NURBSFit: Robust Fitting of NURBS Surfaces to Point Clouds

Lizeth J. Fuentes Perez
University of Zurich
fuentes@ifi.uzh.ch

Florent Lafarge
Centre Inria d'Université Côte d'Azur
florent.lafarge@inria.fr

Renato Pajarola
University of Zurich
pajarola@ifi.uzh.ch

Abstract

NURBS surfaces are compact parametric representations widely used in Computer-Aided Design (CAD) modeling. Decomposing raw 3D data measurements into a set of such elements is a challenging problem that existing methods approach by learning from CAD databases to both segment synthetic data and fit parametric shapes on each segment. Unfortunately, these methods generalize poorly to raw data measurements, with low robustness to imperfect data and complex objects and low scalability. To address this issue, we propose NURBSFIT*, an algorithm that fits NURBS surfaces to unorganized 3D point clouds, such as those generated by laser and photogrammetry acquisition systems. Starting with a fine configuration of planar patches that approximate the object geometry, our algorithm performs merging operations that progressively regroup pairs of adjacent patches into fewer, more expressive NURBS surfaces. This process is designed to be both robust and performant with a series of technical ingredients that include an energy that controls the global quality of a configuration of NURBS surfaces and an efficient ordering of the merging operations based on a cost-efficient quadric surface fitting analysis. We show the potential of our algorithm on both synthetic and real-world data and its efficiency against existing primitive fitting methods with results both simpler and geometrically more accurate. Our implementation is available at: https://github.com/lizOnly/nurbsfit*

1. Introduction

Widely used in Computer-Aided Design (CAD) for the design and production of 3D content, Non-Uniform Rational Basis Splines (NURBS) surfaces are powerful geometric tools for representing 3D shapes in a compact manner. In particular, they can be easily edited by intuitive manipulation of their control points [33]. During the past few years, these tools have shown potential in reverse CAD modeling, to capture objects with a CAD-based representation of 3D data measurements [6, 26, 47].

Decomposing 3D data measurements – typically point clouds – into a set of NURBS surface patches constitutes a key problem in reverse CAD modeling, with multiple objectives: we seek jointly (i) fidelity with a low geometric error between the surface patches and the input data, (ii) simplicity with as few as possible patches to represent the structural parts of the object, and (iii) efficiency with algorithms both scalable and robust to real-world data. Existing methods [10, 22, 24, 27, 37, 46] typically learn from CAD databases how to group input data into geometrically regular regions and how to combine them with parametric functions that approximate their surface. These methods perform well on point clouds ideally sampled from simple CAD models, but they generalize poorly to more complex objects and scenes, and to non-synthetic point clouds. Unfortunately, this lack of generalization strongly limits their usability in real-world reverse CAD applications.

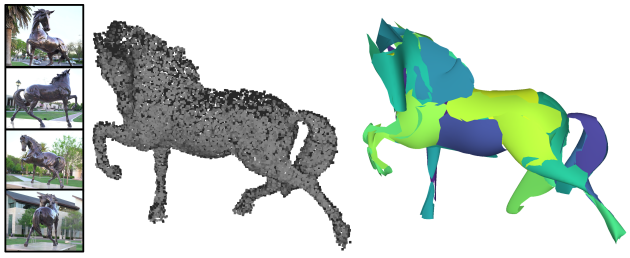


Figure 1. NURBSFIT. Our algorithm decomposes a 3D point cloud (left) into few NURBS surface patches that approximate the shape of the observed object (right, each NURBS is represented by a colored surface patch). The input point cloud is here generated by MultiView Stereo from the Tanks and Temples dataset [18].

To address this issue, we propose NURBSFit, an algorithm designed to also fit NURBS surface patches to imperfect, non-synthetic point clouds. By observing that recent planar shape detection methods offer high efficiency while accurately approximating freeform shapes by piecewise-planar layouts, we adopt a bend-and-merge strategy from such planar shape configurations. Instead of considering a data clustering problem as in previous work, our approach consists of a progressive merging of pairs of adjacent NURBS patches after having been bent from a set of planar shapes. The merging operations are proposed dynamically and prioritized based on a cost-efficient quadric

surface fitting analysis between the patches: this accumulative formulation allows us to avoid the compute-intensive refitting of all possible pairs of NURBS patches by simply considering sums of covariance matrices. The gain of each proposed merge is measured over the entire configuration of NURBS, and operated effectively when positive.

We show the potential of our algorithm on different datasets and its efficiency against existing methods. In particular, our algorithm outperforms competitors by a large margin on real-world input data from the KSR42 dataset [2] while remaining competitive on synthetic point clouds sampled from CAD models of the ABC dataset [19].

2. Related Work

Our review of previous work covers models to fit parametric surface primitives to 3D point clouds, as well as reverse CAD approaches that additionally assemble primitives.

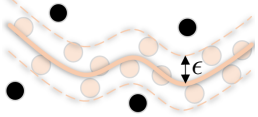
Traditional methods. They focus on fitting simple primitives – mostly planes and quadrics – with fast and scalable iterative mechanisms [17]. RANSAC [36, 41] and region growing [28, 32] are two popular approaches widely used on real-world scans. Such mechanisms have been extended to also discover and enforce geometric regularities in between primitives [23, 30] or detect primitives at multiple structural scales [9, 21]. Interestingly, GoCoPP [48] reformulated the fitting problem to approximate freeform shapes with planar primitives. Partly inspired by this approach, our method operates on more generic NURBS primitives that cannot efficiently and optimally be computed by principal component analysis as for planes. This leads us to design a smart exploration strategy instead of massively called, cheap geometric operators as in GoCoPP. From such generic primitives, existing works [1, 7, 49] typically rely on mapping functions that are unlikely to parametrize patches correctly on topologically complex shapes, and only perform well on simple, defect-free data. In contrast, our approach progressively merges patches from an initial planar approximation of the 3D shapes into NURBS patches, without unstable mapping operators.

