

SepicNet: Sharp Edges Recovery by Parametric Inference of Curves in 3D Shapes

Kseniya Cherenkova^{*†}

kseniya.cherenkova@uni.lu

Ilya Arzhannikov[†]

iarzhannikov@artec3d.com

Elona Dupont^{*}

elona.dupont@uni.lu

Gleb Gusev[†]

gleb@artec3d.com

Anis Kacem^{*}

anis.kacem@uni.lu

Djamila Auoada^{*}

djamila.aouada@uni.lu

^{*}SnT, University of Luxembourg

[†]Artec 3D

Abstract

3D scanning as a technique to digitize objects in reality and create their 3D models, is used in many fields and areas. Though the quality of 3D scans depends on the technical characteristics of the 3D scanner, the common drawback is the smoothing of fine details, or the edges of an object. We introduce SepicNet, a novel deep network for the detection and parametrization of sharp edges in 3D shapes as primitive curves. To make the network end-to-end trainable, we formulate the curve fitting in a differentiable manner. We develop an adaptive point cloud sampling technique that captures the sharp features better than uniform sampling. The experiments were conducted on a newly introduced large-scale dataset of 50k 3D scans, where the sharp edge annotations were extracted from their parametric CAD models, and demonstrate significant improvement over state-of-the-art methods.

1. Introduction

Today’s high-precision 3D scanning technologies help to obtain extremely realistic 3D models of real objects, which can be used in various applications, e.g., in 3D design and modeling [4, 8, 20, 21], for reverse engineering purposes in the context of Computer-Aided Design (CAD) [3, 6]. One of the major issues encountered in the process of scanning the object, is the sharpness degradation effect. For instance, 3D scanning tends to smooth the high-level geometrical details of real objects such as sharp edges. Depending on the quality of a 3D scanner, this effect is less or more prominent.

In this work, we propose to recover sharp features from a 3D reconstructed model in a data-driven manner by parametric inference of sharp edges of a 3D model. We de-

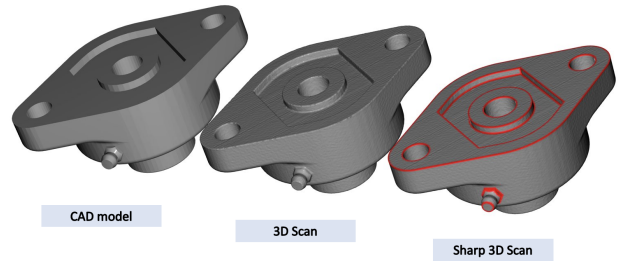


Figure 1. CAD model vs. scanned 3D model vs. sharp recovered 3D model and the detected sharp edges marked in red color.

velop an end-to-end trainable network architecture called SepicNet (Sharp Edges recovery by Parametric Inference of Curves). SepicNet consists of two main modules. The first module takes as input a sampled 3D point cloud, detects sharp edge points and groups them into different segments with predicted primitive types per segment among line, circle, and b-spline. The second module estimates the corresponding parameters of the identified primitives throughout a differentiable fitting. As the output of our network we have a set of parametrized sharp edges, labeled on a point cloud, that are further used to recover the sharp edges of a corresponding 3D scan by reprojecting nearby mesh vertices on parametrized edges. Considering the feature intense nature of 3D models, we also propose a novel adaptive point sampling scheme designed to approximate their geometry better than the standard uniform sampling. The design of SepicNet is substantially novel in its effort to incorporate the priors learnt from a vast collection of corresponding pairs of CAD models and their 3D scanned counterparts.

Our contributions. We develop a modular SepicNet ar-

chitecture for parametric inference of sharp edges from 3D scans of objects. We propose an edge points consolidation approach based on a distance field to closest sharp edge estimation. To account for topological complexity of an object we introduce an adaptive sampling technique for high-resolution details based on principal curvatures of model’s surface. We outsource a large-scale CC3D-PSE dataset with pairs of CAD models and their virtually scanned counterparts annotated with sharp edge parametrizations as basic curves: lines, circles and b-splines.

2. Related Work

Scanned 3D point sets are irregular and non-uniform, and need to be consolidated to enhance the surface reconstruction quality. One possible solution is to introduce edge-awareness in the consolidation of point sets in a data-driven manner. The EC-NET [25] network processes patches of points and learns to consolidate points using an edge-aware joint loss function when learning from the data. The performance of the model was demonstrated on a very limited set of 12 manually labelled scans. Another way is to introduce the a-priors with the objective to preserve sharp features for surface reconstruction methods as done in DeepPrior [24]. In both cases, the presence of high-frequency features and noise in the input scanned data, makes it extremely challenging to recover the sharpness of the scans.

The inference of edges as parametric curves is an alternative that arises directly from CAD surface parametrization as boundary-representation (b-rep). Following this direction, the PC2WF model [12] infers a wireframe of linear edges from a point cloud based on a vertex localization and an edge detector that identifies the pairs of vertices connected with an edge. The work [14] proposes a parametric approach to extract a wireframe based on an estimated scalar distance field DEF [15] that represents the proximity to the nearest sharp feature curve. PIE-Net [23] proposes to jointly detect edge and corner points, after which a curve proposal module generates an over-complete collection of curves that are further ranked.

The performance of all the above learning-based mentioned methods is heavily dependent on the variance of 3D data used for training. Several related datasets have been collected by the community. For instance, EC-Net [25] used a subset of data from ShapeNet dataset [1], which totals to 3M shapes with semantic and category annotations. Despite its large size, ShapeNet dataset does not offer any information about the edges of CAD models. The ABC dataset [9] used in PIE-Net [23], is a collection of 1M parametric CAD models, where the edge sharpness information can be extracted from surface patches smoothness labels. The availability of CAD models in this dataset makes it appealing for parametric sharp edge inference, but the scanned 3D counterparts are missing. More recently, the CC3D dataset [2]

has been proposed to bring the ongoing research on to a real-world scenario. It consists of more than 50k pairs of scanned 3D shapes and CAD models unrestricted to any category with varying complexity. In our work, we exploited the information from the CAD models in CC3D dataset to extract and parametrize ground truth sharp edges of each scan.

