DEF: Deep Estimation of Sharp Geometric Features in 3D Shapes

ALBERT MATVEEV*, Skoltech, Russia
RUSLAN RAKHIMOV*, Skoltech, Russia
ALEXEY ARTEMOV†, Skoltech, Russia
GLEB BOBROVSKIKH, Skoltech, Russia
VAGE EGIAZARIAN, Skoltech, Russia
EMIL BOGOMOLOV, Skoltech, Russia
DANIELE PANOZZO, New York University, USA
DENIS ZORIN, New York University, USA
EVGENY BURNAEV, Skoltech, AIRI, Russia

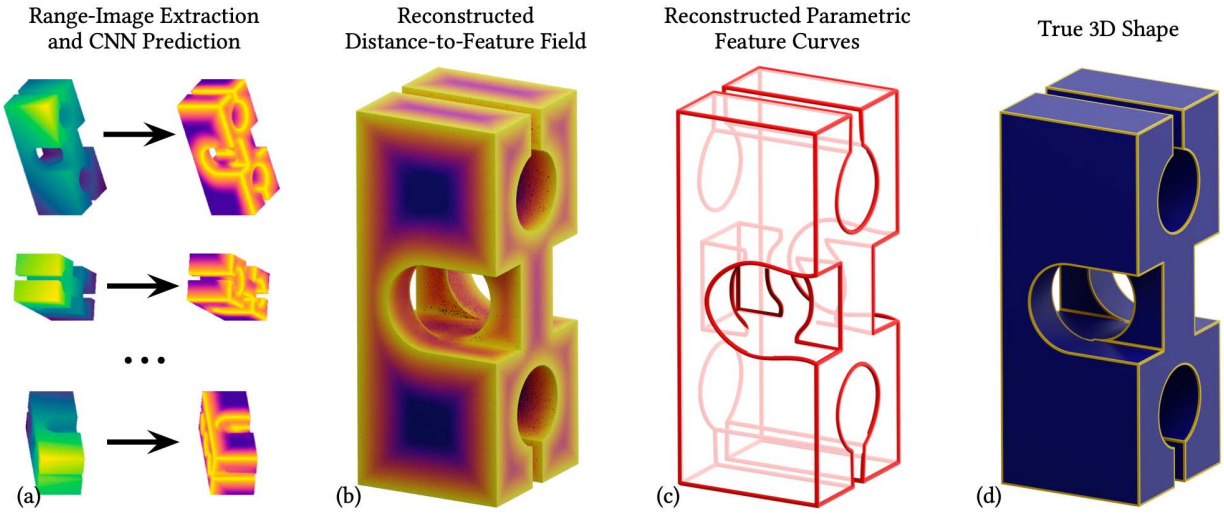

Fig. 1. (a) We leverage large collections of annotated geometric data to learn highly efficient patch-based deep models of distance-to-feature fields for range scan data. (b) We develop a view synthesis-based approach to combining the inference of such distance-to-feature predictions into a complete estimate for a full 3D shape. (c) Building upon our fields, we demonstrate the usage of our distance field in a downstream application, where we extract explicit representations of parametric feature curves from raw range scan data. (d) As a result, we deliver an accurate reconstruction of geometry and topology for both straight and curved feature lines, as displayed by a reference CAD model.

We propose Deep Estimators of Features (DEFs), a learning-based framework for predicting sharp geometric features in sampled 3D shapes. Differently from existing data-driven methods, which reduce this problem to feature classification, we propose to *regress a scalar field* representing the distance from point samples to the closest feature line on *local patches*. Our approach is the first that scales to massive point clouds by fusing distance-to-feature estimates obtained on individual patches.

We extensively evaluate our approach against related state-of-the-art methods on newly proposed synthetic and real-world 3D CAD model benchmarks. Our approach not only outperforms these (with improvements in Recall and False Positives Rates), but generalizes to real-world scans after training our model on synthetic data and fine-tuning it on a small dataset of scanned data.

We demonstrate a downstream application, where we reconstruct an explicit representation of straight and curved sharp feature lines from range scan data.

We make code, pre-trained models, and our training and evaluation datasets available at https://github.com/artonson/def.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Computer vision**; *Shape modeling*.

Additional Key Words and Phrases: sharp geometric features, curve extraction, deep learning

*Both authors contributed equally to the paper
†The author served as a technical lead for the project

Authors' addresses: Albert Matveev, Skoltech, Moscow, Russia, albert.matveev@skoltech.ru; Ruslan Rakhimov, Skoltech, Moscow, Russia, ruslan.rakhimov@skoltech.ru; Alexey Artemov, Skoltech, Moscow, Russia, a.artemov@skoltech.ru; Gleb Bobrovskikh, Skoltech, Moscow, Russia, g.bobrovskikh@skoltech.ru; Vage Egiazarian, Skoltech, Moscow, Russia, vage.egiazarian@skoltech.ru; Emil Bogomolov, Skoltech, Moscow, Russia, e.bogomolov@skoltech.ru; Daniele Panozzo, New York University, Courant Institute of Mathematical Sciences, New York, USA, panozzo@nyu.edu; Denis Zorin, New York University, Courant Institute of Mathematical Sciences, New York, USA, dzorin@cs.nyu.edu; Evgeny Burnaev, Skoltech, AIRI, Moscow, Russia, e.burnaev@skoltech.ru.

1 INTRODUCTION

Most human-made shapes have sharp geometric features, narrow curve-like regions with normals changing rapidly across the region. Sharp features are manually defined and explicitly stored in CAD models, and they are fundamental to faithfully represent the shape and function of CAD models. Detecting and reconstructing sharp features from scanned data is a vital geometry processing task: sharp feature curves can be used to improve the quality of many algorithms, such as surface reconstruction, including approximation with smooth patches, shape classification, and sketch-style rendering of surfaces.

Algorithms based on a priori analytic models of geometric features (*e.g.*, using curvature and its derivatives) often require per-object manual parameter tuning to detect features on a specific object (Section 2), making them difficult to apply to large collections of data or use as building blocks in a larger processing pipeline. Data-driven, learning-based methods, including ours, are a natural alternative for this task as they can leverage global information extracted from a training dataset and automatically adapt to a particular input shape without user interaction.

Our goal is to develop a reliable feature detection algorithm for sampled geometric data. While such data comes in a variety of forms, we focus on point-sampled data, specifically of the type produced by range scanners. Many other geometry representations (*e.g.*, level set meshes obtained from grid-sampled densities) can be easily converted to this form. Some of the most important characteristics of sampled geometric data include: (1) samples are almost never directly on (sharp) features; (2) the number of samples can be high (*e.g.*, for a complex model, a large number of depth images are typically combined into a single dataset with millions of points); (3) the data may be noisy.

We propose Deep Estimators of Features (DEF), a new approach to extracting sharp geometric features from sampled shapes, designed to work with this type of data. We designed our algorithm with the goals of capturing features without the need to sample them exactly, scaling to complex 3D models and large, possibly noisy, point clouds naturally, while at the same time enabling compatibility with real-world 3D acquisition setups (see Figure 1).

Our approach is based on defining features implicitly, by a *distance-to-feature function*; the problem we solve is a regression problem for this scalar function sampled in input points. The advantage of using a continuous distance-to-feature function, compared, *e.g.*, to a binary classification of points as feature and non-feature points, is that it is much more natural for samples not aligned with feature and noisy samples.

To address the need of handling large and complex models, we use local patch-based distance-to-feature prediction instead of a single-pass global prediction on the entire shape.

As for any supervised learning method, the quality of the results depends on the quality and size of the training dataset. Obtaining real 3D scanned data with ground truth is difficult, as it requires either manual annotation of scanned models, or precise fabrication and scanning of CAD data with annotated features; we follow the latter approach for our real dataset. For this reason, our method uses a two-stage training process (*cf*. [Gaidon et al. 2016] and [Handa et al. 2016]): we train an initial model on a large synthetic dataset and fine-tune it on a smaller dataset of 3D scanned data. The former is obtained by using a simplified simulated scanning process for a large number of models from ABC dataset [Koch et al. 2019]. For the latter, we fabricate and scan a smaller subset of ABC models, transferring annotations from the original CAD models.

We demonstrate that our method performs favourably on a number of metrics (RMSE, Recall, FPR) to four classical and learning-based state-of-the-art methods: VCM [Mérigot et al. 2010], Sharpness Fields [Raina et al. 2019], EC-Net [Yu et al. 2018], and PIE-NET [Wang et al. 2020].

As a sample application using our algorithm, we show that an explicit parametric representation of feature curves can be extracted from the estimated distance-to-feature fields produced by our algorithm (Figure 1 (c)), producing higher quality results, both qualitatively and quantitatively, than recent learning-based methods [Liu et al. 2021; Wang et al. 2020].

In summary, our contributions are:

(1) A method for estimating coherent distance-to-feature fields for high-resolution, high complexity sampled 3D shapes, including localized, CNN-based initial estimation of the field and global fusion of local estimates.
(2) A pipeline for constructing large simulated training datasets with controllable noise and different sampling patterns. This pipeline is used to produce a collection of benchmarks suitable for comparison of geometric feature detection algorithms.
(3) A process for constructing a real 3D scan dataset with ground truth distance-to-feature annotations and a new publicly available labelled set of range scans that can be used as a realistic benchmark.

2 RELATED WORK

Estimation of sharp features has been studied extensively in computer vision and computer graphics. We review both algorithmic methods relying on local estimation of differential surface properties and data-driven methods.

Normal Estimation, Clustering and Feature Detection on Local Sets. A popular family of methods [Bazazian et al. 2015; Demarsin et al. 2007; Weber et al. 2010], which can be applied directly on a point cloud or a triangle mesh, identifies a group of samples in a small area, computes their Gauss map using the samples' normals, and then performs clustering on the Gauss map to classify the neighborhood as belonging to a feature or not. Similar ideas can be applied to point set resampling with feature preservation [Huang et al. 2013].

A special case of such local estimators is Voronoi Covariance Measure estimator (VCM) [Mérigot et al. 2010]. It is based on constructing Voronoi cells of the local neighborhoods of points and computing covariance matrices of these cells. From these matrices, normals, curvature, and feature curves can be estimated. These methods require per-model tuning of parameters for both normal estimation and feature detection. In comparison, our method exploits the availability of datasets and automatically tunes its parameters to work on a collection of diverse shapes.

