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

Abstract. Plane arrangements are a useful tool for surface and volume modelling. However, their main drawback is poor scalability. We introduce two key novelties that enable the construction of plane arrangements for complex objects and entire scenes: an ordering scheme for the plane insertion and the direct use of input points during arrangement construction. Both ingredients reduce the number of unwanted splits, resulting in improved scalability of the construction mechanism by up to two orders of magnitude compared to existing algorithms. We further introduce a remeshing and simplification technique that allows us to extract low-polygon surface meshes and lightweight convex decompositions of volumes from the arrangement. We show that our approach leads to state-of-the-art results for the aforementioned tasks by comparing it to learning-based and traditional approaches on various different datasets.

Keywords: plane arrangement · low-poly · reconstruction

1 Introduction

Explicit 3D mesh representations such as tetrahedralisations or triangle surface meshes allow for a good approximation of freeform geometry using atomic parts of surface and volume. However, for storing, analysing or manipulating 3D data, these dense representations introduce a computational burden by describing even simple shapes with many redundant elements. Lightweight polygon surface and volume meshes, composed of a low number of polygons or convex polyhedra can facilitate the handling of the underlying 3D data. However, the reconstruction of such low-poly meshes from raw data measurements is still a challenging task. In this paper, we address this task by presenting a scalable plane arrangement method for low-poly surface and volume modelling from point clouds. Plane arrangement methods [3,5,22] use planar shapes detected from data measurements and arrange them to form a polyhedral decomposition, *i.e.* a partition of a 3D domain by polyhedra. Subsequently, a concise mesh can be extracted from the partition by selecting a subset of the polygonal facets or polyhedral cells. This strategy is particularly interesting as (i) it can be used for both surface and volume modelling, (ii) it offers a good robustness to imperfect data and (iii) it comes with desirable geometric guarantees such as watertightness and orientability of

Fig. 1: Our representation of the Meeting Room [18] as a watertight and intersection-free surface mesh with 90k polygonal facets (center) or as a simplified volume mesh with 2500 convex polyhedra (right). Our method inputs 40k planes (middle left) detected from a LiDAR scan of the scene (top left). For comparison, a Screened Poisson [17] reconstruction consists of over 6M triangles (bottom left).

the surface and convexity of the polyhedral cells. The main shortcoming of plane arrangement methods is their poor scalability.

The *exhaustive* plane arrangement iteratively slices the 3D domain with detected planes which leads to a time complexity of $\mathcal{O}(n^3)$, with n being the number of input planes [12]. The *adaptive* arrangement uses convex planar polygons [21] or axis-aligned bounding boxes [7] instead of infinite planes, which drastically reduces the number of intersection computations and the complexity of the resulting arrangement. However, in practice, none of the existing mechanisms can process more than 1000 planar primitives without parallelisation schemes.

The key contribution of our work is an efficient mechanism for constructing more concise arrangements in less time than existing methods. To this end, we directly exploit input points to avoid unnecessary splitting operations and replace intersection tests between polyhedral cells and polygons with simply point-plane orientation tests. We then carefully order the plane insertion operations to further lower the computational complexity of the algorithm. We also introduce a remeshing and simplification strategy that reduces the number of extracted polygonal surface facets or the number of extracted convex polyhedra from the arrangement. The combination of these ingredients allows us to produce concise representations of complex objects and entire scenes from several thousand input planar shapes (cf. Fig. 1). We empirically demonstrate the effectiveness of our algorithm by comparing to previous plane arrangement, mesh simplification and mesh decomposition methods on various datasets.

2 Related Work

Our review of related work covers algorithms for concise surface reconstruction, convex decomposition of 3D shapes and construction of plane arrangements.

Concise Surface Reconstruction. One common way for reconstructing a concise surface mesh from data measurements consists in generating a dense triangle mesh from point clouds or multiview stereo images before reducing its number of triangle facets. While the first step relies upon a vast literature [27] and recent advances on NeRF, mesh simplification is mainly based on a few geometry processing algorithms that iteratively collapse edges [14, 26] or group triangle facets into planar clusters before remeshing [10]. Chen *et al*. [8] extract a feature-preserving offset of the dense mesh before collapsing edges and flowing vertices to be data consistent. These algorithms are fast and scalable, but deliver triangle meshes that usually fail to preserve the structure of objects. In contrast, plane assembly methods detect planar shapes from point clouds [16] and assemble them to form a polygonal surface mesh. Assembling can be done by considering the dual of an adjacency graph between planar shapes. However, the construction of such a graph is unlikely to be consistent and requires to be interactively completed [1, 28]. More robust assembly approaches rather decompose the 3D space into polyhedral cells with splitting operations induced by the planar shapes. The polygonal surface mesh is then extracted by a binary labelling of cells [3, 5, 20] or facets [4, 13, 22] from the decomposition. The construction of such plane arrangements has a high algorithmic complexity and no existing mechanism can process more than 1000 planar shapes without block decomposition schemes that introduce border artefacts. Our work follows an arrangement approach, but with an efficient construction mechanism which is able to handle up to two orders of magnitude more planar shapes.