Neural networks. Neural models typically learn to jointly cluster the input points, identify the primitive type of each cluster, and estimate its primitive parameters. These methods often differ in some key design choices that include the embedding strategy and primitive types. The default set of primitive types usually includes planes, spheres, cylinders, and cones [22, 27, 35]. Closer to our work, ParSeNet [37] and HPNET [46] also consider open and closed B-splines as primitive types, while Deepspline [11] fits both B-spline curves and surface patches. Unfortunately, the efficiency of these methods quickly drops with real-world point clouds due to (i) a low scalability (the maximal number of primitives and input points are limited) and (ii) a poor generalization (training relies on point clouds ideally sampled from simple CAD models, typically the ABC dataset [19]). To our knowledge, only PrimitiveNet [15] proposes a scalable approach by learning local surface properties that are then aggregated by region growing. However, this hybrid approach is limited to planes and quadrics.

Reverse CAD. Reconstructing CAD models requires not only fitting surface primitives but also assembling them into explicit 3D representations such as polygon surface meshes or B-Reps. For planar primitives, existing methods typically compute exact intersections using plane arrangements [2, 14, 31, 40]. This efficient strategy can also be used for quadrics, by embedding triangle surface meshes into a polyhedral partition [16]. The assembly of more complex primitives such as NURBS or freeform surface patches is a more delicate task that usually requires mesh trimming operations [4, 26, 45]. This scenario assumes a correct connectivity graph between primitives, which cannot be guaranteed in practice from real-world data. Some neural models also operate with B-Reps, by recovering geometric primitives of different orders (i.e., corners, curves, and surface patches) and their mutual topological relations [13, 20, 24, 25], or by directly learning the topological structure between the object parts [39, 44]. Reverse CAD can also be done with implicit representations by neural networks that learn sequences of constructive solid geometry operations [34, 38, 47] or sketch extrusions [6]. In contrast, our work does not address the primitive assembly problem and focuses on the robust fitting of NURBS primitives.

3. Algorithm

Our algorithm takes as input a raw 3D point cloud. Note that the latter is not limited in size, in contrast to those processed by neural fitting models. It returns a set of NURBS primitives that approximate the input point cloud. Each primitive is characterized by (i) a NURBS surface patch through a sparse grid of control points $\mathbf{C}_{ij}$, and (ii) a set of inliers, i.e., the subset of input points that are interpolated by the NURBS surface. Similarly to robust plane and quadric fitting techniques, an input point is considered an inlier if its distance to the NURBS surface is below a user-defined tolerance ϵ. This tolerance adsorbs noise and other imperfections contained in 3D scanned input point data, a high value being recommended with defect-laden data. A NURBS surface patch is defined as the tensor product of two NURBS curves, using two independent parameters u and v so that the point of the NURBS surface at (u, v) is given by



$$\mathcal{S}(u,v) = \sum_{i=1}^{K} \sum_{j=1}^{L} R_{ij}(u,v)\mathbf{C}_{ij}$$

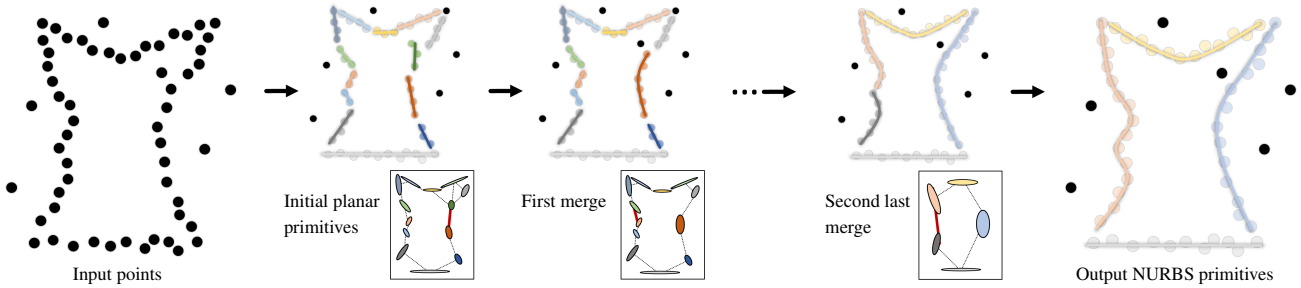


Figure 2. Principle of NURBSFIT. Planar primitives are detected from the input points and used as the initial configuration (colored lines, colored points and black points correspond to the detected planes, their inliers and outliers respectively). To guide the merging process, an adjacency graph is built and quadric surface fitting errors are computed for each primitive (respectively dash black lines and colored ellipses in the bottom frames). The first merge is operated in between the pair of adjacent primitives with the lowest cumulated quadric fitting error (red thick line), before updating the adjacency graph and the quadric covariance matrices. The process is repeated until no merging operation decreases the energy further.

where $R_{ij}(u, v)$ are the rational basis functions. More details on NURBS are can be found in [33].

3.1. Problem Formulation

The output NURBS primitives correspond to clusters of inliers that are each associated with a NURBS surface patch. We denote a configuration of NURBS primitives by $\mathbf{x} = (\mathcal{S}, \mathcal{L})$ where $\mathcal{S}$ is a set of NURBS surface patches with a fixed number of control points living in the continuous domain, and $\mathcal{L}$ is a set of (discrete) labels that specify whether each input point is an outlier or an inlier to one of the NURBS surface patches $S_i \in \mathcal{S}$. We measure the quality of a configuration $\mathbf{x}$ by a three-term energy U of the form

$$U(\mathbf{x}) = \lambda_f U_f(\mathbf{x}) + \lambda_s U_s(\mathbf{x}) + \lambda_c U_c(\mathbf{x}), \qquad (1)$$

where U_f, U_s and U_c are the three energy terms for *fidelity*, *simplicity* and *completeness*. These terms are normalized between 0 and 1, and weighted by corresponding factors λ_f, λ_s, and λ_c. This energy directly expresses the trade-off existing in point-based fitting problems between (i) fidelity with NURBS surface patches close to their inliers, (ii) simplicity with a low number of primitives, and (iii) completeness with a low ratio of outliers.

