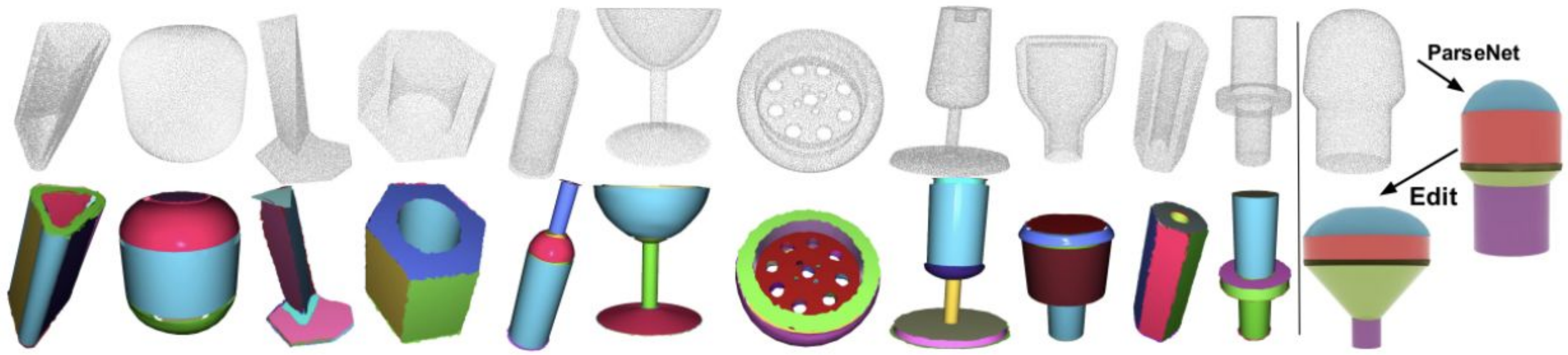
ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds

Ende Shen

Brief Summarization

- End-to-end differentiable deep network,
- 3D point cloud => parametric surface patches (B-spline & basic geometric primitives)



ParSeNet decomposes point clouds (top row) into collections of seamlessly assembled parametric surface patches including B-spline patches (bottom row). On the right, a shape is edited using the inferred parametrization.

Motivation

- 3D point clouds can be rapidly acquired using 3D sensors
- In computer-aided design and modeling, designers often model shapes by constructing several non-overlapping patches placed seamlessly
- Previous work lack the expressivity of multiple patches of B-spline

