parameter free and only depends on input points and planes.

Algorithm 1 Concise Plane Arrangement Construction

Input: P, I: Planes and Inlier Point Sets
Output: C, F, G, T: Cells, Facets, Adjacency Graph and Binary Tree

```
T ← InitTree(0)                                   ▷ Initialise tree with node 0
G ← InitGraph(0)                                  ▷ Initialise graph with node 0
C_0 ← Bbox(I)                                     ▷ Define cell C_0 as bounding box of all points
I_0 ← I                                           ▷ Define inlier set I_0 containing all inlier point sets
P_0 ← P                                           ▷ Define plane set P_0 containing all planes
for all TreeNodes t in DFS(T) do                  ▷ Traverse T with depth-first search
    if I_t is not empty then
        p ← GetBestPlane(I_t,P_t)                                          ▷ Alg. 2
        C_u,C_v,F_t ← SplitCell(C_t, p)                                    ▷ Polyhedron-plane intersection
        I_u,I_v,P_u,P_v ← SplitInliers(I_t, P_t, p)                        ▷ Orientation test
        UpdateTree(T,t,u,v)                                                ▷ Add children u,v with parent t
        UpdateGraph(G,t,u,v)                                               ▷ Split t into u, v & update adjancencies
```

The key steps of our algorithm are (i) the dynamic ordering scheme of split planes using our *GetBestPlane*-function (Alg. 2) and (ii) the hirarchical clustering of inlier points and planes using a binary tree T. Note that, the *GetBestPlane*-function is only evaluated on the subset of planes P_t and corresponding inlier point sets I_t associated with the current tree node t. While the *GetBestPlane*-function evaluation may slow down the construction algorithm in the beginning, it drastically reduces the total runtime of the algorithm. We remind the reader that the time complexity of a plane arrangement construction is $\mathcal{O}(n^3)$, with n being the number of input planes [6]. The main bottleneck of most plane arrangement algorithms is the polyhedron-plane intersection which requires exact geometric constructions. The *GetBestPlane*-function allows us to equally spread the number of planes in each subtree, *i.e.* to minimise n in each subtree, and thereby reduce the number of necessary intersection computations drastically.

Algorithm 2 Get Best Split Plane

Input: P, I: Planes and Inlier Point Sets
Output: p: Split Plane

1: **for all** Planes p **in** P **do**
2: L_p,$R_p \leftarrow \emptyset$ ▷ Initialise point sets that lie fully left and right of p
3: **for all** PointSets I_i **in** $I \setminus I_p$ **do**
4: **if** All points I_i lie left of p **then** ▷ Orientation test
5: $L_p \leftarrow L_p \cup I_i$
6: **else if** All points I_i lie right of p **then** ▷ Orientation test
7: $R_p \leftarrow R_p \cup I_i$
8: **if** L_p or R_p is empty **then**
9: **return** p ▷ Condition (i) in Section 3.2 (main paper)
10: **return** $\arg\max_p(|L_p||R_p|)$ ▷ Condition (ii) in Section 3.2 (main paper)

We use the SageMath library with exact rational numbers for the polyhedron-plane intersection and floating point precision for the orientation tests. The orientation tests, *e.g.* for computing the sets L_p and R_p are fully parallelisable and can be done efficiently on GPU using PyTorch tensor multiplications.

Our implementation will be made available as open source upon publication of the paper.

2.3 Baseline Details

Adaptive Arrangement and Points2Poly. We implement an adaptive plane arrangement based on Points2Poly (P2P) [3]. P2P is a pipeline for concise building model reconstruction. P2P uses axis-alligned bounding boxes (AABB) of input planar primitives to construct a plane arrangement. Alg. 3 shows the core algorithm of P2P and our *Adaptive* baseline. P2Ps arrangement and our Adaptive baseline only differ in their *Sort*-function (Line 2). P2P sorts input planes first by their verticality, *i.e.* vertical planes are inserted first in the arrangement. The authors identify vertical planes as most important to recover the general shape of buildings. Because we mainly conduct experiments on general 3D models, we only use the second sorting criteria of P2P for the Adaptive baseline. That is, we sort planes by their number of inlier points, with planes with more inliers first. The rest of the arrangement computation stays unchanged, *i.e.* is equal for P2P and the Adaptive baseline. However, we find that the occupancy classification of P2P often assigns the wrong label to polyhedral cells of the arrangement and the normal-based energy formulation of KSR [1] labels cells with fewer errors. We thus use the normal-based energy for the Adaptive baseline and our method. Further, we use our own surface extraction for the Adaptive baseline because P2P only outputs a set of unconnected polygon facets, instead of a closed surface mesh[1].