The fidelity term measures the mean distance error between the NURBS surfaces and their associated inlier points across all patches. It is defined as

$$U_f(\mathbf{x}) = \frac{1}{n_{\text{in}}} \sum_{j=1}^{|\mathcal{S}|} \sum_{i=1}^{n_{\text{in}}^j} d_\epsilon(p_i^j, S_j) \qquad (2)$$

where $|\mathcal{S}|$ is the number of NURBS primitives in configuration $\mathbf{x}$, n_{in}^j is the number of inlier points for the j-th NURBS patch, n_{in} is the total number of inlier points across all patches, and $d_\epsilon(p_i^j, S_j)$ is the Chamfer distance between the inlier point p_i^j and the NURBS surface S_j.

The simplicity term encourages configurations with a low number of primitives and is defined as

$$U_s(\mathbf{x}) = \frac{|\mathcal{S}|}{|\mathcal{S}_0|} \qquad (3)$$

where $|\mathcal{S}_0|$ is the maximum number of NURBS primitives which is set as the number of primitives in the initial configuration. We explain this point later when describing the optimization process.

The coverage term favors configurations with a high ratio of inliers and is defined as

$$U_c(\mathbf{x}) = 1 - \frac{n_{\text{in}}}{n} \qquad (4)$$

where n is the total number of input points.

3.2. Optimization

Minimizing energy U cannot be addressed with the traditional optimization toolbox, because the configuration $\mathbf{x}$ lives in a mixed continuous-and-discrete solution space of varying dimension and U is composed of both local and global terms. Hence, we propose an efficient exploration mechanism specific to our problem, which is based on successive merging operations between NURBS primitives from a good initial configuration, as illustrated in Figure 2.

This strategy is mostly motivated by the robustness and efficiency of recent planar shape detection methods to accurately approximate free-form shapes by piecewise-planar layouts. Such a planar patch configuration can indeed be considered as a good oversegmentation of the solution in which local geometry changes are likely to be captured by multiple planar patches. Starting from such a configuration, our exploration mechanism proposes successive merges between two adjacent patches, guided by a priority queue that ranks all possible merges according to a quality criterion. A suggested merge is then performed if the energy of the resulting configuration is lower than the current one. This

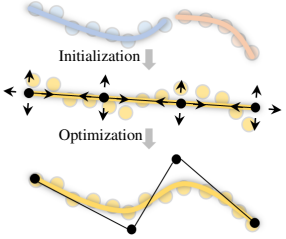
mechanism iterates these proposition-decision cycles until no proposition from the priority queue can decrease the energy anymore. The pseudocode of this mechanism is summarized in Algorithm 1. We now describe in detail each technical component of the mechanism.

Algorithm 1 Pseudocode of NURBSFIT

```
Initialize configuration x
Initialize adjacency graph G, energy U and priority queue Q
while Q ≠ ∅ do
    i ← pop(Q)
    Compute top merging operation i and energy U'
    if U' < U then
        Update configuration x by applying operation i
        Update G, U and Q
end while
Finalize the surface patches of configuration x
```

Initial configuration. The initial configuration $\mathbf{x}$ is based on the GoCoPP planar shape detection method [48]. This choice allows us to start with an oversegmentation of planar patches that (i) conforms to the local shape variations within a user-defined tolerance distance, (ii) facilitates the subsequent merging operations by using the planar patches to define the supporting plane of the NURBS control point grid, and (iii) is typically of good quality even for real-world, defect-laden input data. GoCoPP directly returns a configuration of the form $\mathbf{x} = (\mathcal{S}, \mathcal{L})$ defined in Section 3.1 with $\mathcal{S}$, the set of detected planes (which are a specific case of NURBS surfaces with co-planar control points) and $\mathcal{L}$, a set of labels that specifies whether each point is an inlier to one of the planes or an outlier. Given this initial configuration, we compute an initial adjacency graph $\mathcal{G}$ between the planar patches. In practice, two patches are adjacent if at least a pair of their respective inlier points are adjacent in the k-nearest neighbor graph of the input point cloud.

Merging operation. The merging of two patches constitutes the key atomic operation of our exploration mechanism, which we perform by regrouping the inliers of the two patches and then fitting a new NURBS surface to them. As illustrated in the inset, we first initialize the new NURBS surface by positioning its control points on the best-fitting plane to the merged inliers, i.e., the one obtained by Principal Component Analysis (middle row with control points represented by black dots). We organize them according to a regular grid whose borders correspond to the smallest rectangle containing the projected inliers in the plane. We then optimize the position of the control points with a variant of the learning-based model NURBSDiff [5] that we adapt to our unsupervised problem (bottom row of the inset). In particular, we replace the $\mathcal{L}_2$ loss between the control points with the Chamfer distance between the input point cloud and a set of points sampled from the NURBS surface. We also add a Laplacian loss term over the predicted control points to better regularize the loss function. We weight both loss terms by $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$ in $\mathcal{L} = \lambda_1 \mathcal{L}_{CD} + \lambda_2 \mathcal{L}_{Lap}$, and use gradient descent for optimization. More details on this revisited optimizer are given in supplementary material. In contrast to the squared distance minimization approach [43], our reformulation of NURBSDiff is relatively fast. Once fitted, we update the inliers and outliers by checking if they fall inside the fitting zone defined by ϵ, and repeat the fitting.