Plane Arrangement. The trivial way to construct a plane arrangement is to iteratively slice the 3D domain with the supporting plane of each detected planar shape [12]. Such an *exhaustive* plane arrangement mechanism typically produces dense plane arrangements as even supporting planes of spatially distant polygons are likely to intersect. The exhaustive construction becomes intractable with only a few hundred shapes [22]. To improve on complexity and performance, the slicing operations can be restricted to the polyhedral cells that include the associated planar shape only [21]. This condition can be relaxed by initially dilating the polygon enclosing the shape [5] or by testing inclusion from the bounding box of cells [7]. In such an *adaptive* plane arrangement, the ordering in which input shapes are processed impacts both the quality of the output decomposition and the performance of its construction. However, ordering schemes proposed in the literature remain simple and application-driven, *e.g*. large [21] or vertical [7] shapes first. The *kinetic* mechanism KSR [3] does not perform slicing operations on polyhedral cells in an iterative manner. Instead, it grows 2D polygons at a constant speed until they collide and form polyhedral cells. KSR produces a

concise decomposition without proximity rules, but the collisions are costly to simulate. Our method follows an adaptive plane arrangement construction but with an efficient ordering scheme and the exploitation of raw measurements to significantly gain in performance and compactness.

Volumetric Decomposition of 3D shapes. Exact convex decompositions of 3D shapes typically produce a prohibitively large number of convex cells (in short, convexes) [2,6,15], while approximate convex decompositions offer a trade-off between the fidelity to the input shape and the number of convexes from the decomposition [19, 29]. Traditionally, axis-aligned split planes are used to partition the input shape into convex, or nearly convex parts [19, 29]. The decompositions are refined and simplified by moving cutting planes and grouping cells [29]. In our approach, we use cutting planes detected on the object surface instead, to achieve a high fidelity to the input shape. Furthermore, we efficiently group cells of our plane arrangement to produce a decomposition with low complexity.

Some recent neural network based methods also produce decompositions from other types of primitives such as cubes and cylinders [25, 30] or permit convexes of the decomposition to overlap [9, 11]. However, non-convex and overlapping decompositions are not desirable for applications such as collision detection or point localisation [29]. Moreover, most neural network based methods can only process small objects with simple geometries. In contrast, our method produces convex decompositions from complex objects and scenes.

3 Algorithm

Our approach takes a point cloud representing a 3D object or scene as input (Fig. 2a) and returns either a concise, watertight and intersection-free polygon surface mesh (Fig. 2d) or a concise volume mesh composed of convex polyhedra (Fig. 2e). First, we detect planar primitives defined as the association of supporting planes P and a subset of input points, called inliers, to which supporting planes are fitted (Fig. 2b). We can use any standard plane detection method for this step [23, 31]. We then construct a concise polyhedral decomposition from the planes and their corresponding inlier points (Fig. 2c). Finally, we extract the outer boundaries of the observed object by labelling each polyhedral cell of the decomposition as inside or outside the surface. For this step, we can use a proxy surface, *e.g.* from a Screened Poisson [17] reconstruction of the input point cloud, a normal-based approach [3] or a deep occupancy field [7]. We then apply a remeshing and simplification strategy to group facets (Fig. 2f) and cells (Fig. 2g) into larger components. In the following, we explain the adaptive plane arrangement, on which our algorithm is based, and present our key contributions.

3.1 Background on Adaptive Plane Arrangement

The adaptive construction mechanism first introduced by Murali & Funkhouser [21] converts a set of unoriented and disconnected polygons into a polyhedral

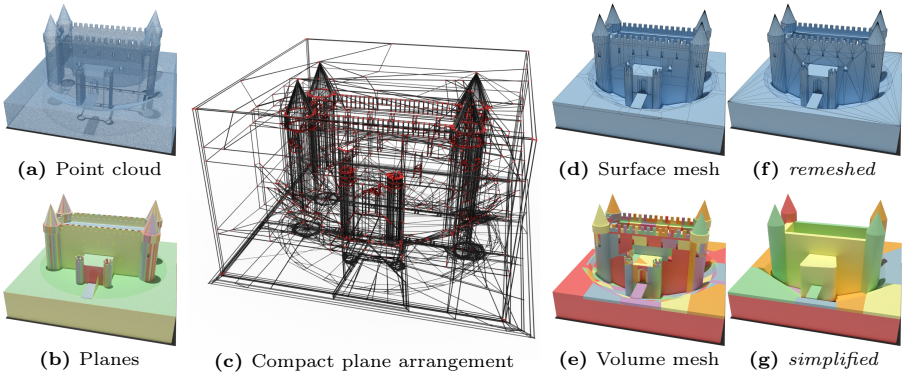
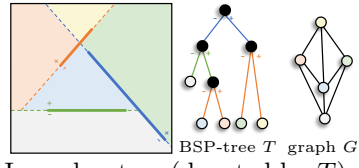


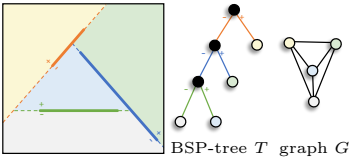
Fig. 2: Pipeline. Our method processes a set of planes with their corresponding inlier points (b) detected from a point cloud (a) into a compact plane arrangement (c). From this arrangement we extract either a watertight and intersection-free surface mesh (d) or a volume mesh with intersection-free convexes (e). The surface mesh can optionally be remeshed to represent each planar region with only one facet, and with Delaunay triangles for regions with holes (f). The remeshed surface is still watertight and intersection-free. The volume mesh can be simplified by merging groups of convex volumes and potentially allowing them to intersect (g).