3. Method

The proposed approach aims to recover sharp edges in a 3D shape with smooth edges as depicted in Figure 1, approximating the ones of the CAD model. These sharp edges are learned from the pairing of 3D scans and CAD models.

A 3D scan χ provided as a triangular mesh is approximated by an intermediate representation as a point cloud $\mathbf{X} = \{\mathbf{x}_j\} \in \mathbb{R}^{N \times 3}$, where N denotes the number of points. The edge points on the scans are assigned based on the distance threshold τ to the closest edge in the CAD model. Ground truth sharp edges are extracted from CAD models by computing the deviations of normals between two neighboring surface patches and considering edges above the selected threshold to be sharp. The resulting set of sharp edges can be then represented by the corresponding parametric set of curves E extracted from the CAD models. Given a set of M pairs of 3D scans and sharp edge annotations $\{\mathbf{X}_i, E_i\}_{i=1}^M$, our objective is to learn a non-linear mapping, Ψ , such that

$$\Psi : \mathbb{R}^{N \times 3} \rightarrow \mathcal{E},$$

$$\forall 1 \leq i \leq N, \quad \Psi(\mathbf{X}_i) = \hat{E}_i \approx E_i,$$

where \mathcal{E} is the set of all parametric edges, E_i is the set of ground-truth parametric sharp edges and \hat{E}_i is the set of predicted ones from the 3D scan \mathbf{X}_i .

We address the learning of the mapping Ψ through approximating it by a neural network we called *SepicNet*, which stands for Sharp Edges recovery by Parametric Inference of Curves.

As illustrated in Figure 3, the proposed approach consists of three main steps: (1) dataset preparation and sampling point clouds on 3D scans; (2) detecting sharp edge points on the sampled point clouds and decomposing the resulting points into different segments; (3) and fitting parametric curves on them. In next subsections, we explain further the aforementioned steps in detail.

3.1. Parametric Sharp Edges Dataset

The proposed approach is a supervised method that requires parametric sharp edge annotations of 3D scans. Consequently, we build on top of a publicly available dataset CC3D [2], which contains more than 50k CAD models in STEP format, unrestricted to any category, with varying complexity from simple to highly detailed designs. In

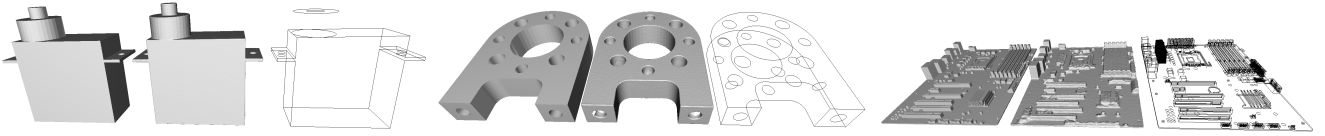


Figure 2. Sample models from CC3D-PSE, from left to right, the CAD model, the 3D scan and the parametric sharp edges.

	Sharp Edge type	Number of sharp edges per model					Total number of sharp edges	Number of models
		Minimum	Lower quartile	Median	Upper Quartile	Maximum		
All Models	Line	0	4	24	102	14278	7320145	50164
	Circular	0	0	0	15	8200	2079922	
	Spline	0	4	12	38	13764	2258783	
	All	0	16	52	186	23064	11658850	
Models with only line edges	Line	1	12	24	76	12480	-	3505
Models with only circular edges	Circular	1	3	4	8	4096	-	7835
Models with only spline edges	Spline	1	4	8	24	1681	-	265
Models with no sharp edges	-	-	-	-	-	-	-	1210

Table 1. Distribution of the number of sharp edges in the CC3D-PSE dataset.

this dataset, CAD models are virtually scanned and reconstructed using a proprietary 3D scanning pipeline resulting in pairs of 3D models in triangular mesh format and CAD models as b-reps. Unlike other alternatives such as ABC dataset, where the noise is usually synthetically added to the sampled point cloud, the 3D shapes in CC3D dataset have geometrical details such as edges smoothed due to the limitations of a scanning technology.

Accordingly, we propose a new dataset called *CC3D-PSE*¹ consisting of 3D models annotated with parametric sharp edges. Some examples are shown in Figure 2. We extracted sharp edge annotations from CAD models and transferred them to the corresponding 3D models. OpenCascade API was used to extract the parametric sharp edges directly from b-rep models of CC3D dataset. Three types of parametric curves are considered consisting of lines, circles and splines, that are common to describe the boundary curves in b-rep formats.

From the *CC3D-PSE* dataset statistics reported in Table 1, one can conclude that lines are the most common primitive curves with more than 7 million lines representing more than 62% of the edges in the whole dataset. The second most common primitive is spline with more than 2 million spline segments. However, models with only spline edges only represent 0.5% of the dataset which suggests that spline segments are usually combined with other types of primitives to construct sharp edges of a model. Circular segments are present in the dataset in a similar number to splines recording more than 2 million circular segments. Nevertheless, circular segments were enough to construct the sharp edges of 7835 models which consists of more than 15% of the dataset. The *CC3D-PSE* has been used in SHARP challenges².

¹The CC3D-PSE dataset is publicly available on <https://cvi2.uni.lu/cc3d-pse/>.

²<https://cvi2.uni.lu/sharp2022/challenge2/>.