[1] https://github.com/chenzhaiyu/abspy/issues/18

Algorithm 3 Adaptive Arrangement Construction

Input: P, B: Planes and AABBs of Inlier Point Sets
Output: C, F, G: Cells, Facets and Adjacency Graph

```
C_0 ← Bbox(B)                          ▷ Define cell C_0 as bounding box of all points
Sort(P,B,order)                        ▷ Sort planes and AABBs according to order
G ← InitGraph(0)                       ▷ Initialise graph with node 0
for all AABBs b in B do
    for all GraphNodes g in G do
        if IntersectionTest(C_g,b) then          ▷ Polyhedron-AABB inter. test
            if IntersectionTest(C_g,p) then      ▷ Polyhedron-plane inter. test
                C_u,C_v,F_t ← SplitCell(C_t, p)  ▷ Polyhedron-plane intersection
                UpdateGraph(G,t,u,v)             ▷ Split t into u, v & update adjancencies
```

BSP-Net. For the reconstruction experiment in the BSP-Net paper ([5, Table 2]) the authors used several networks each trained on a single category of ShapeNet ([5, Section 4.2]). Due to the long training times of BSP-Net we cannot train multiple single class networks. We therefore use the auto-encoder variant of the network trained on all 13 categories of ShapeNet available on the authors' GitHub[2]. Consequently, our results differ from the ones in [5]. This issue has been previously discussed on the BSP-Net GitHub[3]. However, our method compares favourable to BSP-Net, even when comparing to the numbers presented in Table 2 in [5], *e.g.* with a Chamfer distance of 0.447 for BSP-Net and 0.385 for Ours.

Further, note that we use the evaluation pipeline of BSP-Net[4] which computes the *squared* Chamfer distance unlike our implementation (cf Eq. 1). Consequently, the Chamfer values in Figure 7 (main text) are not comparable with the ones in any of our other tables.

3 Additional Experiments

3.1 Comparison with Points2Poly

Experimental Setup. P2P is a pipeline for concise building model reconstruction. P2P uses axis-alligned bounding boxes (AABB) of input planar primitives to construct a plane arrangement. P2P also introduces a strategy to classify the cells of the arrangement with binary occupancies with the combination of a neural surface reconstruction network [7] and a Markov Random Field. We compare our method, and the Adaptive baseline, to P2P on their Helsinki city dataset. The test set comprises 45 buildings which are synthetically scanned using Blensor to mimic a realistic LiDAR acquisiton with noise, and missing

[2] https://github.com/czq142857/BSP-NET-original#datasets-and-pre-trained-weights
[3] https://github.com/czq142857/BSP-NET-original/issues/11
[4] https://github.com/czq142857/BSP-NET-original/tree/master/evaluation

Table 1: Quantitative Comparison with Points2Poly (P2P) on their Helsinki city dataset [3]. We show the number of vertices $|V_S|$ and polygonal facets $|F_S|$ of building models. We also show the one-sided Hausdorff distance ($HD^{r \to g}$) measured from reconstructed to ground truth building models, the inverse distance ($HD^{g \to r}$), and the maximum of the two distances (HD). These values were computed using the evaluation pipeline provided by the authors of P2P. All metrics are computed as the mean value of the 45 building models. Note, that some of the ground truth models contain interior structures which cannot be reconstructed due to missing input data. This corrupts the one-sided Hausdorff distance measured from ground truth models to reconstructions ($HD^{g \to r}$). In the last column, we show the average reconstruction time per building.