decomposition, composed of a set of polyhedral cells C and a set of polygonal facets F. The input polygons are typically computed as the 2D convex hulls of inlier points projected onto their associated supporting planes. The algorithm starts by initializing the polyhedral decomposition to the 3D domain, *i.e.* to a padded bounding box of the input polygons. Then, it performs a series of splitting operations on the polyhedral decomposition, in which the polyhedral cells containing a polygon, or a part of it, are cut by the supporting plane of the polygon. The order in which polygons are processed is determined a priori and stored in a priority queue. In the inset below, the commonly used order, *i.e.* larger polygons first, split the domain by the blue-then-green-then-orange polygon (represented by the line segments), leading to a decomposition with five polyhedral cells at the end. A tree structure, typically a binary space partitioning (BSP)-tree, is used to track the splitting operations during the process. In such a tree (denoted by T), the root node T_0 corresponds to the 3D domain. The path from the root to a tree node T_i stores the splitting operations required to form the polyhedral cell c_i. At the end of the process, the leaves of the tree correspond to the polyhedral cells of the output decomposition. An undirected graph G, updated in tandem to the tree, is also used to store the adjacency between polyhedral cells.



3.2 Our Concise and Scalable Plane Arrangement

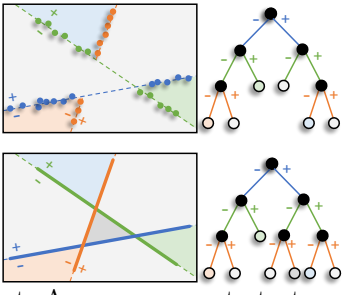
Ordering of Splitting Operations. In the various versions of the adaptive plane arrangement [5,7,21], the priority queue is computed at initialization using considerations on the area or orientation of the input polygons. Unfortunately, such an ordering scheme is too simple to produce polyhedral partitions in a stable and efficient manner. We tackle this problem with a dynamic ordering scheme that seeks a decomposition with a low number of polyhedra, or equivalently a BSP-tree with a low number of leaves. Instead of a priority queue that sorts the input polygons, we use a function that selects the next splitting operation, *i.e.* a pair of one plane and one polyhedral cell, to be processed given the current state of the partition. It relies upon two key ideas:



i. a splitting operation that creates a polyhedral cell that cannot be split anymore, must take priority,
ii. splitting operations that create two cells containing each an as high and identical as possible number of polygons must be favored over other splits.

The intuition behind (i) is that turning nodes into leaves in the BSP-tree as soon as possible reduces the growth of the tree, whereas (ii) aims to reduce the number of intersection computations and balances the number of polygons in the cells of the decomposition to reduce the algorithmic complexity over the 3D domain. In the inset, continuing with the same polygon layout as in the previous example, we first split the domain along the orange polygon, *i.e.* the only one respecting condition (i) at initialisation, then by the blue, then green polygon, also validating this condition. The BSP-tree is simpler and the decomposition has only four cells at the end.

Fast Intersection Tests with Inlier Points. Frequent intersection tests between polyhedra and planar primitives are necessary to identify the cells to split. These tests constitute a computational bottleneck in the various versions of the mechanism, even with approximation by bounding boxes [7]. We address this issue by replacing intersection tests by a simple distribution of the inlier points in the BSP-tree. The idea is that the assignment of inlier points to the nodes of the BSP-tree will easily provide their relative positioning to the cutting planes. We rely on a hierarchical clustering of the inlier points based on simple point-plane orientation tests for reassigning the inlier points after a splitting operation. This strategy leads to a strong reduction of the computational burden and directly exploits the precision of input points. This is particularly beneficial when planar shapes are concave or contain holes, as illustrated in the top decomposition of the inset. An arrangement strategy using only planes [12] or convex polygons [7,21] cannot avoid the creation of the meaningless center cell in the bottom decomposition.

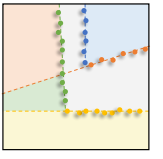


Practical Description. We start by initialising the root node of a BSP-tree with a dilated bounding box of all inlier points. The root node is then associated with all inlier points and their corresponding supporting planes. We now iterate over the leaves of the tree and check if all inlier points associated with the current node can be located on the same side of a supporting plane inside the cell. If we find such a plane, we split the cell along it. Otherwise, for each supporting plane p we compute the sets of inlier groups L_p and R_p that lie fully left and right of p. We then select the split that maximizes the product $|L_p||R_p|$, *i.e.* we cut the current cell with the supporting plane that splits the inlier sets as equally as possible, while intersecting as few other polygons as possible. Finally, we reassign the inlier points and their associated supporting planes to the two new child nodes. When there are no more polyhedra containing points, *i.e.* no points associated with the leaves of the tree, the splitting operations stop.

3.3 Remeshing and Simplification

Aggregation of Convex Facets. For surface modelling, we extract a watertight and intersection-free mesh with convex polygonal facets from the decomposition (Fig. 2d). We then collect clusters of coplanar facets and extract the boundary edges, *i.e.* the edges that only occur once per cluster. For each cluster, we find all cycles in the set of boundary edges [24]. In case of multiple cycles for which one cycle is contained in another, *i.e.* a facet with holes, we apply a 2D Delaunay triangulation constrained by the boundary edges (Fig. 2f).