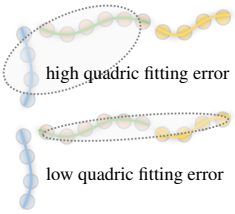
Priority queue. A priority queue Q sorts all possible merges between two patches that are adjacent in $\mathcal{G}$, in a descending quality order where the first is best. The quality criterion typically reflects the capacity of a merge to decrease the energy, the best merges being those with the strongest negative energy variation between the proposed configuration and the current one. This would require an exhaustive computation and evaluation of all possible merges in $\mathcal{G}$, which is unfortunately too costly. Instead, we use a cost-efficient quadric fitting analysis based on the principles of Taubin's normalized quadric fitting [42] to quickly predict the relevant merges. This is similar to Garland et al.'s mesh simplification [12] but using general quadric surfaces and not just plane quadrics. Recently, this has also shown potential to analyze 3D point clouds [29, 50].

The fitting error e of the best quadric surface to a group of points is computed from a generalized eigenvalue problem. The solution is found using a vector of quadric coefficients $\mathbf{c}$, a covariance matrix $\mathbf{M}$, and a gradient covariance matrix $\mathbf{N}$ of the points. The fitting error is computed as

$$e = \frac{\mathbf{c}^T \mathbf{M} \mathbf{c}}{\mathbf{c}^T \mathbf{N} \mathbf{c}}, \tag{5}$$

A key advantage of this method is that the covariance matrix and gradient matrix over a union of point sets is simply the sum of the individual matrices. This property is particularly appealing when groups of points are progressively merged, as it allows a fast error update.

We then organize the priority queue so that the possible merges in $\mathcal{G}$ are sorted from the lowest fitting error (Eq. 5) to the highest one. As illustrated in the inset, we favor merging between two patches that have a similar continuous geometry (bottom) and penalize those with distinct shapes separated by sharp creases (top). Experiments measuring the impact of this quadric fitting-based strategy are provided



in supplementary material, as well as details on the quadric fitting and error computation.

After assigning covariance and gradient covariance matrices to each initial patch of a configuration $\mathbf{x}$, we initialize the priority queue Q by computing the quadric fitting error of each possible merge (i.e., an edge of the adjacency graph $\mathcal{G}$) and sorting the merges by ascending error.

Updates. When a merging proposition at the top of the priority queue allows an energy decrease, the merging operation becomes effective and the configuration $\mathbf{x}$ is updated accordingly. Otherwise, the operation is discarded. The adjacency graph $\mathcal{G}$ is also updated by collapsing the edge associated with the merge. We finally update the priority queue Q by (i) removing all merges associated with one of the two former surface patches, (ii) recomputing the quadric fitting error of the merges affecting the new NURBS surface patch, and (iii) sorting these merges in the priority queue according to their error. Optionally, we can discard merges with a too high quadric-fitting error in order to converge faster.

Stopping criterion. The exploration mechanism stops when there are no proposed merges in the priority queue that can decrease energy.

Finalization. Once the exploration mechanism has converged, we refine the borders of each NURBS patch by removing parts that do not contain inlier points. More concretely, this trimming process first projects the inlier points into the UV-space of the NURBS surface and computes their boundary by Alpha-shape [8]. This alpha shape boundary is then intersected in the UV-space with the grid-based mesh of the NURBS patch via a constrained Delaunay triangulation. Only the triangle facets located inside the alpha shape are then kept. These 2D UV-triangles are then lifted back to 3D to form the refined mesh-based representation of the NURBS patch.

Optimality. Our exploration mechanism is local and does not guarantee to find the globally optimal solution. It relies upon the presence of a good initial configuration, which is a reasonable assumption given the capacity of GoCoPP to be robust to imperfect data and to approximate the local geometry while preserving sharp creases. In particular, Figure 3 illustrates this preservation capacity at various fitting tolerances, which is an important property to fit NURBS patches on object parts of varying size. Note that the use of a non-local optimization algorithm such as a Monte Carlo Tree Search [3] is also possible. In practice, however, such a strategy increases processing time by several orders of magnitude for a final configuration that is only marginally better.

4. Experiments

Implementation details. NURBSFIT has been implemented in Python. Our experiments were conducted on a standard desktop PC with an Intel Core i7-10700K CPU clocked at 3.8GHz with 32GB of RAM memory and an NVIDIA RTX 3090 GPU running Ubuntu. In all experiments, we fixed the number of control points per NURBS patches to a 4×4 grid and the tolerance of a fit ϵ to 1% of the bounding box diagonal to produce results with a level of detail similar to that of competitors. Experiments on the impact of grid size and fitting tolerance are provided in the supplementary material. The energy weights λ_f, λ_s and λ_c were set to 0.1, 0.4 and 0.5 respectively on synthetic data, and to 0.3, 0.4 and 0.3 on real-world data where fidelity is arguably more important than simplicity. For initialization, we used GoCoPP [48] in its default setting, except for the fitting parameter that we fixed as the value of ϵ for real-world data, and as twice its value for synthetic data. This choice is motivated by the fact that synthetic models are significantly simpler and require fewer initial patches.

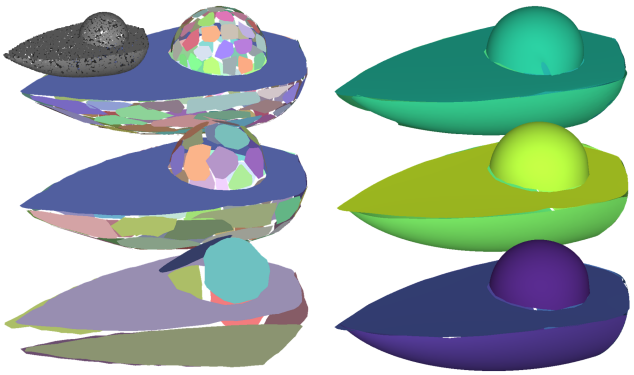


Figure 3. Initialization with configurations of varying complexity. Initial planar patches (left) partition the input point cloud (top left) while preserving the sharp creases, as we can see at the junction between the avocado stone and the slice or between the slice and the skin where no planar patch overlaps. This property is valid from fine (top) to coarse (bottom) initial configurations, which allows our algorithm to output consistent NURBS patches (right). Model from the ABC dataset.