	Complexity		*Accuracy*			*Performance*
	$\|V_S\|$	$\|F_S\|$	$HD^{r \to g}$ ($\times 10^2$)	$HD^{g \to r}$ ($\times 10^2$)	HD ($\times 10^2$)	Time (s)
P2P [3]	419	103	4.71	**7.51**	8.64	4.14
Adaptive	148	131	4.27	7.62	8.62	3.32
Ours	**57.6**	**41.2**	**3.30**	7.65	**7.79**	**2.60**

data due to occlusion. We use the code, pretrained weights, data and evaluation pipeline provided by the authors[5]. We find that some of the provided ground truth models include interior structures. During the scanning procedure no rays can reach facets enclosed within the model. Consequently, the interior structures are not part of the scanned point clouds. Nonetheless, the evaluation pipeline computes the one-sided Hausdorff distance from the ground truth models to the reconstruction ($HD^{g \to r}$). We think that the presence of the interior structures corrupts this distance measure, and consequently also the two-sided distance HD. The one-sided distance from reconstruction to ground truth mesh ($HD^{r \to g}$) is not affected by the interior structures. We still provide all three measurements.

Results. We show quantitative and qualitative results of our comparison with P2P in Tab. 1 and Fig. 2, respectively. Our method produces more concise building models with around 7× less vertices and less than half the number of facets. Our models are also more accurate and our pipeline is faster.

3.2 Sensitivity to Defect-Laden Input

Our modelling pipeline offers a good robustness to noise, outliers and low point density in the input data. Such defects weakly affect the first step, *i.e.* the detection of planar shapes. In particular, the fitting tolerance allows us to absorb a significant amount of noise. The k-nearest neighbor graph of input points commonly used for planar shape detection is robust to varying point density. Finally, plane fitting mechanisms typically exploit information on both position and orientation of input points for selecting plane inliers points. This makes the

[5] https://github.com/chenzhaiyu/points2poly

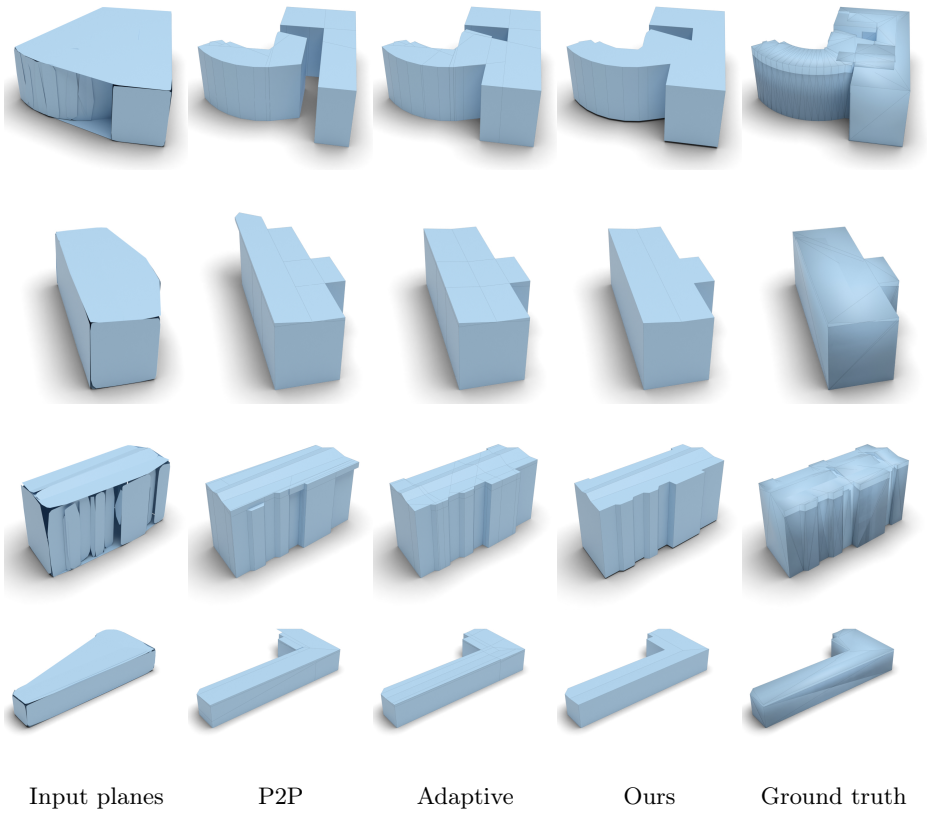
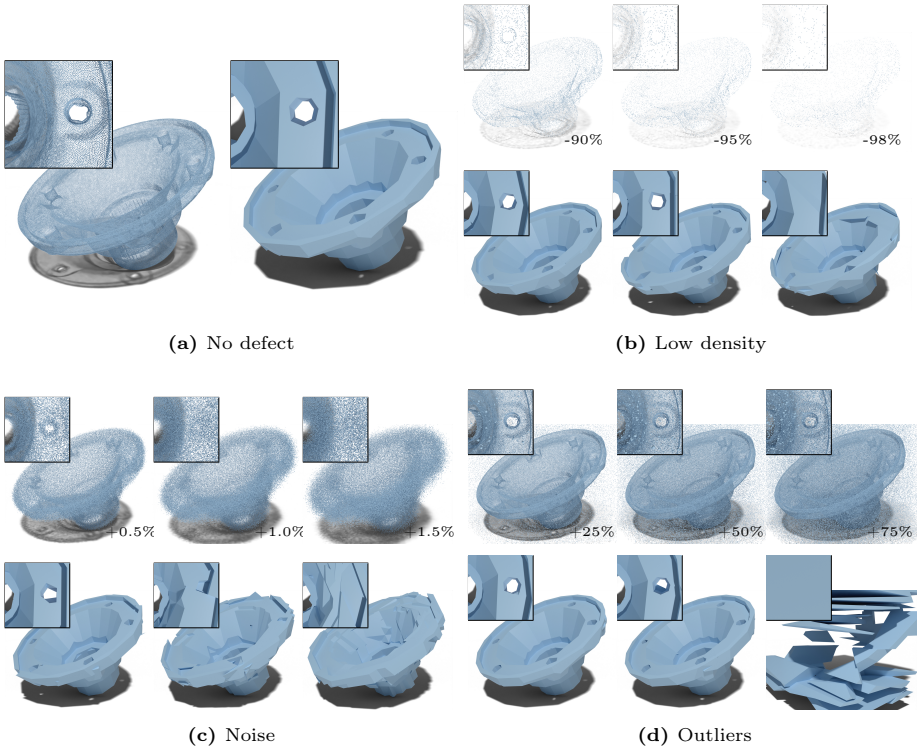