Aggregation of Convex Cells. For volume modelling, the convex cells of the reconstructed mesh can be merged into fewer, larger convex cells. In the inset, the orange and green cell can be merged, to produce a more concise decomposition with the same geometry. While our ordering strategy aims to reduce such unwanted splits, they cannot fully be avoided due to multiple interactions of planes for complex geometries. We can exploit the properties of the BSP-tree in which each parent node corresponds to the union of its two children. Starting from the leave nodes, we recursively replace two siblings by their parent cell if they lie on the same side of the surface. This allows us to reduce the number of cells in the decomposition by around 25% without any geometrical computations. Once all same-sided siblings are processed, the decomposition can still include pairs of same-sided adjacent cells a, b whose union is equivalent to the convex hull of their union $\text{conv}(a \cup b)$. We recursively test this condition for all pairs of adjacent cells of the mesh, given by the adjacency graph G, by comparing the sum of the volumes of the two cells with the volume of the convex hull of their union [19]. We replace adjacent cells a and b with their convex hull $\text{conv}(a \cup b)$, if



$$|V_a + V_b - V_{\text{conv}(a \cup b)}| < \tau, \tag{1}$$

where V_x is the volume of cell x and τ a positive parameter. τ determines how much we allow the volume of the new cell $\text{conv}(a \cup b)$ to differ from the combined

volumes of a and b. If $\tau = 0$ the geometry stays unaltered and the decomposition is guaranteed to be intersection free (Fig. 2e). Relaxing τ allows to alter the geometry and the cells in the decomposition to potentially overlap (Fig. 2g). In the inset, we could then simplify further by replacing the orange, green *and* yellow cell with the convex hull of their union.

4 Experiments

In this section, we first compare the efficacy and efficiency of our plane arrangement against existing construction mechanisms and conduct an ablation study. We then evaluate the competitiveness of our general pipeline on surface and volume modelling tasks. Whenever possible we use the code, data and evaluation pipelines provided by the authors of the corresponding baselines.

4.1 Evaluation on Plane Arrangement Construction

Experimental Setup. We compare our plane arrangement against several baselines on the Thingi10k [32] dataset.

Baselines. (i) We compare to an *exhaustive* arrangement which we implement by intersecting all detected input planes. (ii) We compare to an *adaptive* arrangement, which we implement following ideas of the building reconstruction pipeline P2P [7]. See the supplementary material for a detailed comparison of our implementation to the one of P2P. (iii) We compare to a *kinetic* plane arrangement (KSR) for which we use the implementation provided by the authors. We construct arrangements with the four different mechanisms, occupancy-label the cells of the arrangement with the normal-based method of KSR and extract polygonal surface meshes from the interface of adjacet inside/outside cells.

Dataset. We use 1000 non-degenerate models randomly chosen from Thingi10k. We sample 200k points on each model and detect planar shapes [23, 31] using a fitting tolerance parameter fixed to 0.8% of the models' bounding box diagonal. We input the same plane configurations to all four different construction mechanisms. Because the construction mechanisms do not scale equally, we split the dataset into three groups: *simple*, for models approximated by less than 100 planar shapes, *moderate*, if between 100 and 250, and *complex* for models with more than 250 detected planes.

Metrics. We evaluate (i) the complexity of the arrangement with the number of polyhedral cells $|C|$ and the complexity of the surface with the number of polygonal surface facets $|F_S|$, (ii) the accuracy of the extracted surface with the symmetric Chamfer (CD) and Hausdorff (HD) distances between ground truth and reconstructed surface and (iii) the performance of mechanisms with the construction time and memory peak.

Table 1: Quantitative Comparison with Plane Arrangements. The exhaustive method is tested on the simple group of models S only to not exceed reasonable processing time, *i.e.* $> 300h$. Similarly, KSR is not tested on the complex group L. The moderate group M includes models containing between 100 and 250 planar shapes. $|C|$ refers to the average number of cells in the decompositions. $|F_S|$, CD and HD are the average number of facets of the reconstructed mesh and symmetric Chamfer and Hausdorff distances between the ground truth and reconstructed mesh, respectively.

		Complexity		*Accuracy*		*Performance*
		$\|C\|$	$\|F_S\|$	CD ($\times 10^2$)	HD ($\times 10^2$)	Time (s)
S	**Exhaustive** [12]	5690	890	**0.190**	**1.05**	8.03
	Adaptive [21]	324	70	0.196	1.32	9.99
	KSR [3]	161	202	0.196	1.22	4.08
	Ours	**73**	**53**	0.193	1.15	**3.06**
M	**Adaptive** [21]	1760	267	**0.247**	2.36	97.4
	KSR [3]	757	660	0.266	2.34	139
	Ours	**256**	**167**	0.254	**2.32**	**16.3**
L	**Adaptive** [21]	6150	1020	0.227	2.82	504
	Ours	**705**	**478**	**0.224**	**2.78**	**68.1**