3.2. Decomposition Module

Given an input point cloud \mathbf{X} sampled on a 3D scan, the decomposition module detects sharp edge points, consolidates them along the edge and groups them into different segments with primitive types identified.

Detection: The first part of the decomposition module predicts the sharp edge points by assigning a binary label to each point of the input point cloud indicating its belonging to a sharp edge. Rather than use standard binary cross-entropy loss, we use the focal loss [11] for its robustness to the class imbalance in the data, as the number of sharp edge points is usually smaller than the number of non-edge points. Therefore, the edge loss is defined by

$$\mathcal{L}_e = \mathcal{L}_{focal} = - \sum_{j=1}^N (1 - \mathcal{P}_j^*)^\eta \log \mathcal{P}_j^*, \quad (1)$$

where η is a modulating factor reducing the loss contribution from easy examples, $\mathcal{P}_j^* = \mathcal{P}_j$ if the point \mathbf{x}_j is an edge point and $\mathcal{P}_j^* = 1 - \mathcal{P}_j$ otherwise, with \mathcal{P}_j denoting the model’s estimated probability for the point \mathbf{x}_j being an edge point.

Consolidation: The consolidation of the points along the edge is done by predicting per point displacement vectors (offsets) $\hat{\mathbf{v}}_j \in \mathbb{R}^3$ by the network. After estimating the offsets, or in other words, the vector distance field to the closest edge, we apply them to predicted sharp edge points by a simple addition $\mathbf{x}_j^* = \mathbf{x}_j + \hat{\mathbf{v}}_j$. The loss function for edge distance field estimation consists of a regression of the offsets $\hat{\mathbf{v}}_j$ using L_2 -norm with respect to the ground-truth ones as follows

$$\mathcal{L}_o = \sum_{j=1}^N \|\hat{\mathbf{v}}_j - \mathbf{v}_j\|_2, \quad (2)$$

where the ground-truth offsets \mathbf{v}_j are computed in advance based on point-to-edge distances between the point cloud

sampled on the scan \mathbf{X} and the ground-truth edges E .

Primitive Type Classification: Our SepicNet predicts the type of the primitive of each segment among three possible types, namely, line, circle, and spline segments. The standard categorical cross-entropy loss \mathcal{L}_t is used to learn the primitive types. The primitive type of the segment is defined based on the major voting of its points.

Clustering on Embeddings: The number of different segments in different models vary greatly in CC3D-PSE dataset, thus we find it limiting to have a fixed maximum number of edge segments. In our case the clustering is done without knowing the number of segments a priori to discover groups of them belonging to the same segments. This is achieved by learning a point-wise embedding of the detected sharp edge points $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^d$, and applying the standard density based clustering algorithm on the embedded edge points to obtain a membership matrix $\hat{\mathbf{W}}$, such that $\hat{w}_k = \hat{\mathbf{W}}_{:,k} \in [0, 1]^{K_i}$ indicates the points belonging to k -th segment of a model \mathbf{X}_i with a total number of edges segments K_i . The embedding Φ is learned using a triplet mining strategy. Given an anchor sharp edge point \mathbf{x}_j^* , triplets of edge points $(\mathbf{x}_j^*, \mathbf{x}_j^{*+}, \mathbf{x}_j^{*-})$ are formed by considering an edge point from the same segment \mathbf{x}_j^{*+} and another one from a different segment \mathbf{x}_j^{*-} . The triplets are selected randomly during the training. Point pairs from the same segment are embedded closer to each other to form a cluster, while points belonging to different segments are pushed away using the following triplet loss

$$\mathcal{L}_{emb} = \sum_{j=1}^{n_{\mathcal{T}}} \max(\|\Phi(\mathbf{x}_j^*) - \Phi(\mathbf{x}_j^{*+})\|_2 - \|\Phi(\mathbf{x}_j^*) - \Phi(\mathbf{x}_j^{*-})\|_2 + \theta, 0), \quad (3)$$

where $n_{\mathcal{T}}$ is the number of triplets generated from the edge points and θ is a margin between positive and negative pairs. In our experiments, $\theta = 0.5$. During training, a differentiable version of the mean-shift clustering algorithm is used analogous to the one in [22]. At the inference time, we exchanged it with a GPU-accelerated version of hdbscan from cuML [19], as it improved the results in our experiments. Hdbscan is known to excel when the data has arbitrarily shaped clusters of different sizes and densities, and when a certain amount of noise is involved.

The final decomposition loss is a weighted combination of the all the above mentioned losses

$$\mathcal{L}_{dcmp} = \alpha_e \mathcal{L}_e + \alpha_o \mathcal{L}_o + \alpha_t \mathcal{L}_t + \alpha_{emb} \mathcal{L}_{emb}. \quad (4)$$

3.3. Fitting Module

The curve parameter estimation is an analytic function of segmented and offsetted points \mathbf{x}_j^* , i.e. the predicted membership matrix $\hat{\mathbf{W}}$. An edge segment is represented as one of the following types including a linear segment $H^l = (\mathbf{x}^s, \mathbf{x}^e)$, where $\mathbf{x}^s, \mathbf{x}^e$, are the start and end points respectively; a circular arc $H^c = (\mathbf{x}^c, \mathbf{x}^s, \mathbf{x}^e, r, \rho, \mathbf{n})$, where

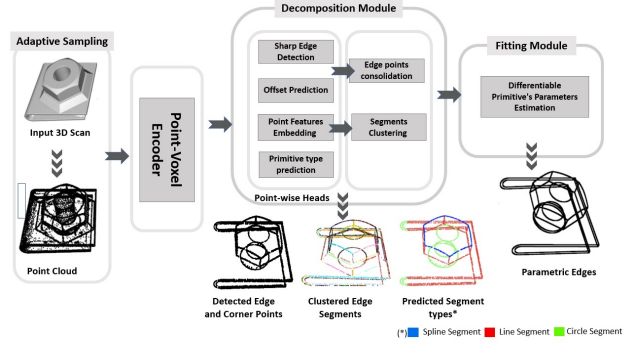


Figure 3. Overview of the SepicNet pipeline. The detection and segmentation modules take a 3D point cloud (with optional normals and curvatures) and decompose it into segments labeled by primitive type. The fitting module predicts parameters of a primitive that best approximates each segment. The three modules are jointly trained in an end-to-end manner.