Pipeline

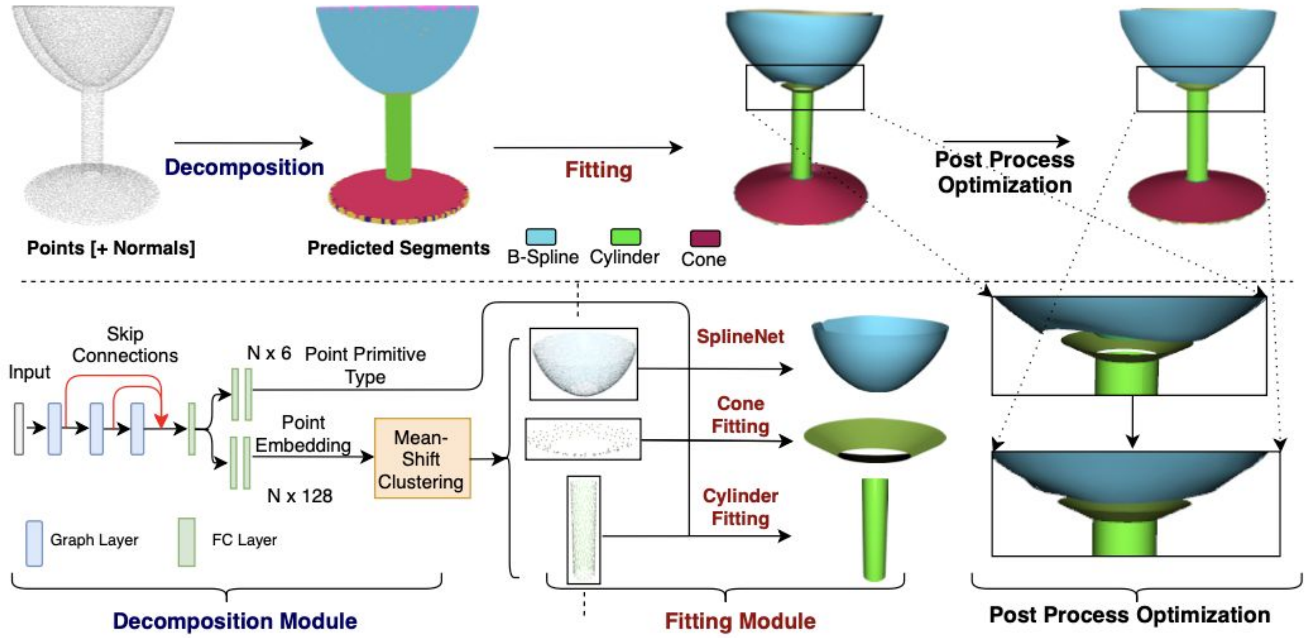
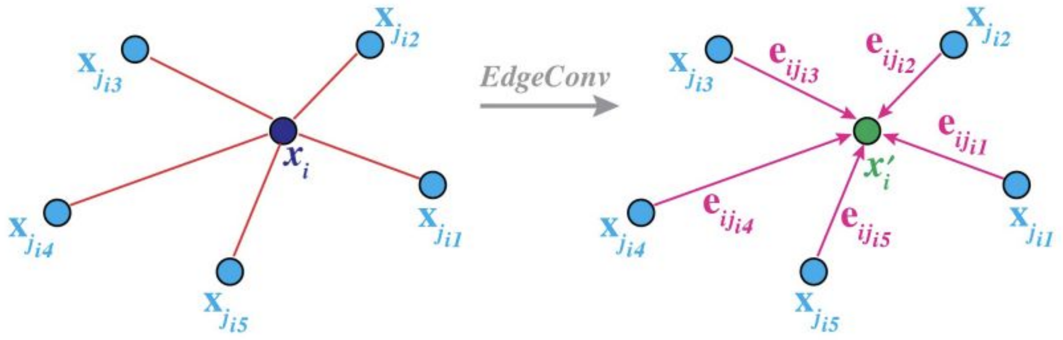


Fig. 2: **Overview of ParSeNet pipeline.** (1) The *decomposition module* (Section 3.1) takes a 3D point cloud (with optional normals) and decomposes it into segments labeled by primitive type. (2) The *fitting module* (Section 3.2) predicts parameters of a primitive that best approximates each segment. It includes a novel SplineNet to fit B-spline patches. The two modules are jointly trained end-to-end. An optional postprocess module (Section 3.3) refines the output.

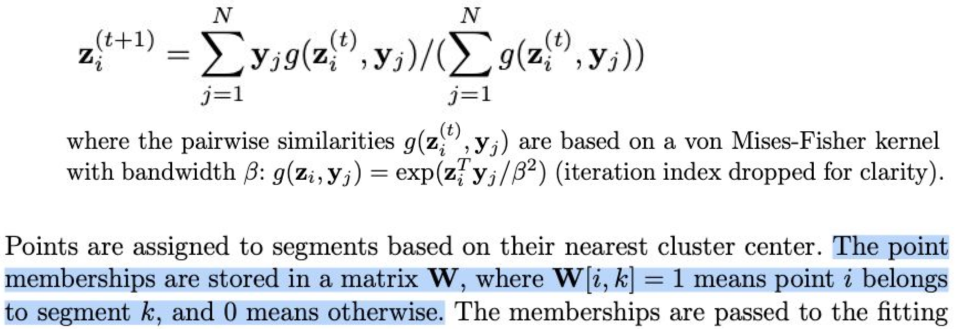
Decomposition module

- Embedding network.
  - edge convolution layers (EdgeConv) from DGCNN
  - neighborhoods are dynamically defined via nearest neighbors
  - stack 3 EdgeConv layers, each extracting a 256-D representation per point, also extract a global 1024-D representation for the whole point cloud
  - Passes through FC & Relu then normalized to unit length (128-D)