Fig. 2: Comparison with Points2Poly (P2P) [3] on their Helsinki City dataset. We compare the adaptive baseline and our method to P2P. Our building models are more concise and more accurate than the two baselines.

method robust to outliers. Fig. 3 shows how our reconstruction pipeline benefits from the robustness of planar shape detection against different input defects.



(a) No defect **(b)** Low density

(c) Noise **(d)** Outliers

Fig. 3: Sensitivity to Defect-Laden Input. Our method exhibits high robustness to different input defects. In (a) we show our reconstruction of the *Carter* model from 200k input points. In (b) we remove 90%, 95% and 98% of input points chosen at random. Our reconstruction pipeline is highly robust to low density input. In (c) we add Gaussian noise with a standard deviation of 0.5%, 1.0% and 1.5% of the bounding box diagonal. Our reconstructions start to degrate around the 1% noise level. In (d) we add 25%, 50% and 75% of outliers randomly sampled within the bounding box of the object. Our method produces no remarkable defects for the lower outlier levels. For 75% outliers our method cannot reconstruct a useable surface because the occupancy for surface extraction cannot be accurately predicted anymore.

3.3 Additional Applications

Shape Editing. Figure 4 shows an example of the use of our algorithm for simple shape editing. The simplification of an input shape into a set of convexes

often associates each convex with a meaningful part of the shape, which can then be transformed or entirely removed.