Datasets. We evaluated our algorithm on the ABC [19] and KSR42 [2] datasets. The former is composed of simple CAD models from which we randomly selected 1,800 models to sample (synthetic) point clouds. The latter corresponds to 42 laser scans and point clouds generated by Multiview Stereo on more challenging objects and scenes. We excluded the irrelevant point clouds for our tasks, i.e., those describing piecewise planar structures such as indoor scenes, and kept the freeform objects only. For fair comparisons, we down-sampled these point clouds to 10k points so that they can be processed by the existing methods.

Evaluation metrics. We used five metrics to evaluate the quality of the fitting. Three of them directly relate to the

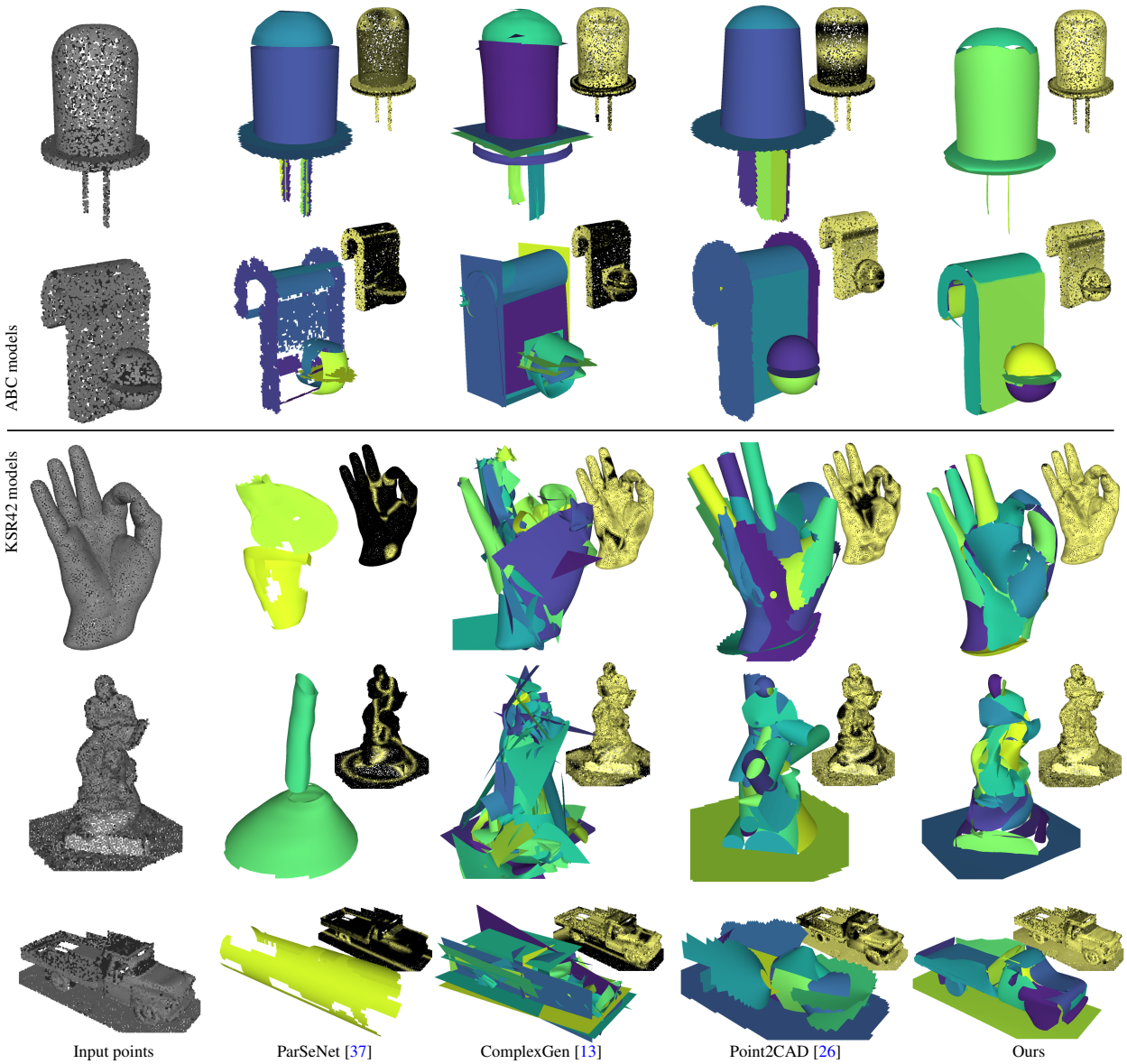


Figure 4. Visual comparisons on some synthetic (top) and real-world (bottom) models. Existing approaches suffer from various issues, especially on the real-world data: imprecise segmentation (ParSeNet), overly complex decomposition (ComplexGen, with the geometric refinement activated) and approximate fitting (Point2CAD). In contrast, our method produces simple and accurate configurations of primitives without generalization issues between synthetic and real world data. The geometric accuracy of the primitives are illustrated by the yellow-to-black point clouds showing the Chamfer distance from the input points to the output primitives (yellow: no error, black: $\geq 5\%$ of the bounding box diagonal). Note that the Point2CAD results use ParSeNet as input segmentation for the three synthetic models (top) and our segmentation for the real-world ones (bottom).

objectives of fidelity, simplicity and completeness of our energy minimization problem. The fidelity error $\mathcal{E}_{Fid}$ measures the mean distance between the output surface and the inliers, i.e., the input points located at a distance lower than ϵ from the output surface. The simplicity and completeness scores correspond to the number of output NURBS patches and the ratio of inliers, respectively. We also derived the P-coverage score used in [37] into two new metrics P^i_{cov} and P^o_{cov} to better distinguish between precision and recall with respect to input data. In fact, P^i_{cov} corresponds to the original P-coverage score [37] with an asymmetric Chamfer distance from input to output. This measure penalizes missing (or partially missing) primitives. In contrast, P^o_{cov} uses the asymmetric Chamfer distance from the output to the input so that primitives too large and complex return a low score. Detailed formulas of these scores are given in SM.