Surface Segmentation. Instead of directly detecting features, methods based on surface segmentation identify surface patches first

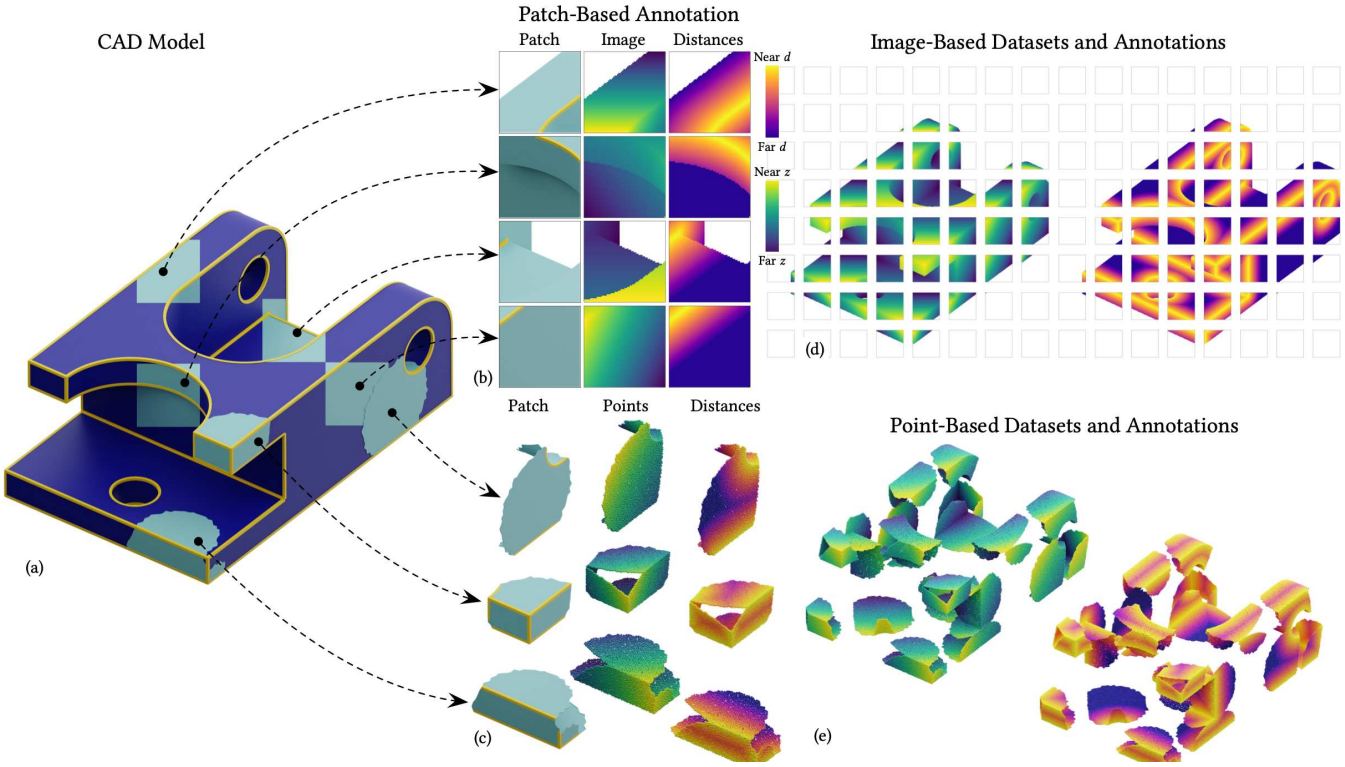

Fig. 2. Our patch-based pipeline for generating image-based (b, d) and point-based (c, e) training datasets proceeds as follows: (a) starts with a 3D CAD model, (b)–(c) extracts local triangulated patches and associated interior sharp feature curves, acquires ray-casted depth images and sampled point clouds, and computes local distance-to-feature annotations. The diversity of image and point patches in our large-scale training datasets (d)–(e) enables us to train highly effective sharp feature estimation models.

and then classify them as features the interface between them [Lin et al. 2017]. Additional priors can be used to help the segmentation, for example, for patches that are known to be developable [Lee and Bo 2016]. Several works [Lê et al. 2021; Li et al. 2019; Sharma et al. 2020] have attempted to fit surface patches after segmentation, however these approaches do not use feature curves and produce a disconnected set of surface patches with rough boundaries. These methods inherently require the entire model and cannot be applied to single views or incomplete models. Differently, our approach is directly applicable to incomplete data.

Patch Fitting. Feature fitting methods use a predefined set of primitives [Cao et al. 2016; Torrente et al. 2018] which are fitted to large regions of the mesh. These approaches are inherently more resilient to noise but increase the computational cost and require the features to be contained within a set of predefined shapes. Typical choices of features vary from a pair of planes sharing one edge [Lin et al. 2015] to spline curves.

A related, but somewhat distinct method [Daniels et al. 2007; Daniels Ii et al. 2008] relies on robust moving least squares (RMLS) [Fleishman et al. 2005]. This approach uses the quality of the local RMLS fit to determine the number of separate patches locally, and computes curve feature points as surface intersections, with several additional processing stages to obtain feature curves in the end. As with other categories, many parameters need to be adjusted to obtain good results.

Ground Truth and Representations. Only recently, multiple synthetic large-scale datasets with annotated features have been released [Kim et al. 2020; Koch et al. 2019; Willis et al. 2020]. In this work, we provide the first large-scale, objective comparison of algorithms working on triangle meshes and point clouds using the ABC dataset [Koch et al. 2019] and a real scan dataset derived from it.

Data-Driven Approaches. The identification of points lying on a sharp feature is most commonly cast as a binary classification problem, using a surface neighborhood (and potentially the normals or curvature of the neighboring points) as (learning) features. Different machine learning models were used, such as random forests [Hackel et al. 2016; Hackel et al. 2017], pointwise MLPs [Raina et al. 2019; Wang et al. 2020; Yu et al. 2018], or capsule networks [Bazazian and Parés 2021]. A recent work [Himeur et al. 2021] presents a lightweight MLP-based architecture paired with differential geometry-inspired scale-space matrices that encode features discriminative for edge detection. The methods that are closest to our work are [Liu et al. 2021] and [Wang et al. 2020]. These approaches classify feature and corner points and fit analytic features connecting the corner points and approximating the detected features. We compare against

state-of-the-art learning-based methods, discussing results and details in Section 7.2.

3 OVERVIEW

The input to our algorithm is a set of depth images (possibly with missing data), for a given object. In the case of real scanned data, these images are obtained directly from the scanner; in the case of synthetic mesh data, we simulate the scanner to generate a collection of depth images from a mesh (Section 4.2). The algorithm outputs estimates of the truncated distance-to-feature scalar function for each input point. Figure 1 (a)–(b) illustrates this process.

The four main components of our method are:

(1) *Training Data Construction* (Section 4). We generate synthetic training data using the ABC dataset [Koch et al. 2019], obtaining collections ranging from 16,384 to 262,144 training instances. To fine-tune the model and evaluate its performance on real scans, we introduce a fabrication, scanning, and semi-automatic annotation pipeline to create a dataset of 84 real-world models. Our data generation pipeline accepts a set of meshes and their associated feature annotations (edges marked as sharp) as input and produces a set of point-sampled local patches with point-wise distance-to-feature labels as output (Section 4.1). We specify the details on the implementation of our two datasets, the synthetic *DEF-Sim* and the real-world *DEF-Scan*, in Sections 4.2–4.3.

(2) *Patch-Based Deep Estimators* (Section 5.1). We train a family of deep feature estimators (DEF), which produce distance-to-feature estimates, on patches (depth images) of the synthetic dataset and fine-tune on a subset of the real-world dataset.

(3) *Estimation on Complete 3D Models* (Section 5.2). The per-patch distance-to-feature predictions produced by DEFs are fused together by transferring estimates from each patch to overlapping patches and combining into a coherent global estimate.

(4) *Feature Fitting* (Section 6). The last (optional) component extracts explicit feature curves from the distance-to-feature function. We show that with our distance function estimate, simple corner detection, combined with kNN clustering and spline fitting, produces higher quality results than state-of-the-art methods.

In the next sections, we describe each component in detail and provide a rationale for each algorithmic choice.

4 DATASETS WITH DISTANCE-TO-FEATURE ANNOTATION

4.1 Dataset Design

Feature Definition. Each CAD model in the ABC dataset is defined by a boundary representation (B-Rep), providing a partitioning of its surface into a collection of CAD regions and associated parametric curves. Analytically, we identify sharp features as curves at the interface between any two regions for which the normal orientations defined in either region differ by more than a particular threshold α_{norm} (we use $\alpha_{\text{norm}} = 18^\circ$) as was done during the construction of ABC dataset [Koch et al. 2019]. The threshold is necessary as CAD models commonly have smooth areas partitioned in multiple regions, which would result in spurious features.

Directly using the original parametric representations, however, makes it difficult to construct a large training dataset, as B-Reps either need to be traversed using off-the-shelf geometric kernels [Open CASCADE Technology OCCT 2021; Parasolid: 3D Geometric Modeling Engine 2021], a software not designed for batch processing, or require re-implementing a set of elementary operations like closest point, which require nonlinear solvers on B-Reps. To avoid these issues, we use the triangulated versions of the ABC models, where CAD region and sharp feature curve labels are available for vertices and edges in each mesh; we introduce a set of easily testable geometric conditions into our data generation procedure to prevent introducing significant geometric errors when sampling B-Rep data. We use the curve annotation provided in the ABC dataset to identify the mesh edges which were marked as sharp to base our distance field on the proximity to the corresponding mesh edge.

Patch and Feature Selection. Mesh models in ABC vary significantly in geometric complexity [Koch et al. 2019], requiring an adaptive number of samples to represent their 3D surface geometry (in the original dataset, meshes are sampled with 10^2–10^7 vertices), see Figure 4. However, having variable size, high resolution 3D shapes as input is not a good fit for training most state-of-the-art learning algorithms, which require a fixed number of samples and require too much memory and training time to handle hundreds of thousands of samples [Henderson et al. 2020]. To address this problem, we decompose each shape into a *collection of patches* with a small and fixed number of samples, see Figure 2 (a)–(c); this is different from a number of existing trainable approaches [Wang et al. 2020] that represent *entire shapes* with the same (fixed) number of samples.

Selecting patches and feature curves for training has a direct impact on performance. We distinguish between *interior, contour,* and proximal *exterior* curves, depending on their visibility status; we keep interior curves for annotation and exclude the latter two types. Features appearing as a contour of a sampled region are difficult to distinguish from smooth features; being adjacent to only a single visible surface patch provides insufficient spatial context for inferring these from point samples. Exterior features pass within distance truncation radius ε but still outside the visible patch. Including exterior features would lead to distance-to-feature annotations indicating feature proximity; however, regressing such features from the local patch context would be impossible due to absence of samples covering them. In contrast, we generate the per-patch annotations locally in each patch, using only feature curves passing through the patch interior. Figure 3 demonstrates example annotations obtained by varying the set of included features.

Similarly, patches with depth discontinuities and gaps represent challenging cases with many contour feature curves, see Figure 3, rows 2–3; however, these naturally occur due to shape self-occlusions or ray misses during both ray-casting and real scanning. We have experimentally observed that including such instances in training improves performance, particularly at near-boundary pixels that are regressed more accurately; we discuss their effect and alternatives in our ablative experiment (Section 7.4).

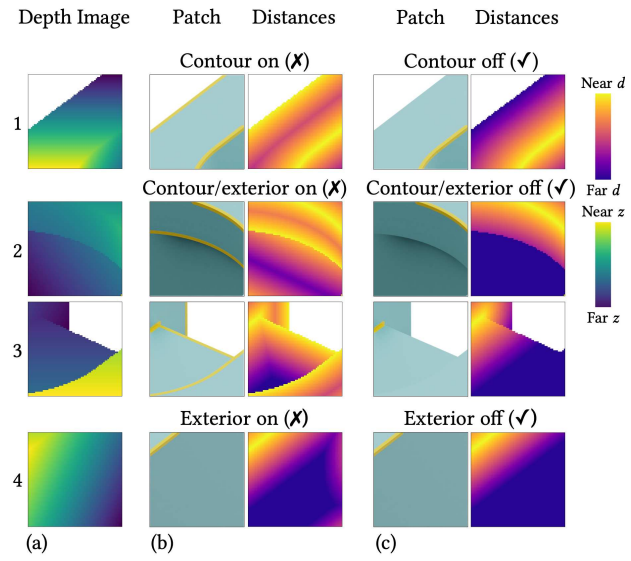

Fig. 3. The same depth data in column (a) may be annotated differently, depending on which adjacent feature curves are included when computing distances. *Contour* features (*i.e.*, features adjacent to only a single visible surface patch; shown in column (b), rows 1–3) are difficult to distinguish from smooth contours; *exterior* features in close proximity (*i.e.*, features passing outside patch but within distance truncation radius ε; shown in column (b), rows 2–4) are impossible to detect due to absence of samples covering them. We opt to generate the per-patch annotations locally in each patch, using only feature curves passing through the patch *interior* (*i.e.*, both adjacent surface patches are sampled, shown in column (c), rows 1–4).