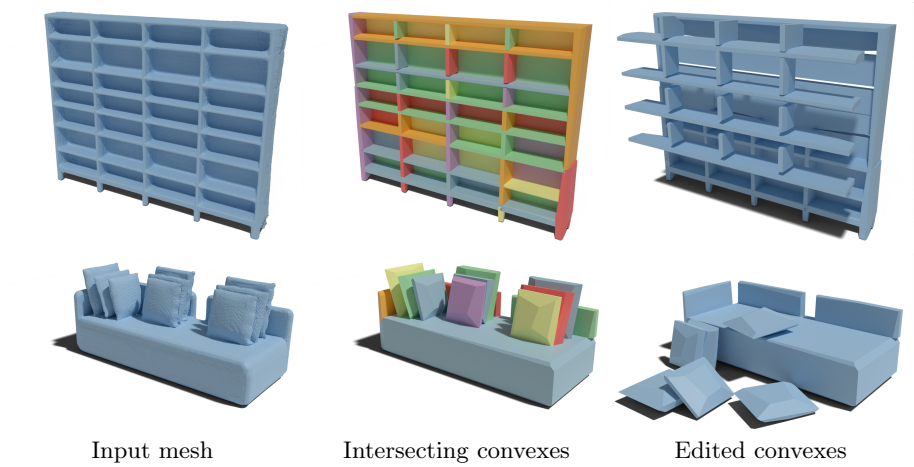


Fig. 4: Shape Editing. Our simplified volume reconstruction produces a set of intersecting convexes that can easily be manipulated and edited. For instance, we can remove the pillows from the couch or disassemble the shelf into numerous vertical and horizontal components.

Simplification at Various Levels of Detail. Figure 5 shows how our aggregation of convex cells can simplify a complex urban scene, from detailed geometry of facades and rooftops, to a single convex cell enclosing each building block. The simplification process does not rely upon a deep analysis of the geometry and semantics of the scene, but simply exploits the information contained in our graph representation. Constructing the plane arrangement from 17k input planar shapes detected from 10M points takes around 30 minutes.

We then initialise and update a priority queue for merging convex cells by computing the left-hand side of Eq. 1 in the main paper for all neighbors in the adjacency graph G. Remember that Eq. 1 measures the volume deviation between the sum of the volumes of two cells, and the volume of the convex hull of their union. We can now iteratively merge the two neighboring convex cells with the smallest volume deviation in the priority queue until reaching the desired level of detail, *i.e.* the desired number of convex cells. Cell aggregation, including initialisation and update of the priority queue takes around 30 seconds only.

3.4 Additional Results from Tanks And Temples

Fig. 6 provides additional results on three MVS point clouds and a laser scan from the Tanks and Temples [8] dataset compared to KSR [1]. Starting from

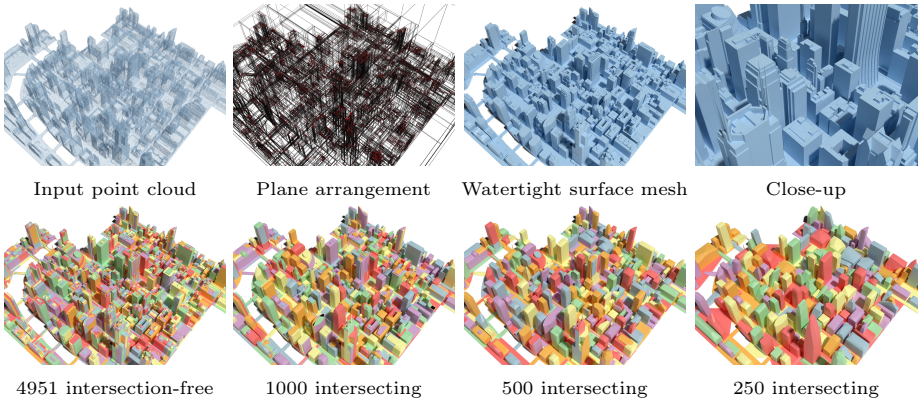


Fig. 5: Simplification at Various Levels of Detail. Our scalable pipeline is designed to finely reconstruct objects and scenes, even at large scale. As shown in the close-up, the reconstructed buildings exhibit many geometric details on facades and rooftops with even the presence of thin structures such as antennas. At the same time, our surface reconstruction only uses very few faces. The 4,951 non-intersecting convexes that finely describe this urban landscape can be simplified by aggregating the convexes and authorizing their overlap. A simplification with 1,000 convexes typically removes facade and roof details. At 500 convexes, only complex buildings are still described by more than one convex cell. Reducing to 250 convexes allows us to represent each building block or skyscraper by a single convex cell.

the same planar shapes, the meshes produced by KSR include significantly less details. This may be due to a high regularisation strength used for their surface extraction, in order to filter out noise from an overly complex partition.

3.5 Additional Results from Thingi10k

We show additional results of our comparison with Robust Low Poly Meshing (RLPM) [4] on their Thingi10k subset in Fig. 7 and Fig. 8.