Decomposition module

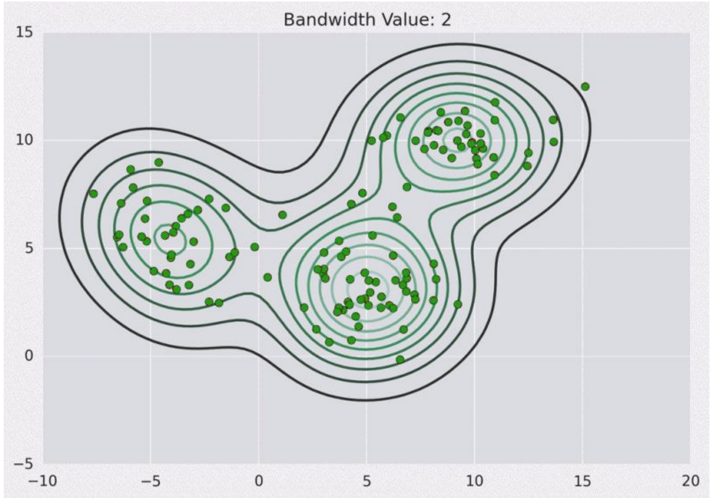
- Mean-shift clustering
  - Advantage: does not require the target number of clusters as input
  - “A kernel is a fancy mathematical word for a weighting function generally used in convolution. There are many different types of kernels, but the most popular one is the Gaussian kernel. Adding up all of the individual kernels generates a probability surface example density function. Depending on the kernel bandwidth parameter used, the resultant density function will vary.”



$$\mathbf{z}_i^{(t+1)} = \sum_{j=1}^{N} \mathbf{y}_j g(\mathbf{z}_i^{(t)}, \mathbf{y}_j) / (\sum_{j=1}^{N} g(\mathbf{z}_i^{(t)}, \mathbf{y}_j))$$

where the pairwise similarities $g(\mathbf{z}_i^{(t)}, \mathbf{y}_j)$ are based on a von Mises-Fisher kernel with bandwidth β: $g(\mathbf{z}_i, \mathbf{y}_j) = \exp(\mathbf{z}_i^T \mathbf{y}_j / \beta^2)$ (iteration index dropped for clarity).

Points are assigned to segments based on their nearest cluster center. The point memberships are stored in a matrix $\mathbf{W}$, where $\mathbf{W}[i,k] = 1$ means point i belongs to segment k, and 0 means otherwise. The memberships are passed to the fitting



Decomposition module

- Segment Classification
  - From per-point representation to FC layers and ReLUs, then softmax for a per-point probability to determine patch type
  - Then majority voting over the cluster points