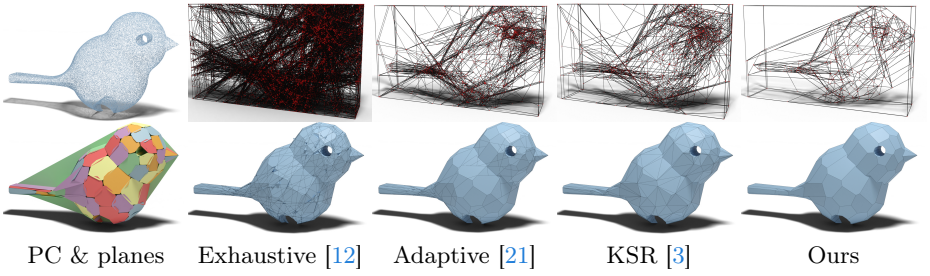


Fig. 3: Comparison with Plane Arrangements. Our algoritm builds a more concise plane arrangement than existing mechanisms (top row), leading to a polyon mesh with fewer facets (bottom row).

Comparison. Table 1 presents the quantitative results. For simple models, our algorithm offers the best complexity with more than two times less cells compared to the second best and one order magnitude less on complex ones. Fig. 3 illustrates this complexity gap on a simple model. The significant gain in complexity is not done at the expense of the quality of the reconstructed meshes as both the Hausdorff and Chamfer distances remain competitive with a lower number of polygonal facets. The exhaustive method, which performs on simple models only, offers the best accuracy, but produces overly complex meshes. KSR and Adaptive exhibit a similar accuracy to our method, but with a higher number of polygonal facets. Our method is also faster, especially on complex models. Both performance and complexity gains originate from the combination of our ordering scheme, the use of the inlier points that accurately describe the input data, and our remeshing strategy.

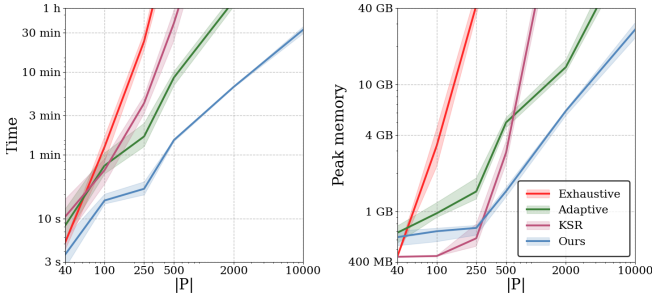


Fig. 4: Scalability. Average runtime (left) and memory peak (right) in function of the number of input planar shapes for different construction mechanisms. The transparent band around each curve indicates the minimal and maximal values measured on various models. Our algorithm offers the best performance and can process 10k planar shapes in around 30 minutes without exceeding the memory consumption of a standard computer. It also exhibits a better stability than other methods whose variation bands are thicker.

Scalability. We further evaluated the scalability of plane arrangements by generating configurations of planar shapes at six levels of complexity, ranging from 40 to 10k planar shapes, from six different real-world scans. Graphs presented in Fig. 4 compare processing time and peak memory as a function of the number of input planar shapes for the different construction mechanisms. Our algorithm offers the best performances and the best stability at the different levels of complexity. The gap is particularly large for configurations with a high number of planes. Only the kinetic algorithm exhibits a lower memory peak for configurations with no more than 250 planes.

Ablation Study. Table 2 shows the impact of our design choices on the various evaluation metrics. We first examine the impact of our ordering scheme. The basic area-based sorting scheme leads to a much more complex decomposition, a less accurate surface and almost 2× longer runtime. Removing the priority condition (i) leads to a decomposition with more cells. The cells of the decomposition have less facets on average, which leads to a slightly faster runtime. However, condition (i) also seeks to better recover the general shape of an object by first inserting all planes that lie on the convex hull. Consequently, removing condition (i) leads to a worse Hausdorff distance. Table 2 also shows the benefits of using inliers points for the intersection tests. We can reduce the complexity and construction time of the decomposition, but strongly degrade the quality of the reconstructed mesh by using only the vertices of 2D convex hulls of inlier points projected onto their supporting plane. A relevant alternative is to use 1M points uniformly sampled on the convex polygons, but this produces a more complex decomposition than our vanilla implementation. Finally, the surface remeshing only impacts the complexity of the reconstructed mesh. Removing the cell simplification (with $\tau = 0$) impacts the complexity of the decomposition and leads to a higher runtime due to a longer surface extraction.

Table 2: Ablation Study. Alternative schemes to the use of inlier points and for the sorting of splitting operations, as well as the deactivation of the remeshing step are evaluated from the complex group of the Thingi10k models.