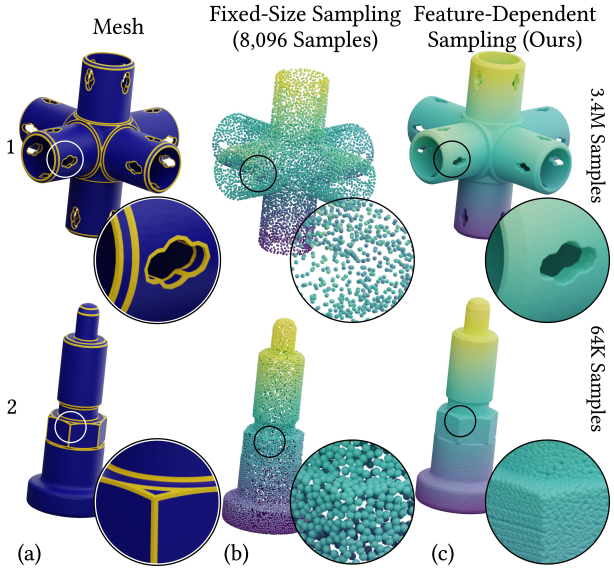

Fig. 4. Differently from existing approaches, that represent all mesh models (a) by a fixed number of samples (b) despite dramatic differences in their geometric complexity (*cf*. rows 1 and 2), we decompose input 3D models into variable-length sets of local patches with a fixed number of samples; as a result, complete 3D shapes sampled using our method have variable number of samples (c).

Distance-to-Feature Computation. As our focus is on sharp feature detection, large values of the distance-to-feature function have little impact on feature localization but require more effort to predict correctly. For this reason, we define a *truncated* distance-to-feature field $d^\varepsilon(p)$ in each location $p \in \mathbb{R}^3$ using the proximity to a subset of mesh edges corresponding to (sharp feature) curve segments $\Gamma = \{\gamma_k\}_{k=1}^K$ in $\mathbb{R}^3$ as follows. We find for p its closest (in Euclidean sense) neighbor located at one of the segments in Γ, *i.e.* a point $q(p)$ such that

$$\|q(p) - p\| = \min_{\gamma_k \in \Gamma} \inf_{q \in \gamma_k} \|q - p\|, \tag{1}$$

and define the $d^\varepsilon(p)$ by

$$d^\varepsilon(p) = \min(\|q(p) - p\|, \varepsilon), \tag{2}$$

where we set our truncation radius ε to a multiple of the sampling distance r (we set $\varepsilon = 50$, $r_{\text{high}} = 1$ where $r_{\text{high}} = 0.02$ is a base sampling step), leaving a sufficiently wide envelope where our distance field may provide meaningful feature-related information.

We use Euclidean distance as opposed to the geodesic distance along the surface. We compute distance-to-feature annotation for a sampled point p by associating it to the closest surface spline region within the patch (this association accounts for sampling noise) and only considering sharp feature curves belonging to the contour of that surface region in the ABC feature annotation, see Figure 5. More generally, we construct a surface region/feature curve adjacency graph where each surface region and feature curve (two nodes) that share mesh vertices are connected by an edge, and perform depth-first search of depth k to determine which features should be included in the distance computation over a particular surface region. We additionally record $q(p) - p$, directions to the closest points on the feature curves, for use in the ablation study.

Feature Size and Sampling Density. To accurately reconstruct the distance-to-feature function, it is not safe to rely on *fixed-size* input point clouds for whole objects (as it is done in recent literature [Liu et al. 2021; Wang et al. 2020]), since many curves are left severely undersampled, see Figure 4. Instead, we assume that most feature curves are sufficiently densely sampled, and that the presence of feature curves can be inferred from the positions of samples; that leads us to have an *adaptive* number of point samples per object. This assumption is motivated by a common practice in high-quality 3D data acquisition of adapting the number of points per object and sensor placement to the geometric complexity and size of the object.

One way to reason about "sufficient" sampling is to choose a characteristic (object-dependent) spatial size l for each shape and require that features of size close to l are represented by, on average, n samples. Formally, we require the following relation to hold:

$$\underbrace{r}_{\substack{\text{sampling}\\ \text{distance}}} \times \underbrace{n}_{\substack{\text{num. samples}\\ \text{per feature}}} = \underbrace{l}_{\substack{\text{characteristic}\\ \text{spatial size}}} \times \underbrace{s}_{\substack{\text{scaling}\\ \text{factor}}}, \tag{3}$$

where we are free to vary either the sampling step r or the object scaling factor s to achieve the equality (in practice, for each particular dataset, we fix r and vary s). Our characteristic spatial size l is a linear measure set to to 25% lower quantile of the distribution of

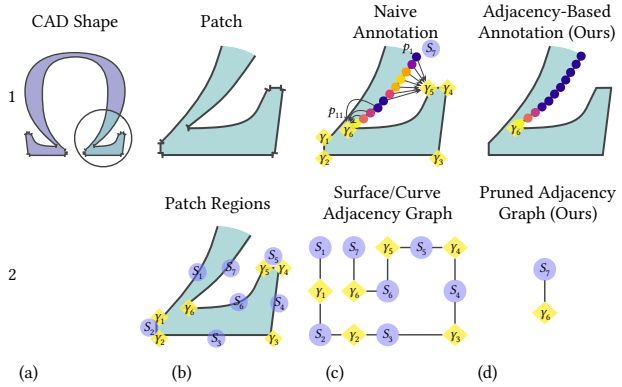

Fig. 5. Extracting a patch from an example 2D CAD shape in (a) produces a mesh fragment consisting of seven surface regions S_I along with six associated interior feature curves γ_k (rows 1–2 (b)). For samples $p_i \in S_7$, naive computation of distances $d^\epsilon(p_i)$ maps $p_1, \ldots, p_7$ to the feature γ_5 (row 1 (c)) which is disconnected from the region S_7, despite proximity in the Euclidean sense (row 2 (c)). In contrast, we compute more natural distances, excluding non-contour curves for each surface region (for S_7, all but γ_6 are excluded as in row 1 (d)) by constructing and pruning the surface/curve adjacency graph (row 2 (d)).

sharp feature curve extents, where "extent" denotes the maximum of three dimensions of an axis-aligned bounding-box enclosing a curve. Figure 6 provides an illustration of this scheme.

Patch-Based Datasets. We run our patch generating algorithm on the first five chunks of the ABC dataset (37,945 3D shapes) and obtain three major data varieties at low, medium, and high resolution by choosing $n_{\text{low}} = 8$, $n_{\text{med}} = 2.5 \times 8 = 20$, and $n_{\text{high}} = 2.5^2 \times 8 = 50$ samples per curve. Each resolution corresponds to sampling distance $r_{\text{low}} = 0.125$, $r_{\text{med}} = 0.05$, and $r_{\text{high}} = 0.02$, respectively. Similarly to related methods [Wang et al. 2020; Yu et al. 2018], we model acquisition uncertainty using additive Gaussian white noise; we use five scales in the viewing direction with a standard deviation $\sigma \in \{\frac{r}{8}, \frac{r}{4}, \frac{r}{2}, r, 2r\}$, for the high-resolution data only. For each of the mentioned variations we obtain training sets of sizes ranging from 16,384 to 262,144 patches to assess the impact of dataset size on performance (see Supplementary material for details).

Complete 3D Model Datasets. Complementing our patch-based data, we constructed datasets of 3D shapes representing object-level data samples of 3D CAD models, both *synthetic* and *real*.

We emphasize that the design principles outlined in this section are used uniformly for both our synthetic and real-world datasets, enabling direct fine-tuning of our networks for the real scenario.

We have selected a diverse set of 68 distinct CAD models from the ABC dataset. Our focus when choosing the models is to cover a variety of qualitative properties, including (1) presence of thin walls and (2) various types of surface regions (*e.g.*, flat, cylindrical, splines, and spheres), (3) curved and straight features, (4) variety of angles incident on sharp features, and (5) presence of fillets. The statistics of the selected models are analyzed in the Supplemental. The models are sampled and annotated as described in this Section to form the input complete 3D shapes.

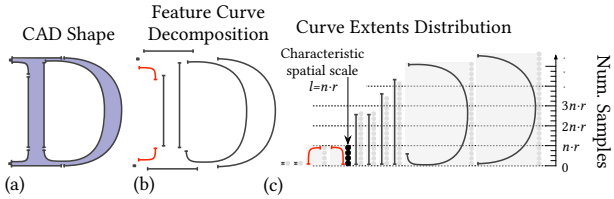

Fig. 6. For an input CAD shape in (a), we analyze the distribution of sharp feature curve extents in (b) and relate a sampling radius r to features of characteristic spatial size l, sampling these with at least n points in (c) (see Equation (3) and surrounding text).

4.2 Synthetic Datasets: *DEF-Sim*

Our synthetic datasets provide collections of local patches and 68 complete 3D models in varieties of low, medium, and high resolution, and several noise levels.

Shape Sampling. We set up n_v virtual cameras with locations evenly distributed on a sphere around an object (we use Fibonacci sampling [Hannay and Nye 2004]) and the z-axis pointing at its center of mass. For each camera, we create a regular grid (image) with 64×64 pixels (we specify r as the pixel size) and cast rays from each pixel's corner in a direction perpendicular to the grid, obtaining patches with up to 4,096 point samples each (some may not correspond to an object point and are set to a background value).

Knowing the camera parameters (K, T) where $K \in \mathbb{R}^{2\times3}$ is an intrinsic matrix transforming point coordinates from the camera coordinate frame to the image plane and $T \in \mathbb{R}^{4\times4}$ an extrinsic camera matrix transformation from the camera coordinate frame to a global coordinate frame [Hartley and Zisserman 2004], sampled points $p_{ij} = (x_{ij}, y_{ij}, z_{ij})$ (in homogeneous coordinates) may be identified with a depth image $I = (z_{ij}^{\text{cam}})$, where $z_{ij}^{\text{cam}} = (KT^{-1}p_{ij})_3$ denotes z-coordinate of point p_{ij} in the camera frame. We create the distance-to-feature annotations image by computing $d = (d^\epsilon(p_{ij}))$ and record the pair (I, d) as the training instance. We use $n_v = 18$ and augment the dataset by rotating and offsetting the image grid during data generation, but maintaining the same orientation of z-axis; we discuss the effect of having varying number of views n_v in the ablation study (Section 7.4).

4.3 Real-World Datasets: *DEF-Scan*

To support generalization to real-world scanning data, we constructed a dataset of 84 real objects and semi-automatically annotated them. Figure 7 presents an overview of the steps involved in the construction of our datasets; details on the selection of CAD models are mentioned in Section 4.1.