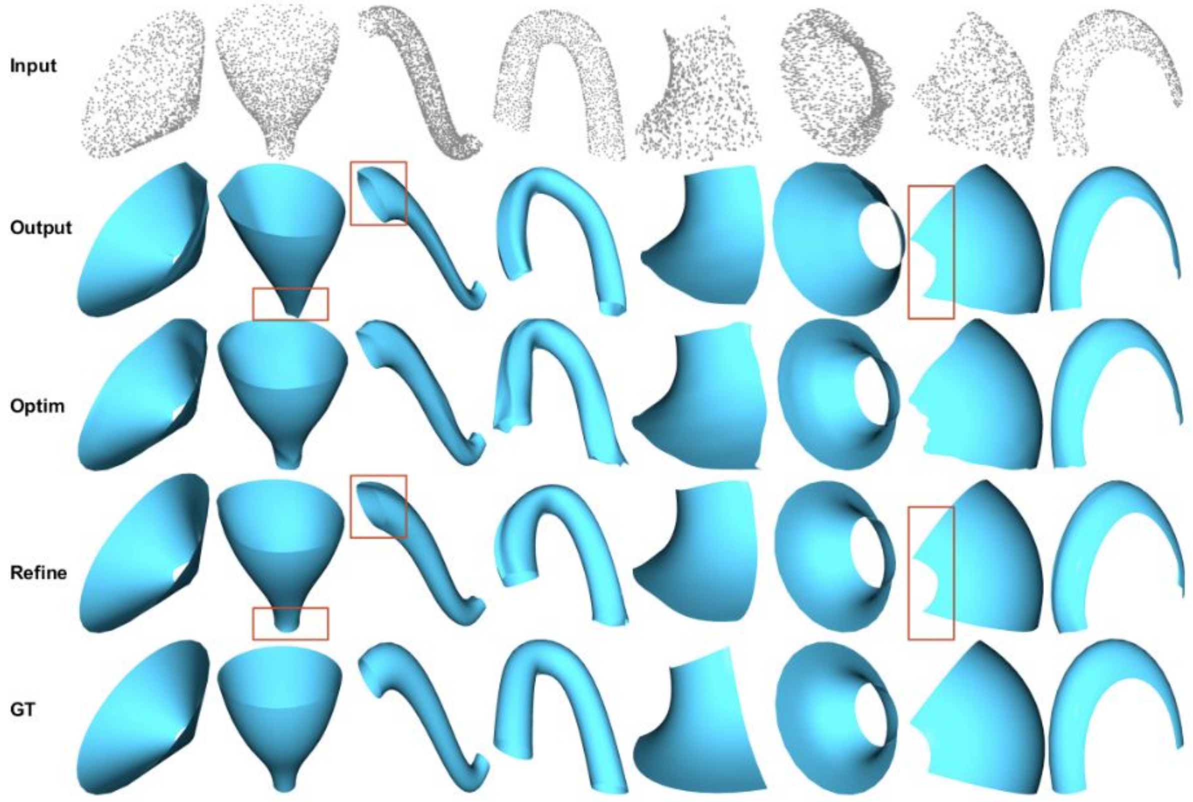
Fitting module

- Basic primitives
  - Least squares fitting: center and radius for spheres; normal and offset for planes; center, direction and radius for cylinders; and apex, direction and angle for cones
- B-Splines
  - We propose a neural network SplineNet, that inputs points of a segment, and outputs a fixed size control-point grid.

$$\boldsymbol{\phi}_k = \max_{i=1\ldots N} (\mathbf{W}[i,k] \cdot \boldsymbol{\phi}_i).$$

  - From theta, to two FC Layers with ReLU to 20x20 control points (unrolled into a 1200-D output vector), trick: if number of points is small for a segment, upsample (nearest neighbor) to 1600

Some results



Post-processing module

However, patches might not entirely cover the input point cloud, and boundaries between patches are not necessarily well-aligned. Further, the resolution of the initial control point grid (20×20) can be further adjusted to match the desired surface resolution.

Optimization: create a grid of 40×40 points, tessellate into quads, maximal matching between the quad vertices and the input points of the segment, using the Hungarian algorithm with L2 distance costs. then perform an as-rigid-as-possible (ARAP) deformation, selected vertices (pivots) achieve targets position, while promoting locally rigid transformations in one-ring neighborhoods

Post-processing module

Refinement of B-spline control points.

After the above optimization, we again perform a maximal matching between the quad vertices and the input points of the segment. As a result, the input segment points acquire 2D parameter values in the patch’s UV parameter space, which can be used to re-fit any other grid of control points [15]. In our case, we iteratively upsample the control point grid by a factor of 2 until a fitting tolerance, measured via Chamfer distance, is achieved.

Training - Loss functions

1. Embedding loss. Given a triplet of points (a, b, c), we use the triplet loss to learn the embeddings

$$L_{emb} = \sum_{\mathcal{S} \in \mathcal{D}} \frac{1}{|\mathcal{T}_{\mathcal{S}}|} \sum_{(a,b,c) \in \mathcal{T}_{\mathcal{S}}} \ell_{emb}(a, b, c).$$

What is triplet loss?