$\mathbf{x}^c, \mathbf{x}^s, \mathbf{x}^e$ are center, start and end points, r is the radius, and ρ specifies the rotation direction with respect to the reference plane normal \mathbf{n} ; and b-spline H^b is defined as b-spline with control points $\{\mathbf{x}_d^{cp}, d \in (3, D - 1)\}$, where D is its degree.

Parametric curves fitting: These parameters of the curves are estimated via least-squares fitting based on differentiable SVD on the set of segmented edge points. Here we describe the estimation of the parameters $\{H_k\}$ for three types of curves (lines, circles, b-splines) from a set of 3D points $\mathbf{X} = \{\mathbf{x}_j\}$ and their predicted membership $\hat{\mathbf{w}} = \hat{\mathbf{W}}_{:,k}$. For simplicity, we will assume a fixed k and omit it in the formulas. Alongside, we also give the parametric definitions of curve segments introducing a single parameter t that is used for sampling points from a parametrized segment and to correctly define the boundary values on the sampling parameter t for each segment later. A line in parametric form is $L^l(t) = \mathbf{x}^s + tu$, where $u = (\mathbf{x}^e - \mathbf{x}^s) / \|\mathbf{x}^e - \mathbf{x}^s\|$ is a unit line direction vector, $t_s \leq t \leq t_e$. A circular arc is $L^c(t) = \mathbf{x}^c + u \cos t + v \sin t$, and $u = \mathbf{x}^s - \mathbf{x}^c$ and $v = u \times \mathbf{n}$. A spline is parametrized as $L^b(t) = \sum_{i=0}^k \mathbf{x}_i^{cp} \beta_i^K(t)$, where $\beta_i^K(t)$ is i_{th} b-spline basis function of order K .

The distance from a point \mathbf{x} to a line H^l can be calculated by the following formula:

$$D_{line}^2(\mathbf{x}, H^l) = \|(\mathbf{x} - (\mathbf{x}^s + tu))\|^2. \quad (5)$$

Defining \hat{H} as the minimizer to the weighted sum of squared distances, we have to solve

$$\mathcal{E}_{line}(H^l, X, w) = \sum_{j=1}^N w_j (X_{j,:} - (\mathbf{x}^s + tu))^2, \quad (6)$$

which becomes minimizing a linear least-square problem. Solving $\frac{\partial \mathcal{E}}{\partial \mathbf{x}^s} = 0$ we obtain $\tilde{\mathbf{x}}^s$ as the mean point $\tilde{\mathbf{x}}^s =$

$\frac{1}{n} \sum_i \mathbf{x}_j$. The parameter \hat{u} on the set of mean-centered points $\mathbb{X} = (\mathbf{x}_0 - \tilde{\mathbf{x}}^s, \dots, \mathbf{x}_{n-1} - \tilde{\mathbf{x}}^s)^T$ is a solution to

$$\mathcal{E}(u, X, w) = \|\text{diag}(w)^{\frac{1}{2}} \mathbb{X} u\|^2. \quad (7)$$

The solution

$$\tilde{u} = \arg \min_{u \in \mathbb{R}^3, \|u\|=1} \|Au\|^2 \quad (8)$$

is given by the right singular vector corresponding to the largest singular value of matrix $A = \text{diag}(w)^{\frac{1}{2}} \mathbb{X}$. As shown in [7], the gradient with respect to V can be backpropagated through the SVD computation.

To estimate the parameters of a circle from a set of 3D points we perform the following steps, first fit the plane onto a set of mean-centered points \mathbb{X} ; then project the mean-centered points onto a fitted plane to get 2D coordinates; use least-squares to fit circle in 2D coordinates to get the estimations of center $\tilde{\mathbf{x}}^c$ and radius \tilde{r} ; and finally, transform the circle center back to 3D coordinates.

To find the estimation of the normal $\tilde{\mathbf{n}}$ of a plane we follow [10], which leads to a homogeneous least square problem same as Equation 8, and can be solved in the same way.

The distance from a point \mathbf{x} to a circle H^c is defined as

$$D_{circle}^2(\mathbf{x}, H^c) = (\|\mathbf{x} - \mathbf{x}^c\| - r)^2. \quad (9)$$

To project 3D points onto the fitting plane the following Rodrigues rotation formula is used

$$X_r = X \cos \theta + (k \times X) \sin \theta + k(k \cdot X)(1 - \cos(\theta)). \quad (10)$$

where $k = \tilde{\mathbf{n}} n_z^T$, $n_z = (0, 0, 1)$, and $\theta = \arccos(\tilde{\mathbf{n}}, n_z)$. The implicit equation of a circle with a radius r and a center (x_c, y_c) in 2D is $(x - x_c)^2 + (y - y_c)^2 = r^2$, or $(2x_c)x + (2y_c)y + (r^2 - x_c^2 - y_c^2) = x^2 + y^2$, introducing $c = (c_0, c_1, c_2)$ as a vector for unknown coefficients. To solve $A_r c = b$ we consider fitting using differences of squared lengths and squared radius, or the following notion of the distance:

$$\mathcal{E}_{circle}(H^c, X, w) = \sum_{j=1}^N w_j (\|c - X_{j,:}\|^2 - r^2)^2. \quad (11)$$