Fabrication. As we sought to fabricate a multitude of arbitrary 3D models with high geometric complexity, we opted for fabricating the models using 3D printing, as it can easily produce shapes directly from CAD models. We used two commodity polylactic acid (PLA) devices (Ultimaker 3 and Ultimaker S5) and considered implications of this choice (most importantly, its accuracy and layer thickness of 0.1 mm). We choose the printed object size to allow acquisition with our 3D scanner at a specific sampling density of the features while

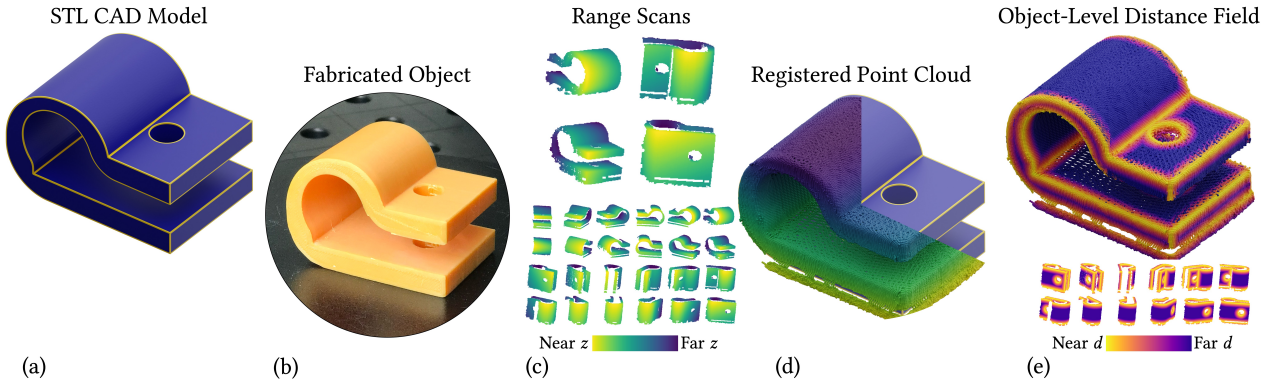

Fig. 7. (a) We have selected a diverse set of 84 3D CAD models from the ABC dataset and (b) fabricated them in thermoplastic using the 3D printing technology. (c) We further obtained 12 scans of each shape in two different orientations (totalling 24 scans per object) using a commercial structured-light 3D scanner. (d) We semi-automatically registered the 3D scans onto the original CAD model, computed distance-to-feature annotations in (e), and finally processed the scans to obtain our patch-based datasets.

simultaneously avoiding scanning any fabrication artifacts. We pick a sampling density value $r > 0.1$ mm for our 3D scanner by selecting a scanning distance (see below), and compute a scaling factor s_i a for each fabricated model M_i individually using the relation (3). The fabricated CAD models are displayed in Figure 8.

Scanning. Our depth acquisition process seeks to obtain a homogeneous set of range scan data capturing most of the surface for the fabricated models and suitable for point-based and image-based training. We use RangeVision Spectrum [RangeVision Spectrum 2021], a commercial structured light 3D scanner, to acquire the geometry of the fabricated objects in the form of depth images. The scanning sequence we use captures the object from two orientations w.r.t. the scanner, differing by 90°; in each orientation, we take a scan every 30° using an automated turntable to minimize the operator time. Our resulting scans are acquired from an average range of 2 m and have the resulting sampling distance r of approximately 0.5 mm. In total, we have acquired 1928 depth images corresponding to 166 scanning sequences of 84 unique objects. We give more detailed statistics on our real-world dataset in the Supplemental.

Fig. 8. A photo of the thermoplastic 3D CAD models fabricated for the evaluation of our approach in a real-world setting.

Registration with the CAD Models. Our 3D scanner automatically provides an initial alignment between the obtained 3D scans; however, we found this alignment too coarse. Hence, we manually registered all scans to their respective CAD models using the Align Tool in MeshLab [Cignoni et al. 2008] by first marking 3 points on each scan-mesh pair for rough manual alignment, followed by running the ICP algorithm for refinement. We find that manual alignment results in significantly tighter fits.

5 DEEP ESTIMATION OF DISTANCE-TO-FEATURE FIELDS

5.1 Learning Patch-Based Deep Estimators

We train our deep regression models by solving the standard learning task: given a set of N training instances, find

$$\min_{\theta} \frac{1}{N} \sum_{i}^{N} L(d_i, f(P_i; \theta)),$$

where d_i is the ground-truth distance-to-feature field for the patch P_i, $f(\cdot;\theta)$ is the model with trainable parameters θ, and L is the loss function. We have considered a few options for elements in this setup, to identify an optimal learning configuration. We summarize these choices below and present the qualitative comparisons of different options in Section 7.4 and their effect on method robustness in Section 7.5.

Network Architectures and Losses. Overall, we found that CNNs working with regularly resampled data outperform point-based networks for our task (Table 6). We require our deep models to generalize to many unseen targets with high geometric variability, thus we search for network architectures with sufficient capacity. We use the U-Net CNN model [Ronneberger et al. 2015], which has proven effective for image-based dense regression [Xue et al. 2019], and probe the ResNet family [He et al. 2016], selecting the largest

Table 1. In our experiments, directly optimizing Histogram loss [Imani and White 2018] significantly improves performance across different quality measures. We present results computed using the validation set of depth images (with background), with sampling distance $r_{\text{high}} = 0.02$, and noise variance $\sigma^2 = 0$.

Loss	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
L_2 (MSE)	101.3	643	24.2	0.11
L_1 (MAE)	108.7	691.2	23.5	0.06
Histogram	**61.5**	**361.1**	**57.4**	**0.06**

(ResNet-152) base network based on the quality of predictions on the validation set. For full details on the influence of model size on performance, we refer to Supplemental.

We compare three types of losses for our regression task: L_1 loss, L_2 (MSE) loss, and the Histogram loss [Imani and White 2018]. The latter one requires the model to produce a histogram of values over a predefined interval; we empirically found out that histograms with 244 bins work best on the validation set. Overall, we observed that learning with the Histogram loss considerably improves regression quality measured by all metrics as presented in Table 1. We attribute this to the restriction being imposed on the range of the possible target (ground-truth) distances, allowing the network to focus on a narrow range of targets. Our final setup with the Histogram loss predicts a confidence score for each bin in the histogram and computes the final output as a weighted sum of bin centers multiplied with their respective normalized predicted scores.

Additional Inputs, Supervision, and Data Volume. The second critical ingredient that we investigate is the dataset size and features available in training datasets.

To assess the gains from *additional inputs,* we concatenate the additional values to the point coordinates: we used the binary sharp feature point segmentation labels obtained by the non-learning algorithm VCM [Mérigot et al. 2010], ground-truth normals, as well as both of these values, keeping distances as our only target variable. Neither of these additional annotations resulted in performance improvement.

To evaluate whether learning configurations for our task benefit from richer *supervision* compared to distances only, we introduce additional network heads regressing either normals, normalized directions towards the nearest sharp feature line, or both simultaneously. During training with these targets, we optimize a multi-task loss consisting of our main loss and a weighted sum of MSE losses with weights chosen to balance the magnitude of losses: 10^{-3} for normals, and 10^{-2} for directions. None of these configurations led to improved regression performance either. We also trained the network on *datasets of increasing size;* we observed that performance stabilizes for datasets with more than 64,000 training instances.

In summary, the best-performing choice of architecture was a CNN U-Net with ResNet-152 backbone, trained using the Histogram loss using the supervision from ground-truth distances $d(p)$ only, on datasets of size at least 64,000. We present detailed results of mentioned experiments in the Supplementary material.

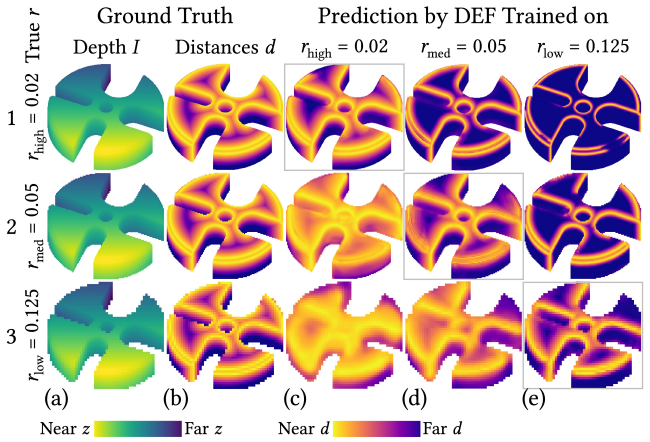

Fig. 9. Network responses to probe depth images sampled at different rates reveal high feature sensitivity and sampling robustness of our deep models; in instances with sufficient samples between feature curves, our method efficiently relates samples to respective closest feature lines. We obtain ground-truth data (a)–(b) by raycasting a 3D model at sampling distances $r_{\text{high}} = 0.02$, $r_{\text{med}} = 0.05$, and $r_{\text{low}} = 0.125$ and produce predictions (c)–(e) using DEFs pre-trained for regressing features at $r_{\text{high}} = 0.02$, $r_{\text{med}} = 0.05$, and $r_{\text{low}} = 0.125$, respectively.

Feature Detection at Varying Sampling Distances. Each DEF network, though trained on data with a specific sampling rate r in (3), can detect interior features sampled at significantly different rates; in Figure 9, features sampled at $r_{\text{low}} = 0.125$ are robustly regressed by DEFs trained on 2.5× ($r_{\text{med}} = 0.05$) and 6.25× ($r_{\text{high}} = 0.02$) finer sampling, and vice versa. Importantly, when sampling distance in inputs matches that of training datasets, DEF predicts a proper *distance field*; otherwise, DEF produces a scale-transformed *proximity field* whose iso-contours capture true features.

5.2 Reconstructing Distance-to-Feature Fields on Complete 3D Models

The trained deep estimators sense distance variations in the direct vicinity of the *interior curves* visible in individual patches of an input shape; predictions in any two distinct patches may diverge substantially if feature curves are captured differently (*e.g.*, a feature appears as an interior curve in one patch but shows up as a contour in another), see Figure 10 (c). Given a set of these partial and inconsistent estimates (with known camera parameters), we reconstruct a distance-to-feature field defined globally on a complete 3D shape; we give an overview of this *fusion* process in Figure 10.