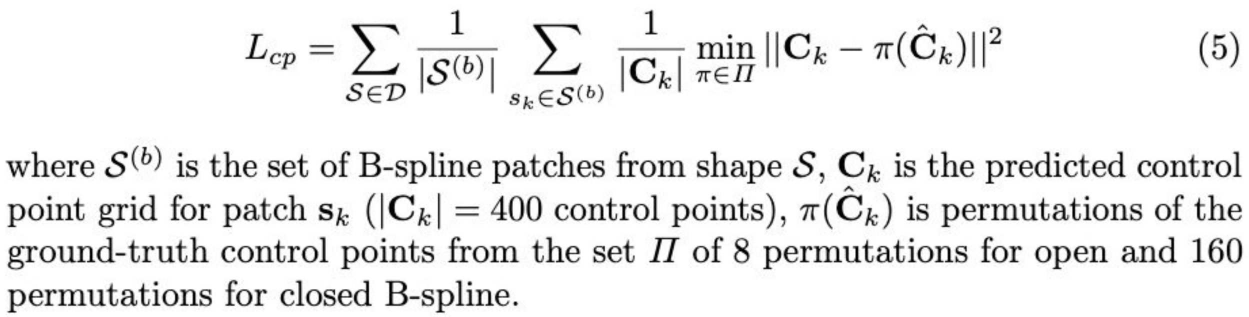
$$\mathcal{L}(A, P, N) = \max\left(\| \mathrm{f}(A) - \mathrm{f}(P) \|^2 - \| \mathrm{f}(A) - \mathrm{f}(N) \|^2 + \alpha, 0 \right)$$

where A is an *anchor input*, P is a *positive input* of the same class as A, N is a *negative input* of a different class from A, α is a margin between positive and negative pairs, and f is an embedding.

Training - Loss functions

2. Segment classification loss: cross entropy loss

3. Control point regression loss. used to train SplineNet. Note: reconstruction loss should be invariant to flips or swaps of control points grid in u and v directions.



$$L_{cp} = \sum_{\mathcal{S} \in \mathcal{D}} \frac{1}{|\mathcal{S}^{(b)}|} \sum_{s_k \in \mathcal{S}^{(b)}} \frac{1}{|\mathbf{C}_k|} \min_{\pi \in \Pi} ||\mathbf{C}_k - \pi(\hat{\mathbf{C}}_k)||^2 \tag{5}$$

where $\mathcal{S}^{(b)}$ is the set of B-spline patches from shape $\mathcal{S}$, $\mathbf{C}_k$ is the predicted control point grid for patch $\mathbf{s}_k$ ($|\mathbf{C}_k| = 400$ control points), $\pi(\hat{\mathbf{C}}_k)$ is permutations of the ground-truth control points from the set Π of 8 permutations for open and 160 permutations for closed B-spline.

Training - Loss functions

4. Laplacian loss. specific to B-Splines using SplineNet

$$L_{lap} = \sum_{\mathcal{S}\in\mathcal{D}} \frac{1}{|\mathcal{S}^{(b)}| \cdot M} \sum_{\mathbf{s}_k \in \mathcal{S}^{(b)}} \sum_{\mathbf{r}_n \in s_k} ||\mathcal{L}(\mathbf{r}_n) - \mathcal{L}(\hat{\mathbf{r}}_m)||^2 \tag{6}$$

where $\mathcal{L}(\cdot)$ is the Laplace operator on patch points, and $M = 1600$ point samples.

5. Patch distance loss.

$$L_{dist} = \sum_{\mathcal{S}\in\mathcal{D}} \frac{1}{K_{\mathcal{S}}} \sum_{k=1}^{K_{\mathcal{S}}} \frac{1}{M_{\hat{\mathbf{s}}_k}} \sum_{n \in \hat{\mathbf{s}}_k} D^2(\mathbf{r}_n, \mathbf{s}_k), \tag{7}$$

where $K_{\mathcal{S}}$ is the number of predicted patches for shape $\mathcal{S}$, $M_{\hat{\mathbf{s}}_k}$ is number of sampled points $\mathbf{r}_n$ from ground patch $\hat{\mathbf{s}}_k$, $D^2(\mathbf{r}_n, \mathbf{s}_k)$ is the squared distance from $\mathbf{r}_n$ to the predicted primitive patch surface $\mathbf{s}_k$. These distances can be computed analytically for basic primitives [12]. For B-splines, we use an approximation based on Chamfer distance between sample points.