	Complexity		*Accuracy*		*Performance*
	$\|C\|$	$\|F_S\|$	CD ($\times 10^2$)	HD ($\times 10^2$)	Time (s)
Ours	705	**478**	**0.224**	2.78	68.1
Insertion order					
Convex hull area (high to low)	1010	549	0.231	3.10	125
arg max$_p$($\|L_p\|\|R_p\|$) only	920	491	0.225	2.87	**57.4**
Use of inlier points					
Convex hull vertices only	**653**	539	0.302	5.25	58.9
Points sampled on convex hull	839	**478**	0.226	**2.73**	66.1
Remeshing and simplification					
Without facet aggregation	705	557	**0.224**	2.78	68.5
Without cell aggregation	846	**478**	**0.224**	2.78	73.5

4.2 Surface Mesh Simplification

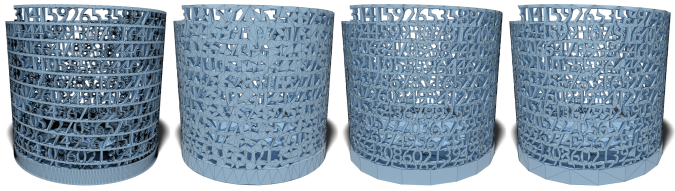
Experimental Setup. Besides the reconstruction of low-poly meshes directly from point clouds, another relevant application of our pipeline is the simplification of dense, high-poly surfaces into low-poly ones. We compare our method to Robust Low-Poly Meshing (RLPM) [8]. RLMP simplifies dense surface meshes by computing an offset surface of the input and iteratively simplifying this offset. The method establishes itself as state-of-the-art in low-poly mesh generation by comparing to over ten other low-poly meshing algorithms. We compare our method to RLMP on the dataset provided by the authors, *i.e.* a subset of 100 models from Thingi10k [32]. We exclude 45 models that have a non-orientable surface, because orientability is a requirement for the inside/outside labelling in the surface extraction step of our pipeline. We sample 2M points per model and use the same plane detection parameters as in the previous experiment. Because RLPM produces triangle meshes and our method produces polygon meshes we triangulate our output and perform edge collapse based on QEM [14] to produce models with the same number of triangles as the ones of RLPM. We call this variant *OursTri*. Note that, the authors of RLPM also experiment with replacing parts of their pipeline with QEM in their paper, which leads to worse results.

Results. Fig. 5 shows quantitative and qualitative results of the output meshes. Our method produces polygon meshes with a similar number of polygons (Ours) and triangles (OursTri) than RLMP, but with a much better accuracy while also exhibiting a shorter runtime. The remeshing and decimation operations of RLPM progressively degrade the accuracy on small details. In contrast, planar shapes that capture such details allow a more precise, yet concise approximation of the local geometry and facilitate the preservation of details.

	Perform.	*Complexity*	
	Time (s)	$\lvert V_S \rvert$	$\lvert F_S \rvert$
RLPM [8]	503	899	1969
Ours	**410**	2051	2045
OursTri	411	**879**	**1936**

Accuracy		
CD ↓ ($\times 10^2$)	HD ↓ ($\times 10^2$)	NC ↑
0.269	2.48	0.923
0.218	**2.10**	**0.941**
0.244	2.32	0.924

(a) Quantitative



(b) Qualitative

Fig. 5: Surface Mesh Simplification. We compare RLPM and our pipeline on models from Thingi10k. Because RLPM produces triangle meshes and our method produces polygon meshes we triangulate our output and perform edge collapse based on QEM to reach a comparable number of triangles (OursTri). (a) The runtime of the full pipeline, number of vertices $|V_S|$ and facets $|F_S|$ of the surface meshes (*i.e.* number of triangles for RLPM and OursTri, and number of polygons for Ours), Chamfer and Hausdorff distance and normal consistency (NC) between ground truth and reconstruction. (b) The reconstructions of the *Tower of Pi* from the dataset. Note how both, our polygon and our triangle mesh is much more detailed compare to the one of RLPM. The meshes of OursTri and RLPM have the same number of triangles.

4.3 Volume Decomposition with Intersection-Free Convexes

Experimental Setup. Another relevant application of our pipeline is the decomposition of volumes into a low number of intersection-free convexes. We compare our pipeline with the specialized method CoACD [29] on 50 challenging objects and scenes from *Thingiverse*. We use the implementation provided by the authors. CoACD cuts an input solid mesh with equally spaced axis-alligned planes to produce a polyhedral cell decomposition. The cells are then merged into larger cells using a multi-step tree search. Cells that border the exterior are replaced by their convex hull. CoACD [29] also provides a way to tune the fidelity and complexity of the decomposition. To compare our volume decompositions with the ones of CoACD we sample the input mesh with 2M points and produce plane configurations with different complexities by varying the fitting tolerance and the minimal number of inliers per plane.

Results. We show complexity / accuracy curves for our and CoACD's decompositions in Fig. 6. To produce decompositions with a small number of cells ($|C_V| < 400$) our pipeline relies on a low number of input planes which are sometimes not sufficient to approximate the input well. CoACD directly relies on the input mesh and thus exhibits a higher volumetric intersection over union. However, this comes at the cost of a much higher number of convex cell facets (see Fig. 6b). For more complex and less approximate decompositions ($|C_V| > 500$) our method is both more accurate and more concise. This is also exemplified in Fig. 6c, where the top row shows two decompositions with different complexity