	ABC dataset [19]					*KSR42* dataset [2]				
	$\mathcal{E}_{Fid}$ (↓)	Compl. (↑)	P^i_{cov} (↑)	P^o_{cov} (↑)	$\lvert N\rvert$	$\mathcal{E}_{Fid}$ (↓)	Compl. (↑)	P^i_{cov} (↑)	P^o_{cov} (↑)	$\lvert N\rvert$
ParSeNet [37]	24.7	69.2	55.2	53.9	10.6	50.2	12.4	25.0	10.9	2.0
ComplexGen [13]	47.3	35.7	21.5	21.9	26.7	47.5	28.0	20.0	18.0	56.7
ComplexGen$^+$ [13]	29.5	72.1	33.3	43.3	26.7	33.4	74.9	32.6	40.9	56.7
Point2CAD [26]	**15.6**	**87.7**	54.1	<u>63.7</u>	10.6	48.6	25.1	22.7	19.4	2.0
Point2CAD* [26]	<u>23.7</u>	81.2	43.6	57.1	7.4	28.1	85.0	47.3	<u>63.2</u>	31.2
GoCoPP [48]	31.2	78.4	<u>64.1</u>	61.2	18.7	**13.4**	<u>91.7</u>	<u>58.1</u>	45.1	158.7
Ours	28.7	<u>85.1</u>	**74.7**	**69.3**	7.4	<u>25.6</u>	**94.9**	**88.9**	**79.0**	31.2

Table 1. Quantitative comparisons on *ABC* and *KSR42*. The scores are expressed in percent, except for $|N|$ that refers to the average number of primitives. The geometric refinement module is activated for ComplexGen$^+$ and deactivated for ComplexGen. Point2CAD and Point2CAD* use the point segmentation of ParSeNet and of our algorithm respectively. Bold and underlined values indicate the best and second best scores respectively.

Methods. We compared our algorithm to the primitive fitting methods ParSeNet [37], ComplexGen [13] which also refines primitives with mesh trimming operations, and Point2CAD [26] which combines quadric primitives with fine mesh patches. We used the authors' models pre-trained on the ABC dataset [19] under the same data split for ParSeNet and ComplexGen. For the latter, two variants were considered: with and without the activation of the geometric refinement module. For Point2CAD, we used the author's version with two variants of input segmentation: one given by ParSeNet, and one by our algorithm. To measure the gain with our initialization, we also included GoCoPP [48] as a comparative method.

Results. Table 1 presents quantitative results on the two datasets while Figure 4 shows the visual quality of the results on some representative models.

On the ABC dataset, our method exhibits a good capacity to describe objects accurately with a few NURBS primitives, with both the lowest average number of primitives and the highest asymetric P-coverage scores. ComplexGen [13] tends to produce an overly complex description of objects with redundant approximate primitives. The activation of the geometric refinement module helps primitives to be more accurately fitted to the input points but does not allow a reduction in the number of primitives. ParSeNet [37] returns more simple solutions that better cover input data than ComplexGen [13]. In particular, the input points are clustered more accurately, as shown in Figure 4 with more nicely decomposed objects. Point2CAD [26], when combined with ParSeNet segmentation, even improves the quality of the fit, both in terms of fidelity and completeness. This gain is partly explained by the higher expressiveness of the (nonparametric) mesh patches. Although geometrically accurate, Point2CAD often suffers from confusion on the primitive type, as shown in Figure 4 where the cylindrical part of the second top model is captured by a conical primitive. Since our primitives are NURBS patches, we are not affected by this issue, which typically degrades the P-coverage scores. Finally, it is interesting to note that our method strongly improves all the accuracy scores compared to the GoCoPP initialization, while reducing the number of primitives by more than half.

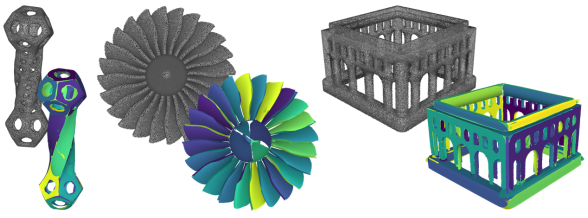


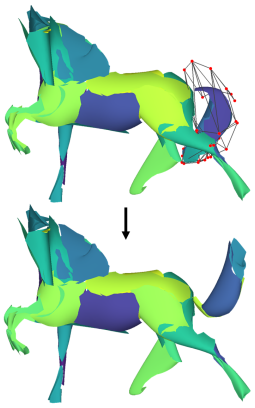
Figure 5. Visual results on Thingi10k models. Note in particular how each helice blade is represented by one NURBS patch in the middle model.

Our algorithm performs best on real-world point clouds generated by common acquisition systems. The gain is high for all the evaluation metrics, and particularly for the asymmetric P-coverage scores. Only ParSeNet [37] and Point2CAD [26] with ParSeNet input segmentation exhibit a lower average number of primitives, that is, 2.0: this excessively low number illustrates the inability of ParSeNet to correctly decompose complex unseen objects. Broadly speaking, learning-based methods generalize poorly to real-world point clouds in which the object decomposition into parts is more difficult. ComplexGen produces overly complex results with at least twice as many primitives as our results, while Point2CAD, even with our input segmentation, often fails to fit geometrically accurate primitives. The GoCoPP initialization exhibits good robustness to real-world data with, in particular, a low fidelity error. Our algorithm exploits this robustness property while consolidating completeness and P_{cov} scores by a large margin and reducing the number of primitives by a factor five.