References

1. Bauchet, J.P., Lafarge, F.: Kinetic shape reconstruction. ACM Transactions on Graphics (TOG) **39**(5) (2020)
2. Chauve, A.L., Labatut, P., Pons, J.P.: Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
3. Chen, Z., Ledoux, H., Khademi, S., Nan, L.: Reconstructing compact building models from point clouds using deep implicit fields. ISPRS Journal of Photogrammetry and Remote Sensing **194** (2022)
4. Chen, Z., Pan, Z., Wu, K., Vouga, E., Gao, X.: Robust low-poly meshing for general 3d models. ACM Transactions on Graphics (TOG) **42**(4) (2023)

5. Chen, Z., Tagliasacchi, A., Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
6. Edelsbrunner, H., O'Rourke, J., Seidel, R.: Constructing arrangements of lines and hyperplanes with applications. SIAM Journal on Computing **15**(2) (1986)
7. Erler, P., Ohrhallinger, S., Mitra, N., Wimmer, M.: Points2Surf: Learning implicit surfaces from point clouds. In: (ECCV). pp. 108–124 (2020)
8. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and Temples. ACM Transactions on Graphics (TOG) (2017)

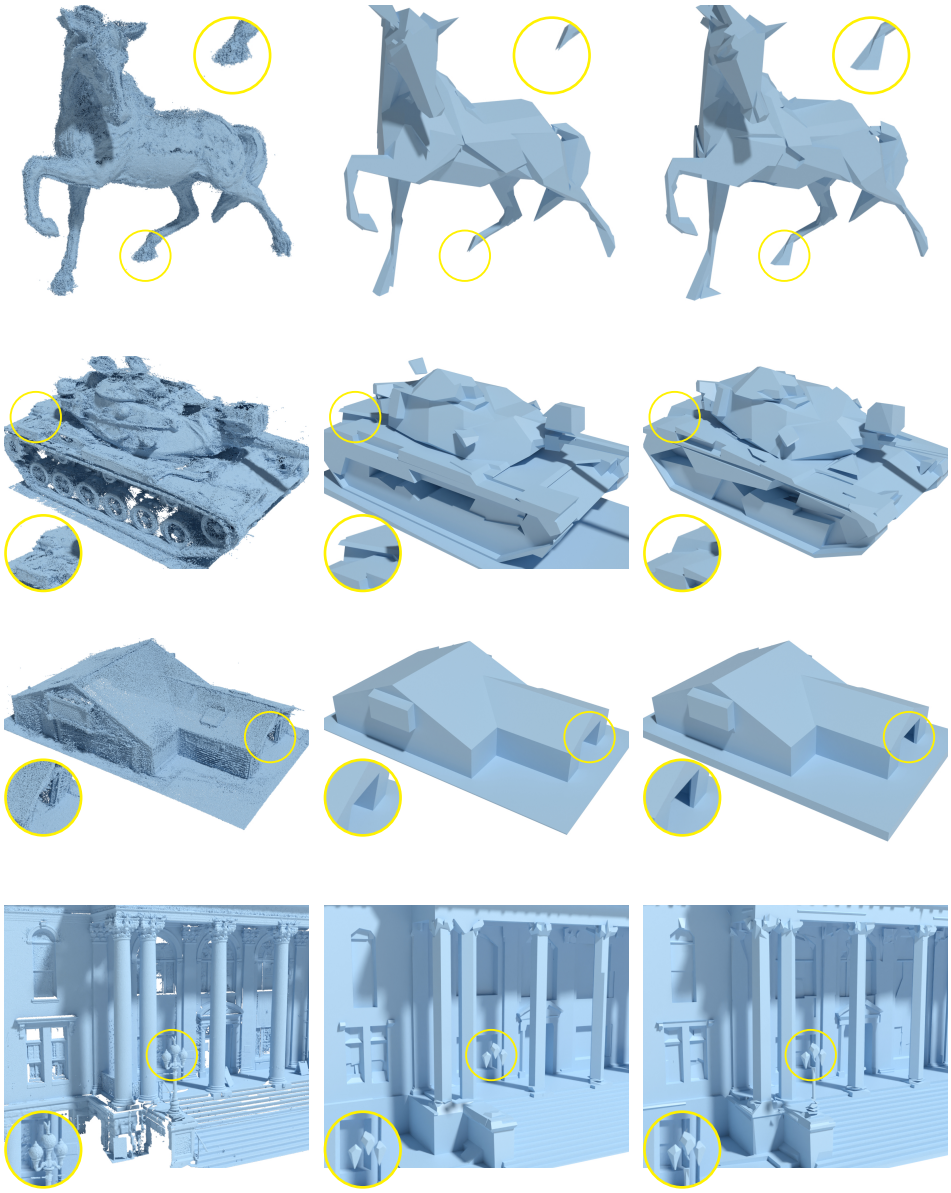


Fig. 6: Comparison with KSR [1] on the Tanks And Temples [8]. We compare to three MVS (top rows) and one LiDAR (bottom) reconstruction from the KSR42 / Tanks and Temples dataset. Our method produces more accurate yet lightweight reconstructions.