Training - Training procedure

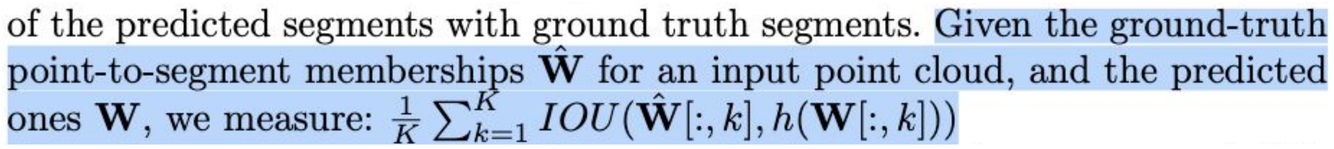
We first pre-train the networks of the decomposition module, using Lemb + Lclass.

then pre-train the SplineNet using SplineDataset for control point prediction exclusively on B-spline patches using Lcp + Llap + Ldist.

We then jointly train the decomposition and fitting module end-to-end with all the losses. To allow backpropagation from the primitives and B-splines fitting to the embedding network, the mean shift clustering is implemented as a recurrent module

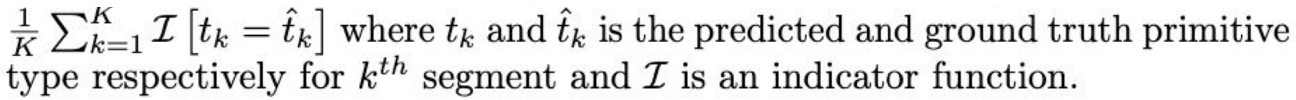
Experiments - Evaluation

Segmentation mean IOU



of the predicted segments with ground truth segments. Given the ground-truth point-to-segment memberships $\hat{\mathbf{W}}$ for an input point cloud, and the predicted ones $\mathbf{W}$, we measure: $\frac{1}{K}\sum_{k=1}^{K} IOU(\hat{\mathbf{W}}[:,k], h(\mathbf{W}[:,k]))$

Segment labeling IOU



$\frac{1}{K}\sum_{k=1}^{K}\mathcal{I}\left[t_k = \hat{t}_k\right]$ where t_k and $\hat{t}_k$ is the predicted and ground truth primitive type respectively for k^{th} segment and $\mathcal{I}$ is an indicator function.

Experiments - Evaluation

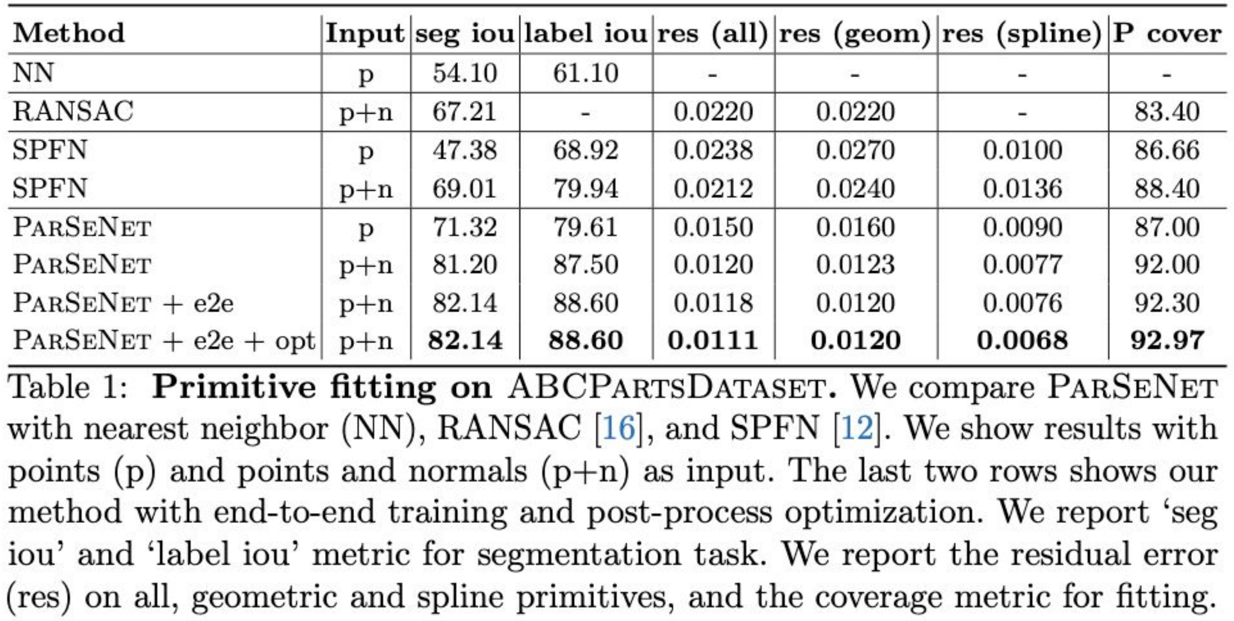
Residual error: measures the average distance of input points from the predicted primitives following