Patch Extraction. (Figure 10 (a)–(b).) We convert an input 3D model into a collection $\{I_i\}_{i=1}^{n_v}$ of n_v range images suitable for our patch-based DEF. We assume that the input 3D shape either already comes as range images (*e.g.*, for range scanning) or can be resampled (*e.g.*, represents volumetric data). In the latter case, we obtain depth maps of the input shape from multiple distinct directions using raycasting. As our deep models are fully convolutional, we employ *full-object views* I_i of input 3D models to compute predictions, which

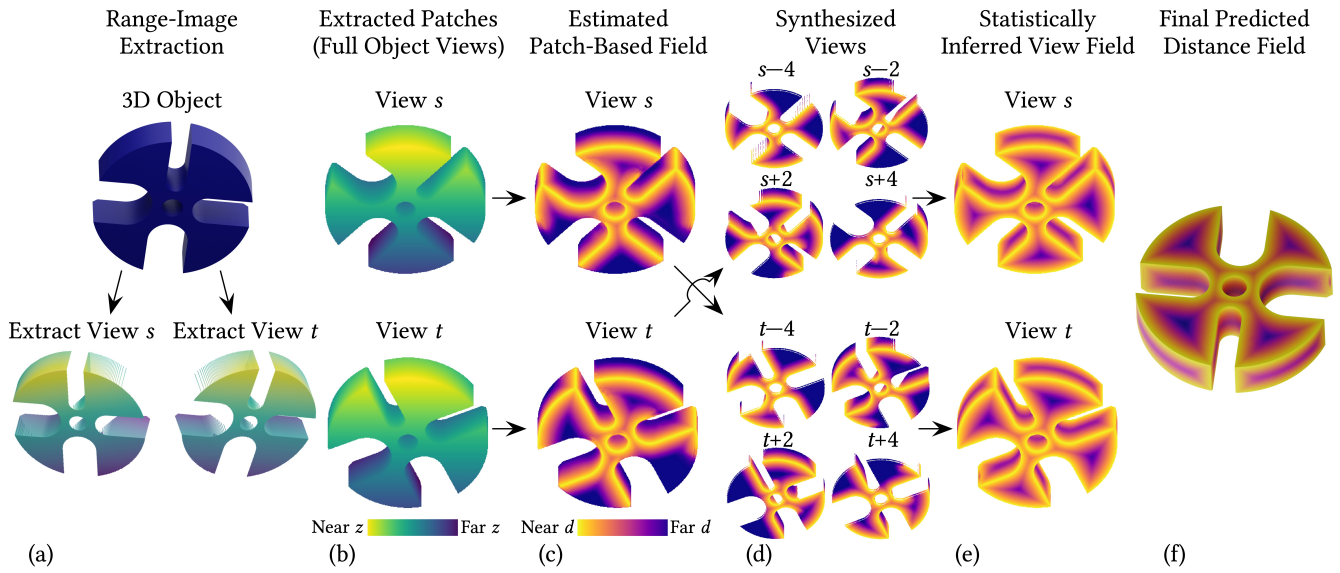

Fig. 10. Our method for reconstructing distance-to-feature fields on 3D shapes is built around postprocessing distance-to-feature predictions obtained in individual patches (or views). (a)–(b) First, we extract a collection of overlapping patches by scanning an input shape from multiple viewing directions. (c) We process each patch using a DEF network to obtain patch-based predictions, sensitive to interior feature lines only. (d) We leverage the multiple view stereopsis machinery to transfer distance-to-feature fields to adjacent views by reprojecting and linearly interpolating single-view predictions (warping-based view synthesis). (e) The final estimate of our field on a complete 3D model is obtained by a robust statistical inference procedure.

we found to perform similarly to predicting on patches of the size our network was trained on, while being more convenient.

Crucially for the completeness of the reconstruction, sufficient number of views of the input shape must be provided to capture most features; features not visible in at least one view are likely to be missed. We observed that for all of the considered 3D shapes, using $n_v = 128$ directions is sufficient to sample more than 97% of triangles of the corresponding meshes with at least 8 samples; we study the influence of the number of input views in Section 7.5. However, some shapes with many parts of their surfaces visible only from narrow cone of directions, different for each (*e.g.*, with many deep indentations) may require many additional directions.

Patch-Based Distance-to-Feature Estimation. Each patch I_i is processed independently using our neural network (Section 5.1), yielding predictions $\widehat{d}_i$ sensitive to interior feature curves, as shown in Figure 10 (c).

Transfer of Predictions across Patches. The aim of this stage is to gather predictions from multiple processed patches in each sampled point, integrating feature-sensitive information across the complete 3D shape. The central idea is to employ a warping-based view synthesis mechanism (similar to [Khot et al. 2019]): taking each pair of source and target views, we synthesize distance signal in the target view conditioned on the information inferred from the source view. Computational complexity of our distance estimation method depends on the number of sampled points in each view and (quadratically) on the number of views n_v.

Let a particular pair (s, t) of source and target views be represented by depth images I_s, I_t, their associated intrinsic K and extrinsic T_s, T_t matrices, and distance-to-feature estimate $\widehat{d}_s$ available in the source view; we seek to construct a warped signal $\widehat{d}_t^{s\to t}$ from this information. For each pixel $p = (u, v)$ in a target image I_t, we compute the warped coordinates $\widehat{p}$ in the source view by re-projecting p to the image plane of I_s:

$$\widehat{p} = K T_s^{-1} T_t \big(I_t(p) \cdot K^{-1} p\big).$$

To compute the warped distance-to-feature estimate $\widehat{d}_t^{s\to t}(p)$ at the target pixel p, we resample a local continuous distance field obtained by bilinearly interpolating $\widehat{d}_s$ on the grid of samples of the source patch I_s around the warped coordinates $\widehat{p}$:

$$\widehat{d}_t^{s\to t}(p) = \widehat{d}_s(\widehat{p}).$$

We additionally compute a binary visibility mask $v_t^{s\to t}(p)$ indicating which pixels have been correctly interpolated as some pixels have insufficient number of neighbors to resample from (see Supplementary material for details). The number of predictions for a pixel p is equal to the number of depth images from which the pixel is visible. Example interpolation results are shown in Figure 10 (d).

As a result, each 3D sample p captured by each depth image I_i is described by a set D_p of valid predictions interpolated from all views $\{I_s\}_{s=1}^{n_v}$:

$$D_p = \big\{d_s | d_s = \widehat{d}_i^{s\to i}(p) \text{ where } v_i^{s\to i}(p) = 1\big\}_{s=1}^{n_v}. \tag{4}$$

Inference of the Final Distance Field. The assembled predictions are processed to form a final distance estimate by feeding the set

D_p into an inference set-function g. We have considered a number of approaches to constructing g (we present an ablation study in Section 7.4); computing a minimum over all predictions of the distance $\widehat{d}(p) = \min_{d_s \in D_p} d_s$ proved to be the most accurate among all approaches we tried, which includes computing simple, robust, or truncated averages, variants of weighting schemes, and fitting a robust locally linear regression. More details on computing the variants of the inference function are presented in the Supplemental.

6 APPLICATION: EXTRACTION OF PARAMETRIC FEATURE CURVES

To evaluate the quality of distance-to-feature fields reconstructed using our method, we designed an algorithm for extracting parametric feature curve networks employing the estimated fields. Our algorithm is based on simple local classifiers for detecting corner vertices, heuristic graph structure analysis, and spline fitting. Making a number of careful choices, we are able to fit significantly more accurate feature curve networks compared to recent methods PIE-NET [Wang et al. 2020] and PC2WF [Liu et al. 2021].

A preliminary version of our method was presented in [Matveev et al. 2021]; similarly to the method described in this section, it uses DEF's distance-to-feature output to produce a set of feature curves. We keep the overall structure of the approach, re-use its segmentation and spline fitting steps, and follow the same stages as in the earlier work. However, we contribute an improved corner and curve endpoint detection criteria in (5), (7); a more robust kNN-based polyline construction stage and an optimization functional in (9); a post-processing technique in (11), all resulting in significant performance improvements of the method. We refer the reader to Figure 18 for qualitative demonstration of the difference between the two algorithms.

Initialization. At the initial stage, given a point cloud P, we select P_{sharp} that consists only of points with estimated distance $\widehat{d}$ less than d_{sharp}. To further reduce the number of points, we apply Poisson disk sampling, leaving only 10% of points to reduce the size of the set and make the point distribution more even.

Corner Detection. Corner detection is designed as an aggregation procedure of several corner estimates constructed from a grid of parameters. We sample anchor points across P_{sharp} (we use 20% of points in P_{sharp} chosen by farthest point sampling) and build sets B_i of points contained in overlapping Euclidean balls of a radius R_{corner} centered at the anchor points and covering P_{sharp}.

We approximate each of these local sets by an ellipsoid by computing PCA on points in the set and obtain vector of variances $(\lambda_1, \lambda_2, \lambda_3)$ such that $\lambda_1 \leqslant \lambda_2 \leqslant \lambda_3$ and $\sum_{k=1}^{3} \lambda_k = 1$, describing lengths of ellipsoid axes. For each specific set B_i, we use these vectors to compute a squared distance-normalized aggregate:

$$\Lambda_i = \sum_{k=1}^{3} \sum_{j \in \mathcal{N}_i} \left(\frac{\lambda_k^i - \lambda_k^j}{\delta_{ij}} \right)^2, \tag{5}$$

where $\mathcal{N}_i$ is a collection of indices of the sets B_j nearest to the set B_i, and δ_{ij} is a Euclidean distance between anchor points of sets B_i and B_j. This quantity measures how much a specific ellipsoid deviates from the neighboring ones.

We decide whether a local set B_i belongs to corner cluster by comparing Λ_i against the characteristic threshold T_{variance}, and mark B_i as either corner or curve type set:

$$\begin{aligned} \mathcal{B}_{\text{corner}} &= \{B_i \mid \Lambda_i > T_{\text{variance}}\}, \\ \mathcal{B}_{\text{curve}} &= \{B_i \mid \Lambda_i \leqslant T_{\text{variance}}\}. \end{aligned} \tag{6}$$

We evaluate this classification for all combinations of $\mathcal{N}_i$, T_{variance}, and R_{corner}, each varying over a small range, for a total of 60 combinations, and compute a probability of B_i to be a corner based on the fraction of corner classifications in this set. Refer to Section 7.3 for more details.

This value is available only for the anchor points of B_i. To extend it to the whole point cloud, we apply k nearest neighbors regressor with $k = 50$, thus obtaining per-point values $0 \leqslant w(p) \leqslant 1$.

The set of points near corners is obtained by thresholding weights:

$$P_{\text{corner}} = \{p \in P_{\text{sharp}} : w(p) > T_{\text{corner}}\}.$$

Curve and Corner Segmentation. For curve segmentation, we consider the set of corner points P_{corner} and the set $P_{\text{curve}} = P_{\text{sharp}} \setminus P_{\text{corner}}$ consisting of near-sharp points not detected as corners; we process both these sets to extract clusters defining individual corners and curves, respectively. To segment points belonging to individual curves, we construct a dense kNN graph by creating edges between all points in P_{curve} located within sampling distance r (3) from each other, and cut it into connected components. We treat each connected component as defining one of n_{curve} curves, together they constitute the set of point clusters corresponding to each curve:

$$\mathcal{P}_{\text{curve}} = \left\{ P_c \subseteq P_{\text{curve}} \mid \forall p \in P_c \, \exists q \in P_c, p \neq q : \|p - q\| \leqslant r \right\}_{c=1}^{n_{\text{curve}}}.$$

For corner points P_{corner}, the procedure is similar; we extract the final corner clusters $\mathcal{P}_{\text{corner}}$ by separating connected components of the detected corner sets.

Extraction of Curve Graph. From the segmentation, we construct a curve graph fitted to P_{sharp}, separately processing each set of points corresponding to a curve. The next steps include (1) detecting endpoints for each curve, marking curves as either open or closed based on the detections, (2) approximating each curve with a short path polyline, (3) connecting fitted polylines, corners, and endpoints into a complete shape curve graph, and (4) refining endpoint and corner locations.