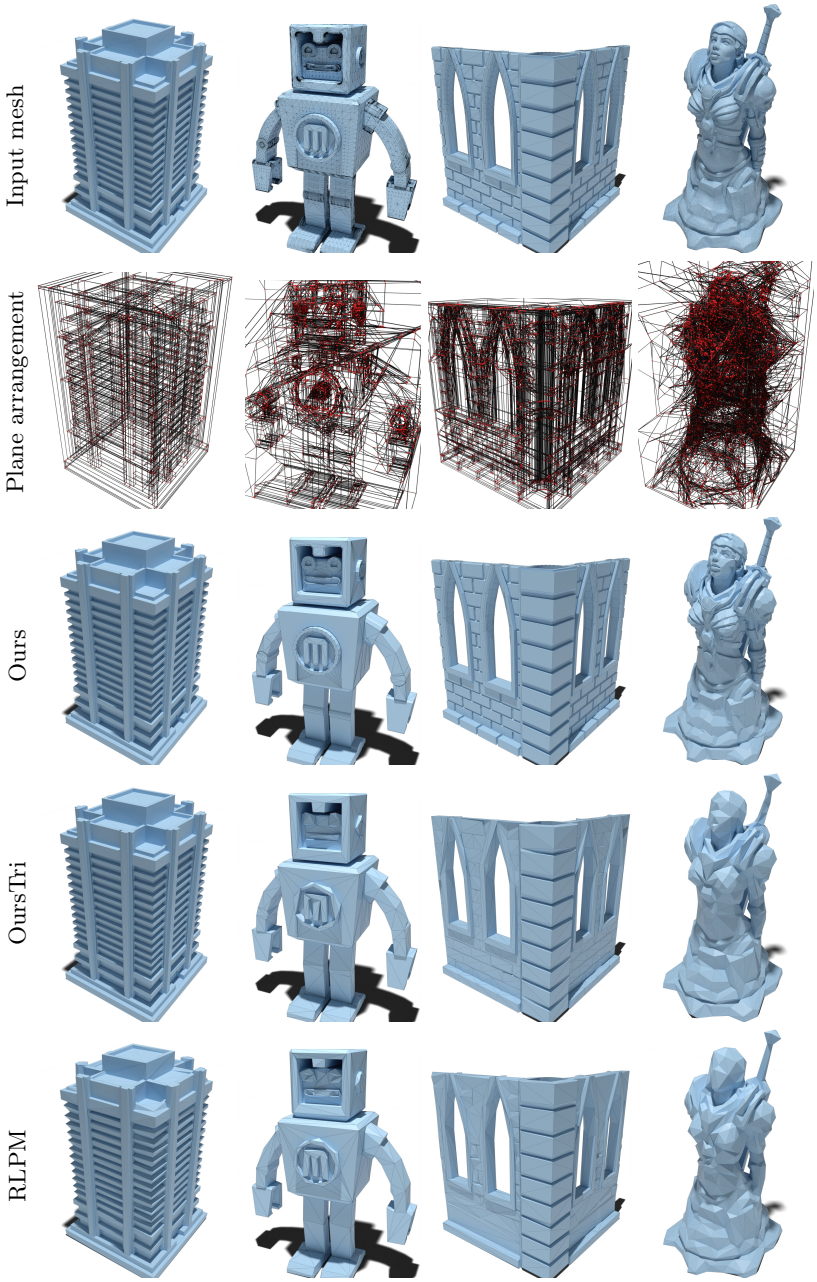


Fig. 7: Additional Results from the Thingi10k Subset of Robust Low Poly Meshing (RLPM) [4]. We show the input mesh (first row) which we sample and extract planes to construct our concise plane arrangement (second row). Our extracted polygon mesh (third row) approximates the input with up to 40× less facets while still representing almost all details. While our method is tailored to produce polygon meshes we can triangulate the facets and apply edge collapse with QEM (fourth row). OursTri even compares favourable to the specialised triangulation based method RLPM (last row). Note that, OursTri and RLPM have the same number of triangles.