predicted primitives following [12]: $L_{dist} = \sum_{k=1}^{K} \frac{1}{M_k} \sum_{n \in \hat{\mathbf{s}}_k} D(\mathbf{r}_n, \mathbf{s}_k)$ where K is the number of segments, M_k is number of sampled points $\mathbf{r}_n$ from ground patch $\hat{\mathbf{s}}_k$, $D(\mathbf{r}_n, \mathbf{s}_k)$ is the distance of $\mathbf{r}_n$ from predicted primitive patch $\mathbf{s}_k$.

P-coverage: the coverage of predicted surface by the input surface

$$\frac{1}{N} \sum_{i=1}^{N} \mathbf{I} \left[\min_{k=1}^{K} D(\mathbf{p}_i, \mathbf{s}_k) < \epsilon \right] \; (\epsilon = 0.01).$$

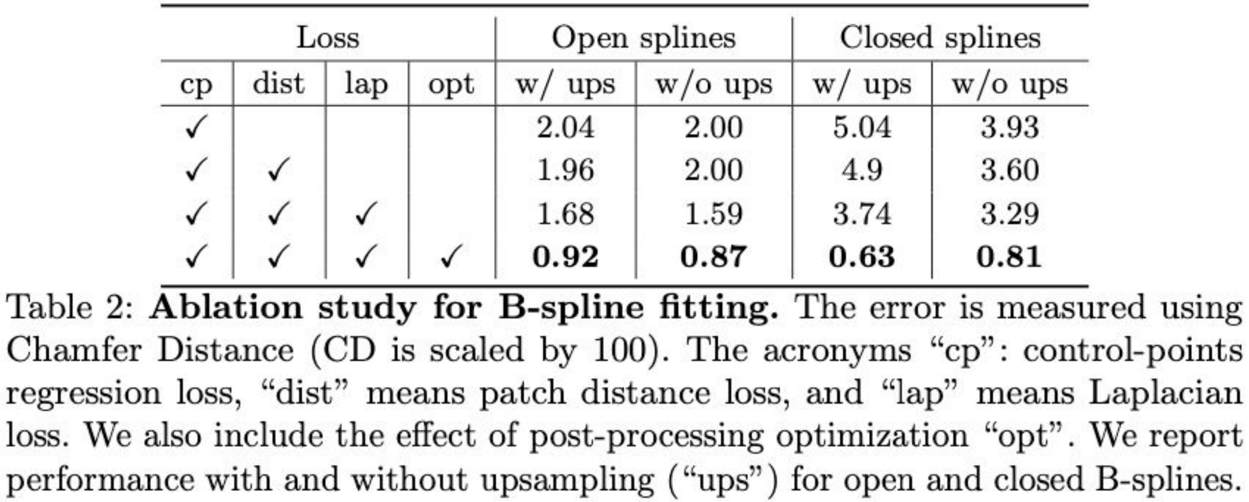
Experiments - Against other methods



Method	Input	seg iou	label iou	res (all)	res (geom)	res (spline)	P cover
NN	p	54.10	61.10	-	-	-	-
RANSAC	p+n	67.21	-	0.0220	0.0220	-	83.40
SPFN	p	47.38	68.92	0.0238	0.0270	0.0100	86.66
SPFN	p+n	69.01	79.94	0.0212	0.0240	0.0136	88.40
ParSeNet	p	71.32	79.61	0.0150	0.0160	0.0090	87.00
ParSeNet	p+n	81.20	87.50	0.0120	0.0123	0.0077	92.00
ParSeNet + e2e	p+n	82.14	88.60	0.0118	0.0120	0.0076	92.30
ParSeNet + e2e + opt	p+n	**82.14**	**88.60**	**0.0111**	**0.0120**	**0.0068**	**92.97**