To detect endpoints for a segmented curve cluster P_c, we construct a neighborhood-based endpoint detector similar to our corner detector. We construct Euclidean neighborhoods E_i with the radius R_{endpoint} centered at the anchor points p_{ai} sampled in P_c, compute their straight-line approximations (we compute PCA on points in E_i and reduce its dimensionality to one), and parameterize each point $p \in E_i$ by a single coordinate $t(p)$ obtained from PCA. To identify curve endpoints, we compute the share of points $p \in E_i$ whose parametric coordinates $t(p)$ are greater or smaller than the parametric coordinate t_{ai} of the anchor p_{ai}:

$$V_i = \left| \frac{1}{|E_i|} \sum_{p \in E_i} \operatorname{sign}(t(p) - t_{ai}) \right|, \tag{7}$$

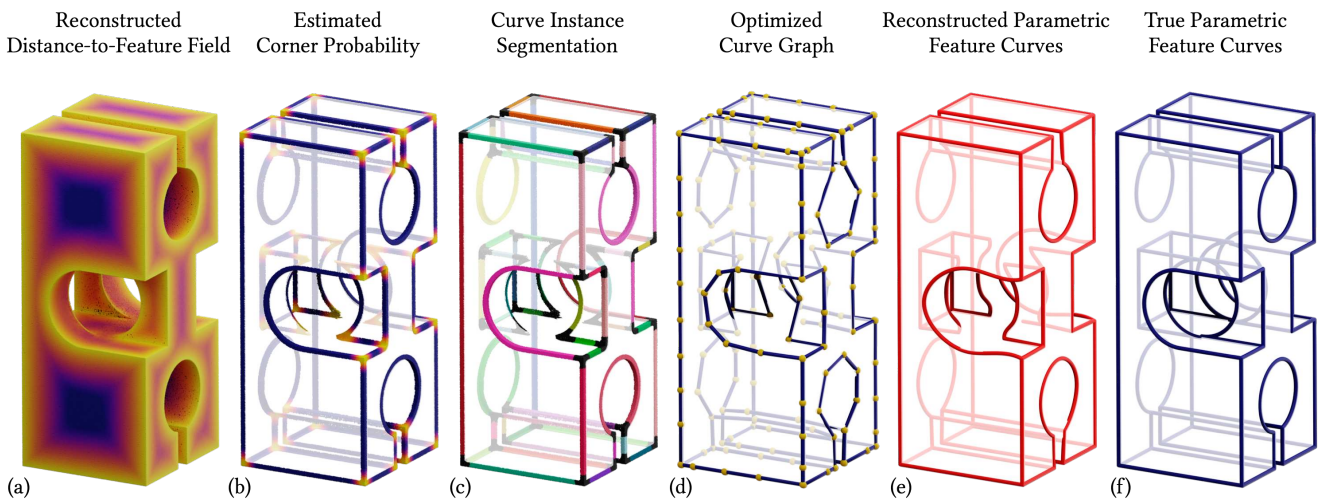

Fig. 11. (a) We propose a parametric curve extraction method based on an input dense point cloud with a per-point estimated distance-to-feature field. We threshold distances to obtain a subset of samples P_{sharp} that we use to (b) estimate corner probabilities and (c) construct curve instance segmentation (black clusters correspond to the detected corner neighborhoods). (d) Detected corners and curves allow building and optimizing a curve graph that reflects the curve connectivity. (e) We finally translate the curve graph into a set of accurate parametric curves that reflect feature geometry of the reference shape (f).

declaring p_{ai} an endpoint if V_i is greater than threshold T_{endpoint}. Intuitively, $V_i = 0$ corresponds to a fully symmetric case (equal shares of points parameterized by coordinates with either sign) while $V_i = 1$ indicates strong prevalence of points on either side of an anchor. For a curve cluster P_c, if only one such anchor exists, we select an anchor p_{ai} with the second largest value of V_i as a second endpoint; for more than two detected endpoints, we select the two most distant ones; if no such points are detected, the curve is considered to be closed.

Next, we compute polyline approximations of curves. For an open curve, we construct kNN graph by connecting all the curve anchor points p_{ai} sampled in P_c within twice the average sampling distance from each other, and initialize the polyline with a shortest path in such graph connecting the detected endpoints.

To create a polyline for a closed curve, we sample three points from the cluster by farthest point sampling, connect them to compose a triangle, and proceed with the subdivision strategy. The candidate subdivision points are identified by computing

$$p_{\text{split}} = \arg\max_{p_i \in P_c} \left| \widehat{d}_i - \|p_i - \min_l \pi^l(p_i)\| \right| \tag{8}$$

over points p_i from the current curve cluster $P_c \in \mathcal{P}_{\text{curve}}$, where $\min_l \pi^l(p_i)$ is a projection of p_i onto the nearest polyline segment l. To proceed with subdivision, we check an absolute difference between the estimates $\widehat{d}_i$ and the actual distances $\|p_i - \pi^l(p_i)\|$ against the threshold T_{split}; for candidate points p_{split} exceeding this value, we subdivide the polyline by assigning p_{split} a new polyline node and splitting the corresponding segment in two. This choice of p_{split} aims to keep the maximum polyline approximation error below T_{split} for individual curves.

Finally, we substitute the detected open curve endpoints with the respective nearest corner cluster centers, yielding a final curve graph $G(q, e)$ defined by the node positions q (corner cluster centers and nodes of polylines) and connections e between them. The last step is node position optimization:

$$\min_q \left(\frac{1}{|P_{\text{sharp}}|} \sum_{p \in P_{\text{sharp}}} |\widehat{d}(p) - \|p - \pi^{G(q,e)}(p)\|| - \sum_{\overline{q} \in l[G(q,e)]} \cos \overline{q} \right), \tag{9}$$

where $\pi^G(p)$ is the projection of a point p onto the nearest edge in the curve graph G, and $\sum_{\overline{q} \in l[G(q,e)]} \cos \overline{q}$ is sum of cosines of angles between the two consecutive edges incident to the node $\overline{q}$, computed only for the set of nodes $l[G(q, e)]$ such that they have exactly two incident edges (hence, it is locally linear). Intuitively, the second term represents rigidity of polylines that prevents the acute angles between edges. Optimization helps to position graph nodes more accurately, especially at the intersections of multiple feature curves, and the rigidity term makes polyline segments more straight. After this step is finished, we can identify the final corner positions as coordinates of graph nodes with more than two incident segments.

Spline Fitting and Optimization. For spline fitting one needs to obtain a consistent parameterization of each feature curve. We do that by partitioning the curve graph into shortest paths between graph nodes with degree not equal to 2, each path serving as a proxy to a curve that defines parameter coordinates of points along feature curve. For a path g represented as a sequence of graph nodes $q_g = \{q_i\}_{i=1}^{|g|}$ we get a set of nearest points $P_g \in P_{\text{sharp}}$, and compute projections $\pi^g(p_i)$, $p_i \in P_g$ and obtain values of parameters $u_g = \{u_i\}_{i=1}^{|P_g|}$ as a cumulative sum of norms of $\pi^g(p_i)$ along the path g. Simultaneously, we compute knots t_g as evenly spaced parameters; number of knots is defined as $\max\left(5, \frac{|g|}{2}\right)$.

Fitting a spline s_g to the path g results in a set of control points c_s that define the exact shape of the spline curve. Once the spline is

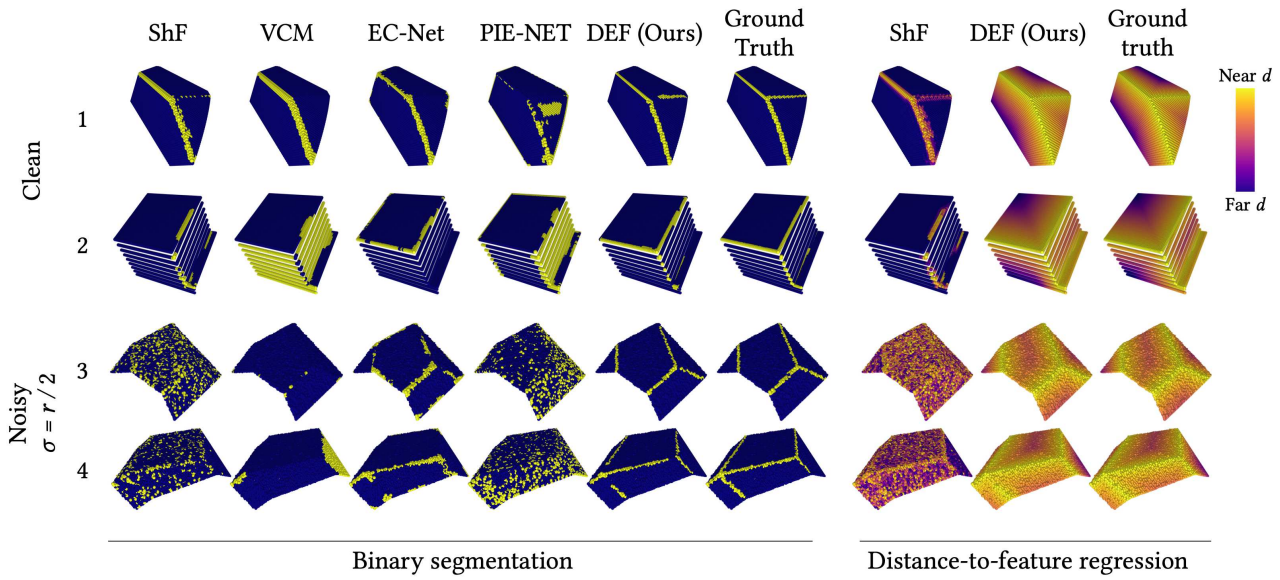

Fig. 12. Visual comparison of DEF vs. competitor approaches on challenging image patch instances (synthetic image patches, $n = 50$, $r = 0.02$). Observe that, for segmentation (left part of gallery), VCM struggles to detect subtle features (rows 1, 3) and leads to substantial amounts of false positives when encountering large density variations or noisy inputs (rows 2, 4); EC-Net likewise tends to miss features (rows 1–2) and yield overall unstable predictions in presence of noise (rows 3–4). Most evidently, ShF and PIE-NET deteriorate drastically in presence of noise (see rows 3–4) while producing imperfect predictions for clean data. Additionally, PIE-NET, EC-Net, and VCM were not designed to estimate distances to nearest sharp edges (right gallery part); the only previous method for predicting distances, ShF, shows extreme sensitivity to sampling and noise (rows 1–4). In contrast to most competitor methods, our deep models are able to accurately perform segmentation and robustly estimate distance-to-feature fields; DEF successfully survives non-uniform, irregular, or noisy sampling patterns, remaining sensitive to less pronounced features.

fitted, we can evaluate points $P_s(c_s) = \gamma(u_g, P_g, t_g, c_s)$ on the spline curve s_g. These points, ideally, should be precisely as far away from point cloud points P_g as a distance field $\widehat{d}$ suggests. To enforce this property, we optimize over control points to shape the spline to the distance values:

$$\min_{c} \sum_{i=1}^{|P_g|} \left(\widehat{d}_i - \|p_i - \gamma(u_i, p_i, t_i, c)\| \right)^2, \tag{10}$$

where $p_i \in P_g$, $\widehat{d}_i$ is a corresponding distance value, and $\gamma(u_i, p_i, t_i, c)$ is a point corresponding to p_i evaluated on the spline s_g. Additionally, we impose constraints on the spline endpoints to match the polyline endpoints.

The optimization problem and constraints are similar for the closed curves: endpoints of the spline should meet at the same point, and the tangents at the endpoint positions should be equal.