Setting $\frac{\partial \mathcal{E}_{circle}}{\partial r^2} = 0$, we obtain the radius $\tilde{r} = (\frac{1}{n} \sum_j (\|\mathbf{x}_j - \tilde{c}\|^2))^{\frac{1}{2}}$. Plugging \tilde{r} back in Equation 11 we end up with

$$\mathcal{E}_{circle}(c, X, w, r) = \|\text{diag}(w)^{\frac{1}{2}} \mathbb{X} c - \mathbf{y}\|^2, \quad (12)$$

where $\mathbb{X}_{i,:} = 2(-X_{i,:} + \frac{\sum_{j=1}^N w_j X_{j,:}}{\sum_{j=1}^N w_j})$ and $\mathbf{y}_i = X_{i,:}^T X_{i,:} - \frac{\sum_{j=1}^N w_j X_{j,:}^T X_{j,:}}{\sum_{j=1}^N w_j}$. This least-squares problem is solved in differentiable manner [16] via Cholesky decomposition.

The fitting of b-spline curves is done via approximation using least-squares method with fixed number of control points. To fit b-splines we implement an algorithm A9.1

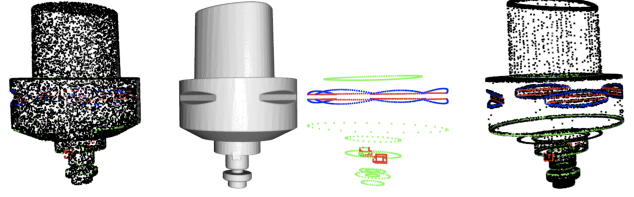


Figure 4. Adaptive curvature-based vs. uniform sampling, left-to-right: a uniformly sampled point cloud, original mesh, adaptively sampled point clouds with intensity factor $\gamma = 1.0$. The size of all point clouds is fixed to 10k points in the examples.

described in the book [17]. For this approximation to work, we need to specify the order of points in which the b-spline uses them in the formulation. This ordering we perform on the basis of the minimal spanning tree on the input embeddings returned by hdbscan. For each spline segment in this tree we find the longest path through breadth-first search, which gives the ordering of the corresponding points.

The fitting residual loss is a sum of segment distances between a predicted parametrized segment \hat{H}_k and a set of points \tilde{x} sampled uniformly $U(H_k)$ from ground truth segments H_k of type t_k defined as

$$\mathcal{L}_{fit} = \mathcal{L}_{res} = \sum_k \mathbb{E}_{p \sim U(H_k)} D_{t_k}(\tilde{x}, \hat{H}_k). \quad (13)$$

For basic primitives (i.e., lines and circles), the distances D_{t_k} between a point and a parametrized curve are computed analytically, while for b-splines, we approximate it with Chamfer distance [5]. The end-to-end training is performed with a sum of decomposition and fitting losses

$$L_{total} = \alpha \mathcal{L}_{dcmp} + \beta \mathcal{L}_{fit}. \quad (14)$$

3.4. Adaptive Sampling

The complexity of the models in CC3D-PSE dataset, as well as real 3D shapes can significantly vary from basic shapes to highly detailed designs with small elements as shown in examples in Figure 2. While uniform or Poisson sampling are typically used for point cloud generation from a 3D surface, these methods are insensitive to the high resolution details. To be able to recover sharp features, we develop a new adaptive sampling algorithm that produces dense point sets around regions with high-resolution geometrical details. The principal curvatures there undergo large changes compared to the low resolution regions. The weighted sample elimination technique, described in [26], is adapted to the weights computed on the principal curvatures κ_1 and κ_2 at a given point on a 3D models. For each point $\mathbf{x}_j \in \mathbb{R}^3$ in the point cloud of a 3D shape \mathbf{X}_i , we assign the weight according to the equation below:

$$\omega(\mathbf{x}_j) = \exp(|\kappa_1(\mathbf{x}_j)| + |\kappa_2(\mathbf{x}_j)|)^\gamma, \quad (15)$$

where γ is a controllable intensity factor.

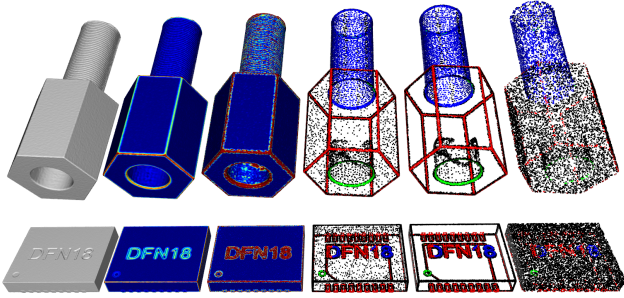


Figure 5. Adaptive curvature-based sampling, left-to-right: original mesh, calculated mean and gaussian curvatures on the mesh, adaptively sampled point clouds with intensity factor $\gamma = 1.0$ and $\gamma = 2.0$, a uniformly sampled point cloud. The size of all point clouds is fixed to 10k points.

Additionally, these principal curvatures are exploited further to enrich the point features that propagate through the network with Gaussian $\mathbf{k}_j = \kappa_1^j \kappa_2^j$, and mean $\mathbf{h}_j = \frac{1}{2}(\kappa_1^j + \kappa_2^j)$ curvatures. In Figure 4, we demonstrate an example of a point cloud uniformly sampled compared to our proposed adaptive scheme. In contrast to the standard uniform sampling, the proposed adaptive sampling captures sharp features better. In Figure 5, we also present a couple of examples that showcase the difference between the uniform sampling and our adaptive curvature-based sampling algorithm with respect to varying intensity factor γ .