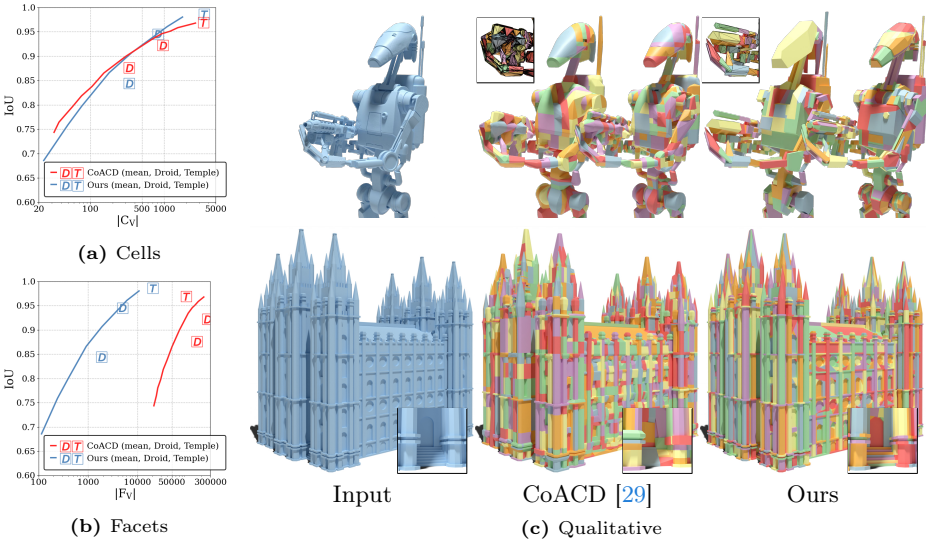


Fig. 6: Volume Decomposition with Non-Overlapping Convexes. We plot the volumetric IoU in function of the number of cells $|C_V|$ (a) and facets $|F_V|$ (b). Our algorithm offers a trade-off between IoU and number of convexes similar to the one of CoACD [29], but with convexes with fewer facets. In (c), the top row shows two decompositions for both methods of the *Droid* with different levels of complexity. The bottom row shows a decomposition of the *Temple*. For a similar number of convexes (colored cells), our algorithm captures the geometry better (see stairs on the bottom close-up), with fewer facets (see edges on the top close up).

of the *Droid* model and the bottom row a decomposition of the *Temple* model. Because our method uses cutting planes that are detected on the surface of the input, the cells of our decomposition represent the geometry much better. See for example the gun of the *Droid* or the stairs in the close up of the *Temple*.

4.4 Volume Decomposition with Overlapping Convexes

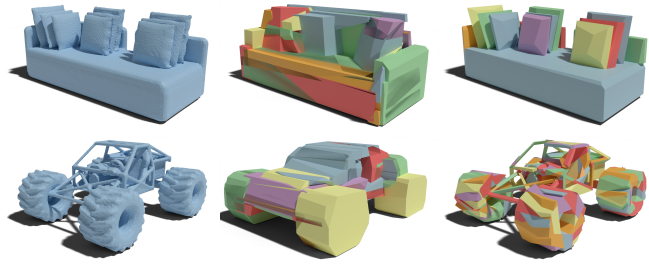
Experimental Setup. A variant of the convex decomposition problem is to relax the non-overlapping constraint between convexes. To address this task, we set the merge threshold τ of Eq. 1 to a strictly positive number, which allows us to merge neighboring cells, and extract a decomposition at any desired number of convexes. We compare our method to BSP-Net [9] on the test sets of all 13 categories of ShapeNet, *i.e.* 8762 models in total, provided by the authors of BSP-Net. BSP-Net is a neural network based approach that learns to fit planes to input observations and assemble the planes into a set of overlapping convex polytopes. We use the auto-encoder variant of the network with weights trained (provided by the authors) on all 13 categories of ShapeNet. The network inputs a voxel-grid of 64^3 occupancy values per model, while we run our method on 100k points sampled on the models' surface. To find the best trade-off between complexity and accuracy of the produced models we operate a small grid

search on the ShapeNet train set to determine the merge threshold τ and the two parameters of the planar shape detection. Alternatively, we could also merge neighboring cells until our models have the same number of convexes as the models produced by BSP-Net. However, we find that BSP-Net often does not output a suitable number of convexes to accurately describe a models geometry. We use the evaluation pipeline provided by the authors of BSP-Net to compute the *squared* Chamfer distance (CD^2) and normal consistency (NC). Note that, the pipeline computes these metrics only for surface points by using the occupancy of input voxels to determine whether a point lies inside a shape or on its surface.

	Complexity	
	$\lvert C_V \rvert$	$\lvert F_V \rvert$
BSP-Net [9]	60.9	1646
Ours	**41.7**	**773**

Accuracy	
$CD^2 \downarrow$ ($\times 10^3$)	NC ↑
0.771	0.817
0.385	**0.919**

(a) Quantitative



(b) Qualitative

Fig. 7: Comparison on Volume Decomposition with Intersecting Convexes. BSP-Net produces highly overlapping convexes (see the z-fighting) and exhibits generalisation issues (see right armrest of the sofa). Our approach produces weakly overlapping convexes that more accurately capture the fine components of the models. Note, how each pillow is described by a single convex.

Results. Fig. 7 shows that our method offers both better accuracy and complexity than BSP-Net. The two visual results illustrate this quality difference with convexes capturing more details and more meaningful components of the models. Our method manages to capture thin components with single convexes whereas BSP-Net tends to regroup them into large convexes that strongly overlap. Note that the same set of parameters found by grid searching generalises well to various shapes with different feature sizes.