Spline Post-Processing. To improve the final result, we apply post-processing procedure that helps to keep only the curves that have a good fit. First, we compute the quality metric as an F_1 score of the Chamfer distances between sampled curves and P_{sharp} and vice versa, thus getting the fit quality. Second, we turn off each curve separately and compute the metric again. If the quality drops or stays the same, we keep the curve in the final set of curves. Otherwise, we eliminate that curve. The quality metric is given by:

$$\begin{aligned} \text{CD}_{X \to Y} &= \frac{1}{N_X} \sum_{x \in X} \inf_{y \in Y} \|x - y\|^2, \\ F_1(T_{\text{metric}}) &= \frac{2 \cdot \mathbb{1}(\text{CD}_{X \to Y} \leqslant T_{\text{metric}}) \cdot \mathbb{1}(\text{CD}_{Y \to X} \leqslant T_{\text{metric}})}{\mathbb{1}(\text{CD}_{X \to Y} \leqslant T_{\text{metric}}) + \mathbb{1}(\text{CD}_{Y \to X} \leqslant T_{\text{metric}})}, \end{aligned} \tag{11}$$

where $\text{CD}_{X \to Y}$ is a Chamfer distance from point set X to point set Y, $\mathbb{1}$ is an indicator function, and T_{metric} is a threshold to convert the real-valued distances into 0-1 hard labels. When using this metric for post-processing, we assigned P_{sharp} as one of the point sets, and a discretized set of curves as another.

Finally, we apply filtering of curves based on their length. This includes detecting the connected sets of curves, for each set we count the number of curves that form it and compute the total length of all curves in it. If the set contains less than four curves with total length smaller than $20r$, we discard such set altogether.

Our method requires setting the following parameters: threshold on distances for selection of points near feature lines d_{sharp}, corner detector threshold T_{corner}, endpoint detector radius R_{endpoint}, endpoint detector threshold T_{endpoint}, polyline optimization threshold T_{split}. We express all of the parameters in the scale of sampling distance r (3). We discuss the exact values of parameters in Supplementary material.

For the illustration of the vectorization pipeline and the results of our spline fitting procedure, refer to Figure 11 and Figure 17.

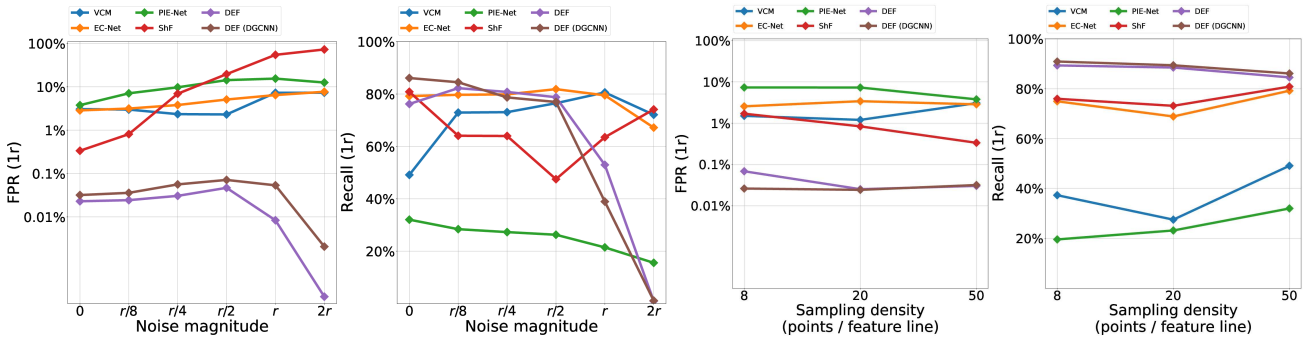

Fig. 13. DEF is significantly more robust to acquisition noise, compared to other approaches (the two left plots). Compared to the baseline approaches, DEF is robust to feature sampling density (the two right plots).

7 EXPERIMENTS

We start our experimental study by introducing the measures of quality and providing training details in Section 7.1. We further evaluate our models against prior art in a variety of synthetic and real-world settings in Section 7.2. Section 7.3 demonstrates a parametric curve extraction application. We investigate alternative choices of model architecture and training configurations in Section 7.4. We conclude with testing the robustness of our approach w.r.t. sampling patterns and density variations in Section 7.5.

7.1 Experimental Setup

Measures of Quality. We evaluate our feature estimation method in terms of several quality measures (distance-to-curve regression and segmentation, as both are relevant in our case). We compute the following measures to assess feature estimation performance:

- *RMSE*: the root mean squared error between the predicted distances $\widehat{d}(p)$ and the ground-truth distance-to-feature field $d(p)$. For a set of instances, we report the mean RMSE across the respective items.
- *RMSE-q_{95}*: the 95% quantile value of RMSE across a set of instances captures the width of distance error distribution.
- *Recall* (T): we compute Recall using the predicted thresholded labels $\widehat{s}_i = \mathbb{1}(\widehat{d}_i < T)$ and the ground-truth distances $s_i = \mathbb{1}(d_i < T)$. We use $T_{\text{sim}} = r$ for synthetic instances but increase the threshold for real data to $T_{\text{scan}} = 4r$ to account for scan misalignments. Recall estimates the quality of feature line estimation *in the direct proximity* of the ground-truth feature line. As before, we report the mean value of Recall computed across test instances.
- *FPR* (T): we compute the False Positives Rate using the thresholded predictions and report mean FPR across patches or full models. FPR estimates the fraction of points predicted as belonging to a sharp feature line but located *outside the direct proximity* of the ground-truth feature line.
- *CD*, *HD* and *SD*: We use *Chamfer Distance*, *Hausdorff Distance* and *Sinkhorn Distance*, respectively, for evaluating parametric curve extraction. These measures assess the discrepancy between the extracted and the ground-truth sets of curves.

We provide the exact formulae for our quality measures in Supplementary material. Unless specified otherwise, we present measure values averaged across test instances (patches or full models).

Data and Training. We train networks on 4 nVidia Tesla V100 16Gb GPUs in parallel; we use the synchronous version of batch normalization in all our architectures. All experiments were performed using the PyTorch framework [Paszke et al. 2019], its higher-level neural network API PyTorch Lightning [Falcon 2019], and the Hydra framework [Yadan 2019] for configuring experiments. We use Adam optimizer [Kingma and Ba 2014] with an initial learning rate of 0.001, multiplying it by 0.9 every epoch, and train all our models with a total batch size of 32. We validate network performance on a validation set of patches every epoch, stopping training when the RMSE metric has no improvement over the ten consecutive epochs, and select the model with the best performance on the validation set of patches.

All training patches consist of 4096 (64×64) pixels. We divide depth values in each patch by the 95% quantile value computed among max depths for each patch across the training dataset; no augmentations were applied to depth images. Unless specified otherwise, our training datasets consist of 65,536 patches. The validation set and test set include approximately 32,000 patches. We observed that increasing the size of the training set further does not lead to significant improvement in performance, and report more details in the Supplementary material.

7.2 Comparisons

Baseline Approaches. We compare DEF against five state-of-the-art methods either directly designed or adapted for extracting feature lines from sampled 3D shapes. Four of these methods are deep learning-based, representing natural interest for comparisons [Liu et al. 2021; Raina et al. 2019; Wang et al. 2020; Yu et al. 2018]; the fifth method is the best-performing traditional approach based on local set-based feature detection [Mérigot et al. 2010] (see Section 2 for more context). We briefly review the main principles underlying these approaches below. Most competitor methods have a number of tunable parameters, commonly adjusted to obtain the best results for a specific input shape; as we aim to compare on relatively large datasets, we determine fixed parameters that maximize

method performance on the whole validation set, as explained in the Supplemental; to obtain predictions, we run each method with the selected set of its parameters on both local patches and complete point-sampled 3D shapes.

Voronoi Covariance Measure (VCM) [Mérigot et al. 2010] is a non-learning method for hard segmentation of a point cloud into sharp and non-sharp points. For this, *VCM* computes the Voronoi covariance measure of a point as a covariance matrix of the intersection of an estimated Voronoi cell with a ball of radius R, where R is a parameter of the method; a convolution radius ρ is used for smoothing the measure. The input points are labelled by thresholding the ratio of the smoothed covariance matrix's eigenvalues, with threshold T being another parameter. We have optimized the parameters (ρ, R, T) to maximize Recall$(1r)$ on each dataset, by a direct search, for each data variety. *VCM* is expected to perform robustly across a range of noise and sampling variations.

Sharpness Fields (ShF) [Raina et al. 2019] is a CNN for predicting the sharpness field — a real-valued function with values close to 1 for points near the feature lines and 0 in smooth areas. To this end, *ShF* constructs local neighborhoods with fixed-size (30×30), uniformly spaced points sampled from the underlying Moving Least Squares proxy surface of the point cloud. The method requires normals as an additional input, that we estimate using a neighborhood-based method with the number of neighbors empirically set to 100. *ShF* accepts a noise-free, uniformly sampled point cloud as input, thus, we expected its performance to deteriorate for noisy inputs. We have observed that, in most cases, predicted values do not increase monotonically with distance to the feature line; however, the predicted field is suitable for producing segmentation by thresholding; we thus run a sweep to select the threshold value that would produce the highest Recall on the training set. We also made an effort to compare our distance-to-feature field outputs to the sharpness fields produced by *ShF* directly: to that end, we find the most suitable linear transformation of our field on the train subset.

Edge-Aware Consolidation Network (EC-Net) [Yu et al. 2018] includes a PointNet++ [Qi et al. 2017] derived method for detection of sharp feature lines as an auxiliary signal for point cloud upsampling. The network predicts point locations exactly on the sharp feature curves; we map this output to our patches by selecting one nearest neighbor for each of the sharp points from *EC-Net,* resulting in a hard segmentation-like output. In our comparisons, we use the original pretrained model, that was trained on sampled patches with an additive noise, possibly making it robust to noise variations of the kind we use for evaluation.

PIE-NET [Wang et al. 2020] has a two-stage prediction pipeline which (1) segments sharp feature curves and corner points using a PointNet++ architecture [Qi et al. 2017] and (2) generates parametric curve proposals using a separate network, refining these using an optimization approach. *PIE-NET* expects a noise-free, uniform sample with 8,096 points representing a complete 3D shape, moreover, samples are expected to land exactly on the sharp feature lines; for these reasons, *PIE-NET* is unlikely to perform robustly on most of our datasets. We use their pre-trained models to both segment points lying in the proximity of the sharp feature curve and extract parametric curves in the form of their point samples.

PC2WF [Liu et al. 2021] is a learning-based approach to infer parametric sharp feature lines, assuming only straight lines segments are present. From an input point cloud, possibly noisy, *PC2WF* detects corner points and infers edge segments connecting these corners; the method is able to process relatively large point sets of up to 200,000 points. *PC2WF* was not designed to detect sharp features in point clouds, so we compare the wireframe extraction quality only. We use their pre-trained models.

Wireframes [Matveev et al. 2021] is an earlier version of our parametric curve extraction pipeline. It accepts the same input as our current vectorization method, a point cloud of arbitrary size with per-point distance-to-feature estimates from DEF neural network. Although *Wireframes* share the overall structure with our current method, previous approach has major flaws in its design which we have resolved in the current method.

Patch-Based Comparison (DEF-Sim). We start with comparisons to prior art by evaluating DEF vs. the baselines using our synthetic *patch datasets (DEF-Sim)* to provide a direct network-to-network comparison and eliminate the influence of postprocessing. We present a statistical evaluation in Table 2, compare results visually in Figure 12, and plot dependencies of performance vs. noise and resolution parameters for all methods in Figure 13.