Figure 5 shows that the unsupervised nature of our algorithm offers good generalization capacity on a wide variety of shapes, including CAD-based models from the Thingi10k dataset [51] which are significantly more complex than those of the ABC dataset.

Scalability. While the comparative study was done on smaller data (10k points to satisfy the conditions of the other methods), our algorithm works on much larger data. As shown in Figure 6, it can digest input points of several order magnitudes larger than the existing approaches and process complex initial configurations with sub-linear scalability behavior. Note that a high number of initial planar patches allows a more accurate description of the complex models. In contrast, a low number is usually preferable with simple models as it reduces processing time without impacting the output quality, as illustrated with the sphere model.

Application scenarios. Our primary objective in fitting NURBS is reverse CAD. To fully reach this goal, a robust assembly method is required to produce standard (connectivity-aware) CAD-based representations, not just a set of disconnected NURBS patches. We leave this challenging task for future work. Other potential uses of NURBSFIT include (i) editing shapes by intuitive manipulation of the NURBS control points, as exemplified in the inset where a few clicks allow us to modify the shape and orientation of the tail (control points are represented by red dots), (ii) segmenting point clouds or (iii) compressing 3D data with a lightweight representation made by a control point sequence.



Limitations. Our algorithm has some shortcomings. Firstly, in our implementation, the number of control points per NURBS patch is fixed. Ideally, this number should vary according to the local geometry of the shape. However, adjusting the degree of NURBS without degrading the performance of the exploration mechanism is a difficult challenge. Secondly, our NURBS patches are necessarily open surfaces. Thus, a closed revolution surface is typically represented by one NURBS patch with disconnected borders or two NURBS patches, as illustrated in Figure 6 (see the right sphere). Third, while our algorithm is scalable and can digest large point clouds, it remains relatively slow, requiring typically a few minutes to process hundreds of thousand input points. For large point clouds, it is therefore recommended to start from a coarse planar patch configuration to reduce the number of merging operations. Note that the experiments were conducted with a basic sequential implementation that we did not optimize with a parallelization of the geometric operations.

5. Conclusion

We presented NURBSFIT, an unsupervised method for fitting NURBS surfaces to 3D point clouds. Our algorithm consists of a progressive merging of pairs of adjacent NURBS patches that is guided by a cost-efficient quadric fitting analysis and controlled by an energy measuring the quality of the NURBS patch configurations. We showed the robustness and scalability of our algorithm on both synthetic and real-world data, as well as its competitiveness against existing approaches, especially from imperfect point clouds generated by standard acquisition systems. This opens the door to designing alternative strategies to deep learning reverse CAD methods that mostly operate on simple CAD models from synthetic point clouds only.

In future work, we will investigate how to adjust the number of control points per NURBS patches in an efficient manner, potentially using a scale-space analysis of the input data. We will also work on the assembly of our NURBS patches into explicit CAD representations. Traditionally performed by error-prone trimming operations, we will investigate a more robust strategy via the construction of a space-partitioning data structure that conforms to the NURBS surface patches.

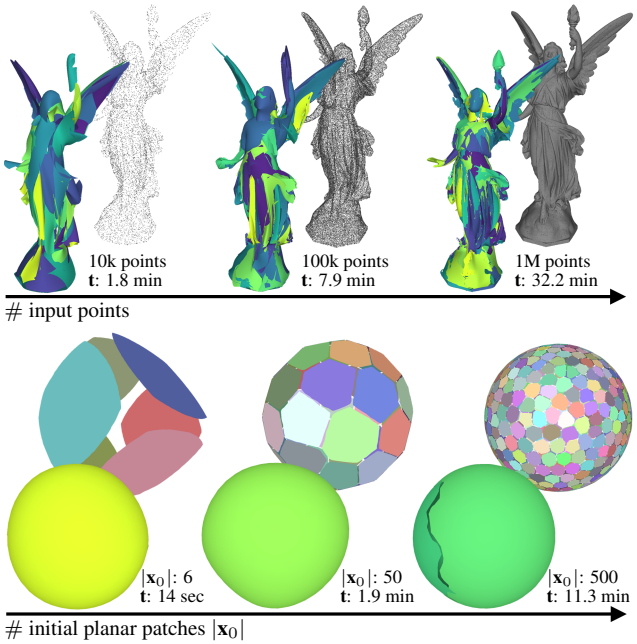


Figure 6. Scalability analysis. The size of input data (top) and the number of initial planar patches (bottom) have a direct impact on the performances of our algorithm. A similar number of initial patches were used in the top experiment whereas the same 50K input points sampled on a sphere were used for the bottom one. **t** refers to the total processing time (including initialization).

Acknkowledgments

This work was partially supported by the European Union's H2020 research and innovation program under grant agreements 813170 (EVOCATION).