5 Conclusion

We propose a scalable method for converting point clouds into concise plane arrangements. We construct our plane arrangement using (i) a specific ordering of splitting operations, (ii) input points for fast intersection queries and (iii) BSP-tree properties for simplifying the arrangement. These steps significantly reduce the computational complexity of the construction algorithm while producing

concise and meaningful arrangements. We also introduce methods to extract lightweight polygonal surface and volume meshes composed of a low number of convexes from plane arrangements. We demonstrated the efficiency, scalability and competitiveness of our algorithm on different low-poly mesh reconstruction problems against the best methods in the field. In future work, we will investigate how to jointly detect and arrange planes from 3D data measurements.

References

1. Arikan, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S.: O-snap: Optimization-based snapping for modeling architecture. ACM Transactions on Graphics (TOG) **32**(1) (2013) 3
2. Bajaj, C.L., Dey, T.K.: Convex decomposition of polyhedra and robustness. SIAM Journal on Computing **21**(2) (1992) 4
3. Bauchet, J.P., Lafarge, F.: Kinetic shape reconstruction. ACM Transactions on Graphics (TOG) **39**(5) (2020) 1, 3, 4, 9
4. Boulch, A., de La Gorce, M., Marlet, R.: Piecewise-planar 3d reconstruction with edge and corner regularization. Computer Graphics Forum **33**(5) (2014) 3
5. 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) 1, 3, 6
6. Chazelle, B.: Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. SIAM Journal on Computing **13**(3) (1984) 4
7. 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) 2, 3, 4, 6, 8
8. 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) 3, 11, 12
9. 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) 4, 13, 14
10. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Transactions on Graphics (TOG) **23**(3) (2004) 3
11. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: Cvxnet: Learnable convex decomposition. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 4
12. Edelsbrunner, H., O'Rourke, J., Seidel, R.: Constructing arrangements of lines and hyperplanes with applications. SIAM Journal on Computing **15**(2) (1986) 2, 3, 6, 9
13. Fang, H., Lafarge, F.: Connect-and-slice: an hybrid approach for reconstructing 3d objects. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 3
14. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques (1997) 3, 11
15. Hershberger, J.E., Snoeyink, J.S.: Erased arrangements of lines and convex decompositions of polyhedra. Computational Geometry **9**(3) (1998) 4
16. Kaiser, A., Ybanez Zepeda, J.A., Boubekeur, T.: A survey of simple geometric primitives detection methods for captured 3d data. Computer Graphics Forum **38**(1) (2019) 3

17. Kazhdan, M., Hoppe, H.: Screened Poisson surface reconstruction. ACM Transactions on Graphics (TOG) (2013) 2, 4
18. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and Temples. ACM Transactions on Graphics (TOG) (2017) 2
19. Mamou, K., Lengyel, E., Peters, A.: Volumetric hierarchical approximate convex decomposition. In: Game Engine Gems 3. AK Peters (2016) 4, 7
20. Mura, C., Mattausch, O., Pajarola, R.: Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. Computer Graphics Forum **35**(7) (2016) 3
21. Murali, T., Funkhouser, T.A.: Consistent solid and boundary representations from arbitrary polygonal data. In: Proceedings of the symposium on Interactive 3D graphics (1997) 2, 3, 4, 6, 9
22. Nan, L., Wonka, P.: Polyfit: Polygonal surface reconstruction from point clouds. In: International Conference on Computer Vision (ICCV) (2017) 1, 3
23. Oesau, S., Verdie, Y., Jamin, C., Alliez, P., Lafarge, F., Giraudot, S.: Point set shape detection. In: CGAL User and Reference Manual. CGAL Editorial Board, 4.14 edn. (2018) 4, 8
24. Paton, K.: An algorithm for finding a fundamental set of cycles of a graph. Communications of the ACM **12**(9) (1969) 7
25. Ren, D., Zheng, J., Cai, J., Li, J., Jiang, H., Cai, Z., Zhang, J., Pan, L., Zhang, M., Zhao, H., et al.: Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In: International Conference on Computer Vision (ICCV) (2021) 4
26. Salinas, D., Lafarge, F., Alliez, P.: Structure-Aware Mesh Decimation. Computer Graphics Forum **34**(6) (2015) 3
27. Sulzer, R., Marlet, R., Vallet, B., Landrieu, L.: A survey and benchmark of automatic surface reconstruction from point clouds. arXiv preprint arXiv:2301.13656 (2023) 3
28. Van Kreveld, M., Van Lankveld, T., Veltkamp, R.C.: On the shape of a set of points and lines in the plane. Computer Graphics Forum **30**(5) (2011) 3
29. Wei, X., Liu, M., Ling, Z., Su, H.: Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. ACM Transactions on Graphics (TOG) **41**(4) (2022) 4, 12, 13
30. Yu, F., Chen, Q., Tanveer, M., Amiri, A.M., Zhang, H.: D2CSG: Unsupervised learning of compact CSG trees with dual complements and dropouts. In: Conference on Neural Information Processing Systems (NeurIPS) (2023) 4
31. Yu, M., Lafarge, F.: Finding good configurations of planar primitives in unorganized point clouds. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2022) 4, 8
32. Zhou, Q., Jacobson, A.: Thingi10k: A dataset of 10, 000 3d-printing models. arXiv preprint arXiv:1605.04797 (2016) 8, 11