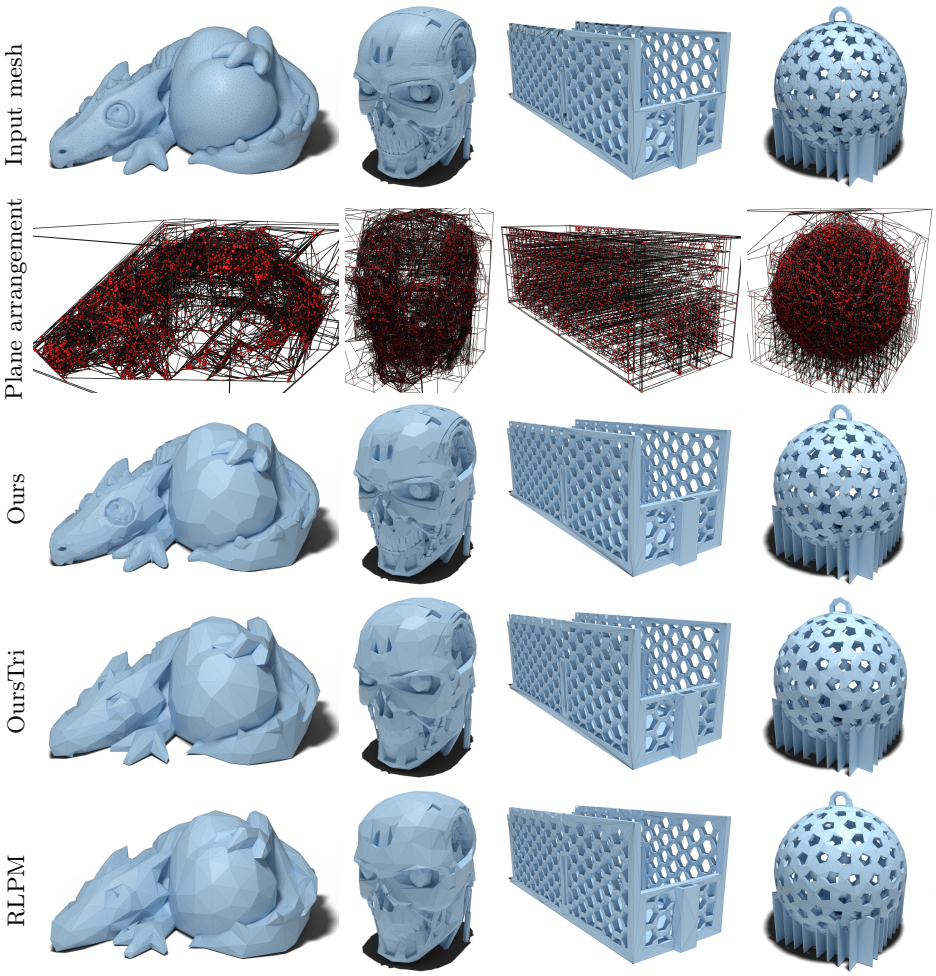


Fig. 8: Additional Results from the Thingi10k Subset of Robust Low Poly Meshing (RLPM) [4]. We show the input mesh (first row) which we sample and extract planes to construct our concise plane arrangement (second row). Our extracted polygon mesh (third row) approximates the input with large planar facets on planar parts and smaller facets to represent details. We can also triangulate the facets of our polygon mesh and apply edge collapse with QEM (fourth row). OursTri compares favourable to the specialised triangulation based method RLPM (last row). Note that, OursTri and RLPM have the same number of triangles.