Table 1: **Primitive fitting on** ABCPartsDataset**.** We compare ParSeNet with nearest neighbor (NN), RANSAC [16], and SPFN [12]. We show results with points (p) and points and normals (p+n) as input. The last two rows shows our method with end-to-end training and post-process optimization. We report 'seg iou' and 'label iou' metric for segmentation task. We report the residual error (res) on all, geometric and spline primitives, and the coverage metric for fitting.

Experiments - Ablation



Loss				Open splines		Closed splines	
cp	dist	lap	opt	w/ ups	w/o ups	w/ ups	w/o ups
✓				2.04	2.00	5.04	3.93
✓	✓			1.96	2.00	4.9	3.60
✓	✓	✓		1.68	1.59	3.74	3.29
✓	✓	✓	✓	**0.92**	**0.87**	**0.63**	**0.81**

Table 2: **Ablation study for B-spline fitting.** The error is measured using Chamfer Distance (CD is scaled by 100). The acronyms "cp": control-points regression loss, "dist" means patch distance loss, and "lap" means Laplacian loss. We also include the effect of post-processing optimization "opt". We report performance with and without upsampling ("ups") for open and closed B-splines.

Experiments - Qualitative Results

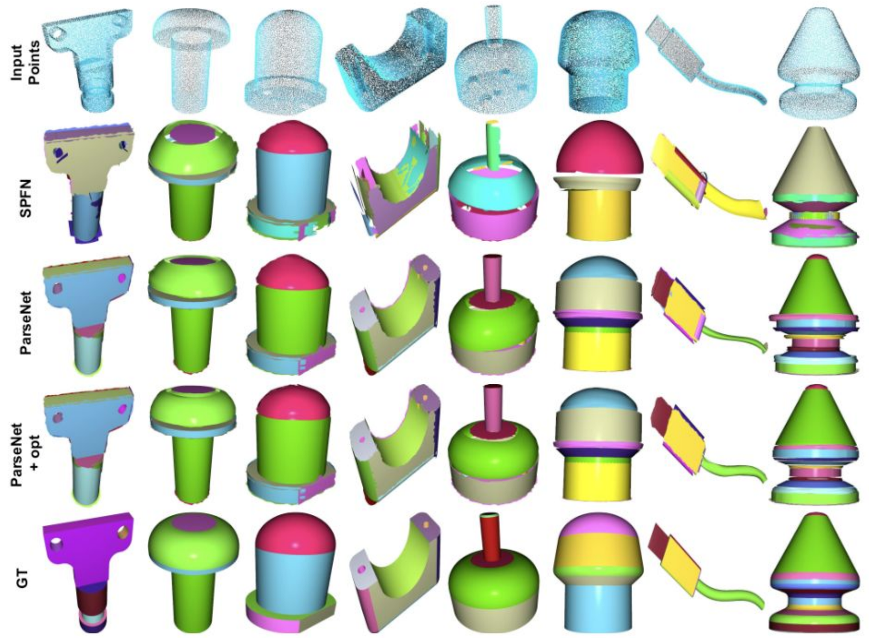


Fig. 4: Given the input point clouds with normals of the first row, we show surfaces produced by SPFN [12] (second row), PARSENET without post-processing optimization (third row), and full PARSENET including optimization (fourth row). The last row shows the ground-truth surfaces from our ABCPARTSDATASET.

Conclusion

- marries 3D deep learning with CAD modeling practices. Modelers can refine our results based on standard CAD modeling operations.
- Limitations: our method often makes mistakes for small parts, mainly because clustering merges them with bigger patches. In high-curvature areas, due to sparse sampling, ParSeNet may produce more segments than ground-truth. Producing seamless boundaries is still a challenge due to noise and sparsity in our point sets.