4. Experiments

4.1. Setup

An input 3D shape is represented by a point cloud $\mathbf{X} \in \mathbb{R}^{N \times 3}$, where $N = 10k$ is the number of points in our experiments. The SepicNet is trained on CC3D-PSE dataset which is randomly split into three non-intersecting folds: 80% for training, 10% for testing and 10% for validation, which is approximately 35k, 7.5k, 7.5k models. Ground-truth point clouds are generated by adaptively sampling 10k points on the 3D scan surfaces, while the ground-truth edges parametrization is extracted from the corresponding CAD model. The data is normalized to a unit sphere. The decomposition model with the embedding size set to 64 is pre-trained to speed up the convergence for 100 epochs with Adam optimizer and Cosine Annealing learning rate schedule in the range of $(10^{-3}, 10^{-4})$. After that, SepicNet is initialized with pre-trained decomposition weights, and it continues fine-tuning together with fitting for 50 more epochs in end-to-end manner. The best model is selected based on the test set overall performance.

Metrics The multi-class classification of primitive type prediction is evaluated via Precision/Recall, and mean

intersection-over-union metric IOU . The matching segments intersection-over-union $sIoU$ is calculated for predicted edge segments that have been matched to the ground truth ones, where we use Hungarian matching to find correspondences between segments. The Chamfer distance between edges ECD is used to measure the quality of the final fitting results.

Computation and Implementation Details All the experiment are performed on a single Nvidia RTX 3090 GPU for all the methods and variations mentioned in this work. To demonstrate the versatility of the method to different architectures, we implemented and trained the SepicNet with two different backbones: PointNet++ [18], as a well established standard backbone for 3D point cloud processing, and a Point-Voxel CNN [13] as an efficient alternative for learning on 3D point clouds. The summary of the SepicNet performance is given in Table 2. The metrics achieved are similar in both cases, which reflects the generizability of the method. The average end-to-end inference time for SepicNet is around 3 sec per model. It is worth noticing that the inference time increases with the number of identified curve segments. The timings favor the PVCNN backbone specifically at training stage, so we report the results in the rest of our experiments for a PVCNN version of SepicNet.

Backbone	Training time days	Inference time sec per model	Precision \uparrow	Recall \uparrow	IoU \uparrow	ECD \downarrow
PointNet++ [18]	7	3.2	0.455	0.491	0.585	0.036
PVCNN [13]	5.5	2.9	0.457	0.488	0.586	0.037

Table 2. Performance of SepicNet with different backbones.

Our method has several parameters which values were selected based on a grid search results on a subset of the data. One of the most important choices which effects the convergence of the training significantly is a focal loss modulating factor η , the best edge detection performance was reached with $\eta = 1.5$. The combination of losses weights $\alpha_e = 1, \alpha_o = 10, \alpha_t = 2, \alpha_{emb} = 1, \beta = 1$ in Equation 14 results in a stable training in all our experiments. The distance threshold to the closest edge τ is set to 0.005 portion of the diagonal length of the bounding box of the model. This value is also well aligned with the maximum resolution of 128 of the voxel grid in PVCNN backbone. We ended up to set the adaptive sampling intensity factor $\gamma = 1.5$ as it does not produce very dense point clusters around the edges for models with only a few edges compared to large γ values, and outlines the edges for a selected size of point clouds (N=10k) for a major part of the dataset. The selection of control points of a b-spline is done via a refinement procedure. In our case, we iteratively upsample the control point by a factor of 2 until a fitting tolerance (0.01 in our experiments), measured as a Chamfer distance between the points of a segment and fitted b-spline sampled points, is achieved.

Dataset	Model	Precision \uparrow	Recall \uparrow	IoU \uparrow	ECD \downarrow
CC3D-PSE	EC-Net [25]	0.333	0.345	0.371	0.172
	PIE-Net [23]	0.321	0.316	0.39	0.153
	SepicNet (ours)	0.457	0.488	0.586	0.037
ABC*	PIE-Net [23]	0.521	0.516	0.590	0.083
	SepicNet (ours)	0.539	0.501	0.655	0.017

Table 3. Comparison with state-of-the-art methods.

4.2. Comparison with state-of-the-art

In this section, the results of the SepicNet model are compared with other state-of-the-art methods, PIE-Net [23] and EC-NET [25] on our CC3D-PSE and a subset of ABC [9] dataset.

The CC3D-PSE dataset was converted to PIE-Net and EC-Net input formats with uniformly sampled points per point cloud, and both were retrained from scratch with provided default parameters. The estimated metrics of the trained models on our CC3D-PSE dataset are provided in the Table 3. SepicNet significantly outperforms state-of-the-art on CC3D-PSE data. Since the authors of PIE-Net [23] did not share the overall data they used from ABC, we followed the protocol for data preparation they shared and collected a subset of around 20k models from ABC. We have retrained PIE-Net and our SepicNet on ABC* dataset and reported the metrics in Table 3. SepicNet demonstrates superior performance to other methods on both datasets, and the performance gain is more obvious on 3D scanned data of CC3D-PSE dataset.

4.3. Ablation Studies

We extend the set of input features with the local features extracted from the sampled points, namely, point location \mathbf{x}_j , point normal \mathbf{n}_j , Gaussian curvature \mathbf{k}_j , and mean curvature \mathbf{h}_j , calculated in this point. The results of the ablation experiments are summarized in Table 4. Given a different set of input features and samplings, we re-train the SepicNet in equal settings otherwise for each case.

As we can see, the uniformly sampled version of SepicNet $_{u}^{xnh}$ loses to other variations of adaptive SepicNet $_a$. The curvature-weighted adaptive sampling certainly captures the sharp features better than uniform and additionally boosts performance of SepicNet. According to the metrics, we have continuously improved the performance extending the set of input features (\mathbf{x} , \mathbf{n} , \mathbf{k} , \mathbf{h}), the points, normals and curvatures. This suggests that, indeed, curvatures bring additional discriminative information to SepicNet training. The qualitative results of our best performing SepicNet $_a^{xnhk}$ model on the test part of CC3D-PSE data are presented in Figure 6. The ECD reaches 0.037 in our experiments.