References

[1] Seok-Hyung Bae and Byoung Choi. Nurbs surface fitting using orthogonal coordinate transform for rapid product development. *Computer Aided Design*, 34, 2002. 2

[2] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *Trans. on Graphics*, 39(5), 2020. 2, 5, 7

[3] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. on Computational Intelligence and AI in Games*, 4(1), 2012. 5

[4] Gianmarco Cherchi, Marco Livesu, Riccardo Scateni, and Marco Attene. Fast and robust mesh arrangements using floating-point arithmetic. *Trans. on Graphics*, 39(6), 2020. 2

[5] Anjana Deva Prasad, Aditya Balu, Harshil Shah, Soumik Sarkar, Chinmay Hegde, and Adarsh Krishnamurthy. Nurbsdiff: A differentiable programming module for nurbs. *Computer-Aided Design*, 146, 2022. 4

[6] Elona Dupont, Kseniya Cherenkova, Dimitrios Mallis, Gleb Gusev, Anis Kacem, and Djamila Aouada. Transcad: A hierarchical transformer for cad sequence inference from point clouds. In *ECCV*, 2024. 1, 2

[7] Matthias Eck and Hugues Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *SIGGRAPH*, 1996. 2

[8] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. on Information Theory*, 29(4), 1983. 5

[9] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar Shape Detection at Structural Scales. In *CVPR*, 2018. 2

[10] Rao Fu, Qian Li, Cheng Wen, Ning An, and Fulin Tang. A novel framework for learning bézier decomposition from 3d point clouds. *IEEE Trans. on Circuits and Systems for Video Technology*, 2024. 1

[11] Jun Gao, Chengcheng Tang, Vignesh Ganapathi-Subramanian, Jiahui Huang, Hao Su, and Leonidas J. Guibas. Deepspline: Data-driven reconstruction of parametric curves and surfaces, 2019. 2

[12] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of SIGGRAPH*, 1997. 4

[13] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *Trans. on Graphics*, 41(4), 2022. 2, 6, 7

[14] Xin He, Chenlei Lv, Pengdi Huang, and Hui Huang. Windpoly: Polygonal mesh reconstruction via winding numbers. In *ECCV*, 2024. 2

[15] Jingwei Huang, Yanfeng Zhang, and Mingwei Sun. Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In *ICCV*, 2021. 2

[16] Jingen Jiang, Mingyang Zhao, Shiqing Xin, Yanchao Yang, Hanxiao Wang, Xiaohong Jia, and Dong-Ming Yan. Structure-aware surface reconstruction via primitive assembly. In *ICCV*, 2023. 2

[17] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Computer Graphics Forum*, 37, 2018. 2

[18] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *Trans. on Graphics*, 36(4), 2017. 1

[19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, 2019. 2, 5, 7

[20] Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models . In *CVPR*, 2021. 2

[21] Thibault Lejemble, Claudio Mura, Loïc Barthe, and Nicolas Mellado. Persistence analysis of multi-scale planar structure graph in point clouds. *Computer Graphics Forum*, 39(2), 2020. 2

[22] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *CVPR*, 2019. 1, 2

[23] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. *Trans. on Graphics*, 2011. 2

[24] Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for primitive fitting of point clouds. In *SIGGRAPH Conference*, 2023. 1, 2

[25] Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Split-and-Fit: Learning B-Reps via structure-aware Voronoi partitioning. *Trans. on Graphics*, 43 (4), 2024. 2

[26] Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad models from 3d point clouds. In *CVPR*, 2024. 1, 2, 6, 7

[27] Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J. Mitra. Cpfn: Cascaded primitive fitting networks for high-resolution point clouds. In *ICCV*, 2021. 1, 2

[28] David Marshall, Gabor Lukacs, and Ralph Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *PAMI*, 23(3), 2001. 2

[29] Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. PoNQ: a Neural QEM-based Mesh Representation. In *CVPR*, 2024. 4

[30] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *Trans. on Graphics*, 34(4), 2015. 2

[31] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *ICCV*, 2017. 2

[32] Sven Oesau, Yannick Verdie, Clément Jamin, Pierre Alliez, Florent Lafarge, Simon Giraudot, Thien Hoang, and Dmitry Anisimov. Point set shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edition, 2021. 2

[33] Les Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag, 1995. 1, 3

[34] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *ICCV*, 2021. 2

[35] Tsahi Saporta and Andrei Sharf. Unsupervised recursive deep fitting of 3d primitives to points. *Computers & Graphics*, 102, 2022. 2

[36] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer graphics forum*, 26(2), 2007. 2

[37] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Chaudhuri, and Radomyr Mech. Parsenet: A parametric surface fitting network for 3d point clouds. In *ECCV*, 2020. 1, 2, 6, 7

[38] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Neural Shape Parsers for Constructive Solid Geometry . *PAMI*, 44(05), 2022. 2

[39] Zeyu Shen, Mingyang Zhao, Dong-Ming Yan, and Wencheng Wang. Mesh2brep: B-rep reconstruction via robust primitive fitting and intersection-aware constraints. *IEEE Trans. on Visualization and Computer Graphics*, 2025. 2

[40] R. Sulzer and Florent Lafarge. Concise Plane Arrangements for Low-Poly Surface and Volume Modelling. In *ECCV*, 2024. 2

[41] Bo Sun and Philippos Mordohai. Oriented point sampling for plane detection in unorganized point clouds. In *ICRA*, 2019. 2

[42] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *PAMI*, 13 (11), 1991. 4

[43] Wenping Wang, Helmut Pottmann, and Yang Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *Trans. on Graphics*, 25(2), 2006. 4

[44] Karl D.D. Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints . In *CVPR*, 2022. 2

[45] Xiao Xiao, Pierre Alliez, and Laurent Rineau Laurent Busé. Delaunay meshing and repairing of nurbs models. *Computer Graphics Forum*, 40(5), 2021. 2

[46] Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. Hpnet: Deep primitive segmentation using hybrid representations. In *ICCV*, 2021. 1, 2

[47] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *CVPR*, 2022. 1, 2

[48] Mulin Yu and Florent Lafarge. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. In *CVPR*, 2022. 2, 4, 5, 7

[49] Mehmet Ersin Yumer and Levent Burak Kara. Surface creation on unstructured point sets using neural networks. *Computer Aided Design*, 44(7), 2012. 2

[50] Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, and Pierre Alliez. Variational Shape Reconstruction via Quadric Error Metrics. In *SIGGRAPH*, 2023. 4

[51] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *arXiv:1605.04797*, 2016. 7