Noticeable, the adaptive sampling performs as designated, the high resolution details are well reflected in the

Model	Precision \uparrow	Recall \uparrow	IoU \uparrow	sIoU \uparrow	ECD \downarrow
SepicNet $_{u}^{xnhk}$	0.343	0.375	0.401	0.250	0.113
SepicNet $_a^x$	0.353	0.380	0.520	0.252	0.103
SepicNet $_a^{xn}$	0.416	0.438	0.530	0.353	0.082
SepicNet $_a^{xnhk}$	0.457	0.488	0.586	0.441	0.037

Table 4. SepicNet ablation performance with different sets of input features (\mathbf{x} , \mathbf{n} , \mathbf{k} , \mathbf{h}), where \mathbf{x} - point coordinates, \mathbf{n} - point normals, \mathbf{k} - Gaussian and \mathbf{h} - Mean curvatures of a point, u is short for uniform sampling, a stands for adaptive sampling.

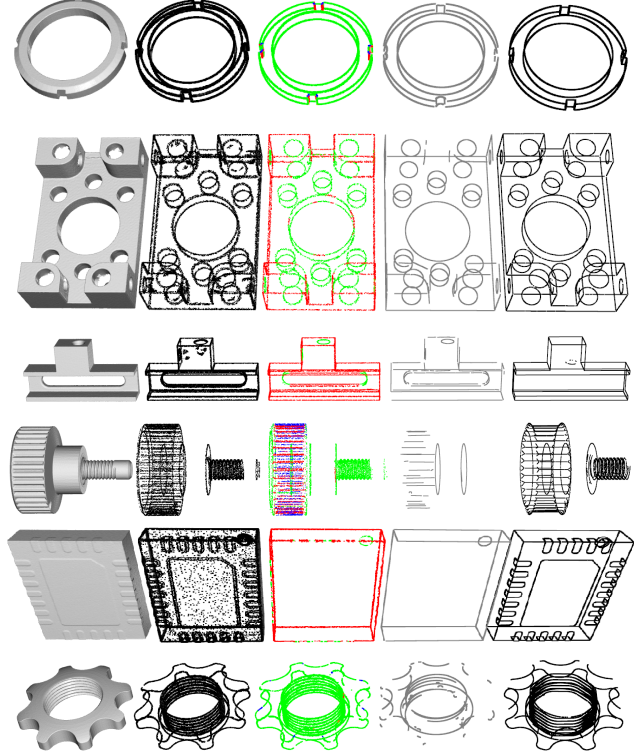


Figure 6. Results of our SepicNet. From left to right: the original 3D scan, the sampled point cloud, the sharp segments detected, primitives fitted and ground truth edges. Challenging examples in the bottom three rows such as dense threads and shallow printed elements

sampled point clouds. Nevertheless, comparing the performance of uniform SepicNet $_u$ with methods in Table 3, we confirm that our approach in case of uniformly sampled point cloud preforms better than both EC-Net and PIE-Net.

5. More visual results and discussion

More examples of the output of our SepicNet models at different steps of the pipeline can be seen in Figure 7. A visual comparison of a recovery of sharp edges from the parametric curves predicted by our SepicNet model are presented in Figure 8. The recovered edges are constructed by

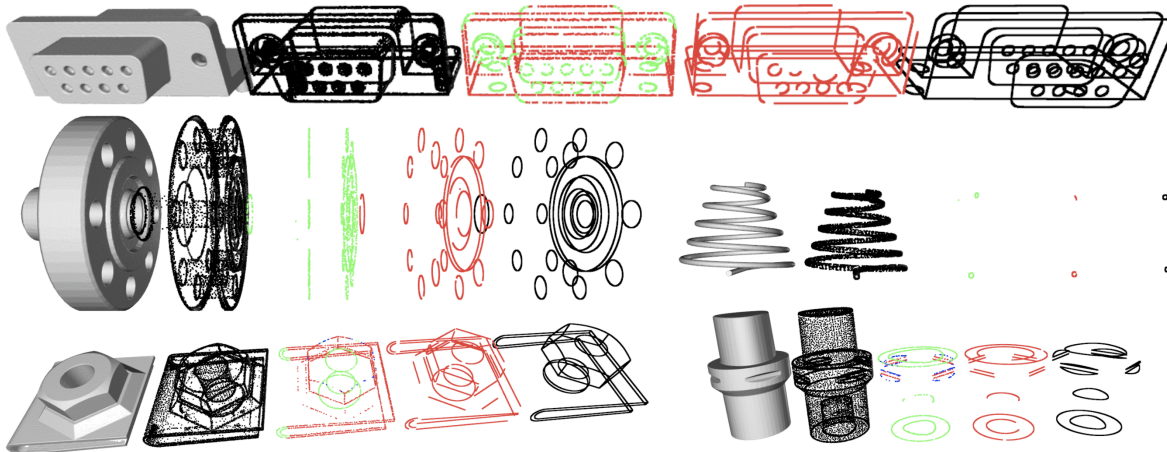


Figure 7. More examples of SepicNet results.

projecting nearby triangle vertices on the closest predicted sharp edge. According to the three main steps in our SepicNet pipeline for parametric inference of sharp edges, the major sources for the errors in the resulting parametrizations come from:

- the incorrect primitive types classification for edge points, resulting in the fitting of a wrong primitive as demonstrated in the bottom right example in Figure 7;
- the incorrect segmentation into curve segments, resulting into edges found at the detection stage, being omitted in the final set of parametric edges. We exclude such edges based on the threshold value of an error between the fitted primitive and the point set being approximated. An example of such case with a spline (blue) segments in third row right example in Figure 7;
- the shallow engravings and the printings on the surface of the model (as the one in Figure 6 fifth row) are common reconstruction failures due to a fixed resolution and a fixed number of points used during sampling;
- some segments are disconnected in the areas around corner points which were missed in the prediction. We believe this situation can be improved with a more dense sampling of points in the corner areas within our adaptive sampling strategy.

The results of SepicNet are more contained in the challenging examples which present in CC3D dataset with small holes and threads such as in three bottom examples in Figure 6. The degradation of the quality of the parametric inference can be explained by insufficient resolution of the point clouds, and a huge amount of segments in the model.

6. Conclusions

We presented a method to infer the parametric formulation of sharp edges of an object from its 3D shape in-

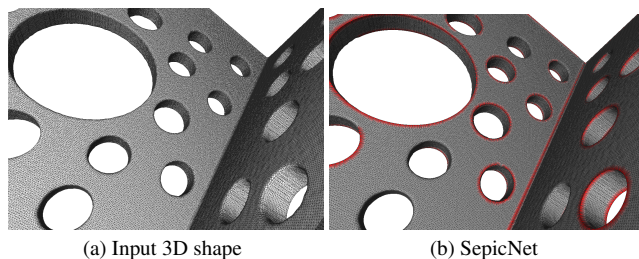


Figure 8. Example of a 3D shape recovered sharp edges with our SepicNet

der to mitigate the smoothing of sharp edges that is specific to 3D scanning. We also present a new large-scale dataset of aligned pairs of CAD models and scanned 3D models annotated with parametric sharp edges. Our end-to-end trainable network performance is further improved by a proposed adaptive sampling of point sets.

The major limitation of our method is the case when the model fails to predict and mark some parts of the edge segments as sharp, resulting in topological artifacts. We suggest that an additional module for the closed loops supervision can help to alleviate topological artifacts. In comparison with similar existing datasets, the CC3D-PSE contains many models with large numbers of edge segments, making the task particularly challenging. The proposed dataset has a broad variation in the complexity of the models, which makes it appealing for the community in the future research.

7. Acknowledgements

The present project is supported by the National Research Fund, Luxembourg under the BRIDGES2021/IS/16849599/FREE-3D and IF/17052459/CASCADES projects, and by Artec 3D.

References

- [1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. [2](#)
- [2] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2741–2745, 2020. [2](#)
- [3] Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Aryhannikov, Gleb Gusev, and Djamila Aouada. Cadops-net: Jointly learning cad operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*, 2022. [1](#)
- [4] Mostafa Ebrahim. 3d laser scanners: History, applications, and future. 10 2014. [1](#)
- [5] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, pages 2463–2471, 07 2017. [5](#)
- [6] Abid Haleem, Mohd Javaid, Ravi Pratap Singh, Shanay Rab, Rajiv Suman, Lalit Kumar, and Ibrahim Haleem Khan. Exploring the potential of 3d scanning in industry 4.0: An overview. *International Journal of Cognitive Computing in Engineering*, 3:161–171, 2022. [1](#)
- [7] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Training deep networks with structured layers by matrix backpropagation, 2016. [5](#)
- [8] Ahmet Serdar Karadeniz, Sk Aziz Ali, Anis Kacem, Elona Dupont, and Djamila Aouada. Tscm-net: Coarse-to-fine 3d textured shape completion network. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, pages 289–306. Springer, 2023. [1](#)
- [9] 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. *CoRR*, abs/1812.06216, 2018. [2](#), [7](#)
- [10] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. *CoRR*, abs/1811.08988, 2018. [5](#)
- [11] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society. [3](#)
- [12] Yujia Liu, Stefano D’Aronco, Konrad Schindler, and Jan Dirk Wegner. PC2WF: 3d wireframe reconstruction from raw point clouds. *CoRR*, abs/2103.02766, 2021. [2](#)
- [13] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019. [6](#)
- [14] Albert Matveev, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. 3d parametric wireframe extraction based on distance fields. *CoRR*, abs/2107.06165, 2021. [2](#)
- [15] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. Def: Deep estimation of sharp geometric features in 3d shapes. *arXiv preprint arXiv:2011.15081*, 2020. [2](#)
- [16] Iain Murray. Differentiation of the cholesky decomposition, 2016. [5](#)
- [17] Les Piegl and Wayne Tiller. *The NURBS Book (2nd Ed.)*. Springer-Verlag, Berlin, Heidelberg, 1997. [5](#)
- [18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. [6](#)
- [19] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020. [4](#)
- [20] Alexandre Saint, Anis Kacem, Kseniya Cherenkova, and Djamila Aouada. 3dboost: 3d body shape and texture recovery. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 726–740. Springer, 2020. [1](#)
- [21] Alexandre Saint, Anis Kacem, Kseniya Cherenkova, Konstantinos Papadopoulos, Julian Chibane, Gerard Pons-Moll, Gleb Gusev, David Fofi, Djamila Aouada, and Björn Ottersten. Sharp 2020: The 1st shape recovery from partial textured 3d scans challenge results. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 741–755. Springer, 2020. [1](#)
- [22] Gopal Sharma, Difan Liu, E. Kalogerakis, Subhransu Maji, S. Chaudhuri, and Radomir Mvech. Parsenet: A parametric surface fitting network for 3d point clouds. In *ECCV*, 2020. [4](#)
- [23] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges, 2020. [2](#), [7](#)
- [24] Francis Williams, Teseo Schneider, Cláudio T. Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. *CoRR*, abs/1811.10943, 2018. [2](#)
- [25] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. *CoRR*, abs/1807.06010, 2018. [2](#), [7](#)
- [26] Cem Yuksel. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, 34(2):25–32, 2015. [5](#)