Qualitatively, we observe that our method compares favorably to all competitors (most evidently, *ShF, VCM,* and *PIE-NET*) on less pronounced features that have smaller normal jumps (Figure 12, rows 1,3); while these methods tend to be less sensitive to such subtle features, DEF demonstrates increased robustness when facing such geometry. For instances with large sampling distance variations (Figure 12, row 2), *ShF* and *EC-Net* miss features while *VCM* and *PIE-NET* produce substantial numbers of false positive, particularly in under-sampled regions; for *VCM,* this is due to the uniform surface sampling assumed in the model; DEF remains capable of accurately localizing feature locations. In comparison with *ShF* and *PIE-NET,* DEF performs notably better on noisy data for noise magnitudes of up to $r/2$, with a moderate decrease in Recall but almost no change in FPR, compared to two orders of magnitude increase in FPR from

Table 2. Our *local patch-based* networks for *distance-to-feature estimation* and feature line *segmentation* are more effective compared to competitor methods across a variety of segmentation and regression quality measures (evaluated on synthetic image patches, $n = 50, r = 0.02$).

Method	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall $(1r)$, %↑	FPR $(1r)$, %↓
Evaluation using DEF-Sim datasets				
VCM [Mérigot et al. 2010]	—	—	49.1	3.1
EC-Net [Yu et al. 2018]	—	—	79.2	2.9
DEF *(Trained on EC data)*	124.1	501.1	56.0	0.15
PIE-NET [Wang et al. 2020]	—	—	32.0	3.8
DEF *(Trained on PIE data)*	86.2	451.8	57.1	0.1
ShF [Raina et al. 2019]	18.0	95.7	**80.9**	0.3
DEF (Ours)	**11.1**	**42.5**	80.02	**0.02**
Evaluation using EC-Net datasets				
DEF *(Trained on EC data)*	192.9	573.1	46.3	1.5
DEF (Ours)	**153.0**	**526.1**	46.4	**1.3**

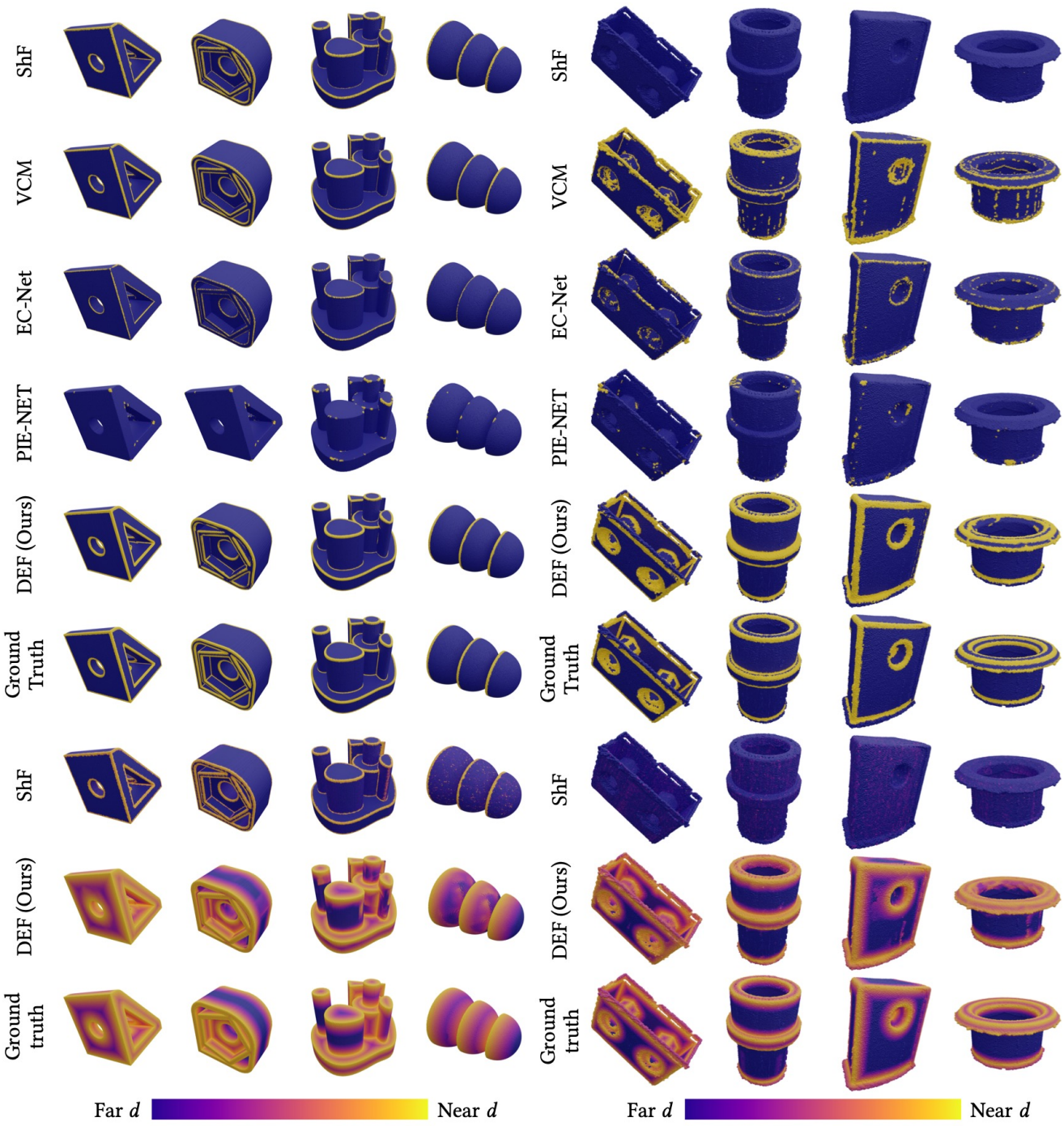

Fig. 14. Comparison to state-of-the-art sharp feature line estimation methods on high-resolution synthetic full shape datasets (a) and real scanned datasets representing full 3D shapes (b). Our method is able to robustly reconstruct a pointwise distance-to-feature field and scales to 3D shapes represented by millions of points.

Table 3. Our method is able to reconstruct a robust estimate of a distance-to-feature field defined for a complete 3D shape. While DEF achieves similar Recall to *VCM*, it does so by truncating an accurate distance field and demonstrates nearly 10× lower FPR.
* *PIE-NET* was invoked with 8,096 samples as input.

Method	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
VCM [Mérigot et al. 2010]	—	—	**79.2**	4.8
EC-Net [Yu et al. 2018]	—	—	48.5	**0.2**
PIE-NET* [Wang et al. 2020]	—	—	73.6	2.9
ShF [Raina et al. 2019]	623	761.4	69.8	0.3
DEF (Ours)	**115.1**	**200.1**	79.0	0.5

Table 4. Compared to the closest state-of-the-art competitor approach, *VCM*, our method achieves 3× higher Recall ($4r$) on noisy and incomplete scanned data, while maintaining a moderate FPR ($4r$). Quantitatively, our method reconstructs the full distance-to-feature field with RMSE = 1.5 mm and RMSE-q_{95} = 2.9 mm at a sampling distance of r = 0.5 mm.

Method	Recall (2 mm), %↑	FPR (2 mm), %↓
VCM [Mérigot et al. 2010]	29.5	10.2
EC-Net [Yu et al. 2018]	10.1	**0.8**
DEF (Ours)	**91.7**	20.1

0.33% to 19% for *ShF* (Figure 13, left two plots). This leads to the results of these methods being unusable for noisy point clouds, see Figure 12; however, such results are expected as *ShF* and *PIE-NET* models that we used were not optimized on noisy datasets. For varying sampling distance values, DEF still compares favorably according to Recall and FPR measures (Figure 13, right two plots).

We made an effort to train our algorithm using the datasets described in [Wang et al. 2020; Yu et al. 2018] to ensure conformity in terms of training sets and input-output requirements. For the EC-Net dataset, we use the original 32 mesh files and feature annotations; to create a PIE-NET-like dataset, we select meshes with up to 30,000 vertices containing only Line, Circle, or BSpline curves; in each case, we generate a dataset of 65,536 images for training our method using the pipeline from Sections 4.1–4.2. We present results in Table 2. Evaluation using DEF-Sim datasets indicate that our method performs significantly better than *PIE-NET*; compared to *EC-Net*, our network keeps having 10× lower FPR but delivers less accurate distance predictions; this is likely due to a low geometric diversity of training data: the volume of the *EC-Net* dataset is two orders of magnitude lower compared to our datasets.

Complete 3D Models (DEF-Sim). To obtain results on complete models, we use DEF-Sim, the synthetic validation set of 68 sampled 3D shapes (see Section 4.1), and apply our patch-based DEF to each view of each shape without any fine-tuning on these data. We further reconstruct a complete, object-level distance-to-feature field using the algorithm described in Section 5.2; for our fusion, we use $n_v = 128$ views and perform view synthesis in orthographic projection using 4 neighbors for each sampled point. To obtain the final statistical estimate, we extract minimum value from the set of valid interpolated predictions in (4).

We compare our approach with competitors statistically in Table 3 and visually in Figure 14 (a). Most our complete 3D shapes include from 10^6 to 10^7 point samples. Qualitatively, our method is able to more robustly regress features with smaller difference in normal orientations, undersampled features, or feature curves with large density variations across the feature line, such as features in internal cavities of a 3D shape.

In Figure 15, we additionally demonstrate an example reconstruction of a complete object-level distance field using DEF trained on patches in the EC-Net dataset described above.

Real 3D Shapes (DEF-Scan). To perform an experimental evaluation of distance-to-feature prediction quality for real-world noisy 3D scans, we use our real-world dataset of complete 3D scanned shapes with sharp feature annotations. We first select a DEF CNN model pre-trained on a synthetic dataset (with sampling distance $r_{\text{med}} = 0.05$) and fine-tune it using the real annotated depth images. To this end, we split the 84 scanned objects into training (42 objects, 981 scans), validation (21 shapes, 479 scans), and final testing (21 objects, 468 scans) subsets, and optimize our model until convergence on the validation set. Next, we apply the optimized network to each view of the testing dataset and reconstruct a complete distance-to-feature field using our fusion algorithm (Section 5.2) using $n_v = 12$ views available for each 3D shape; here we perform view synthesis in perspective projection using 4 neighbors for each sampled point.

Overall, our method reconstructs the complete distance field with RMSE = 1.5 mm and RMSE-q_{95} = 2.9 mm. We report performance against competitor approaches in Table 4 using Recall ($4r$) and FPR ($4r$) measures where the real-world sampling distance r = 0.4 mm. Compared to *VCM* and *EC-Net*, our results suggest that DEF systematically outperforms the competitor methods by a significant

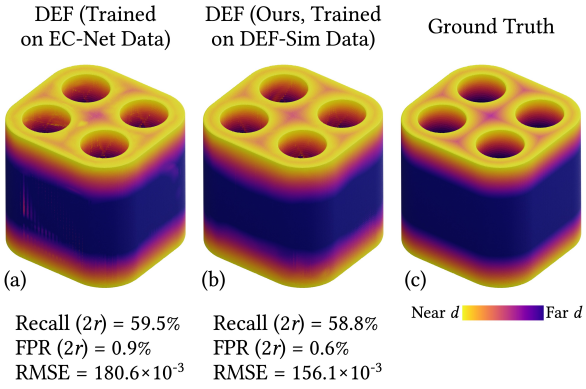

Fig. 15. Our method is able to leverage various feature-annotated training collections. A complete object-level field then can be reconstructed from predictions by a model pre-trained on (a) the *EC-Net* dataset [Yu et al. 2018] and (b) our DEF-Sim dataset (see Section 7.2). As our data is two orders of magnitude larger in size, predictions obtained using our model are generally more accurate.

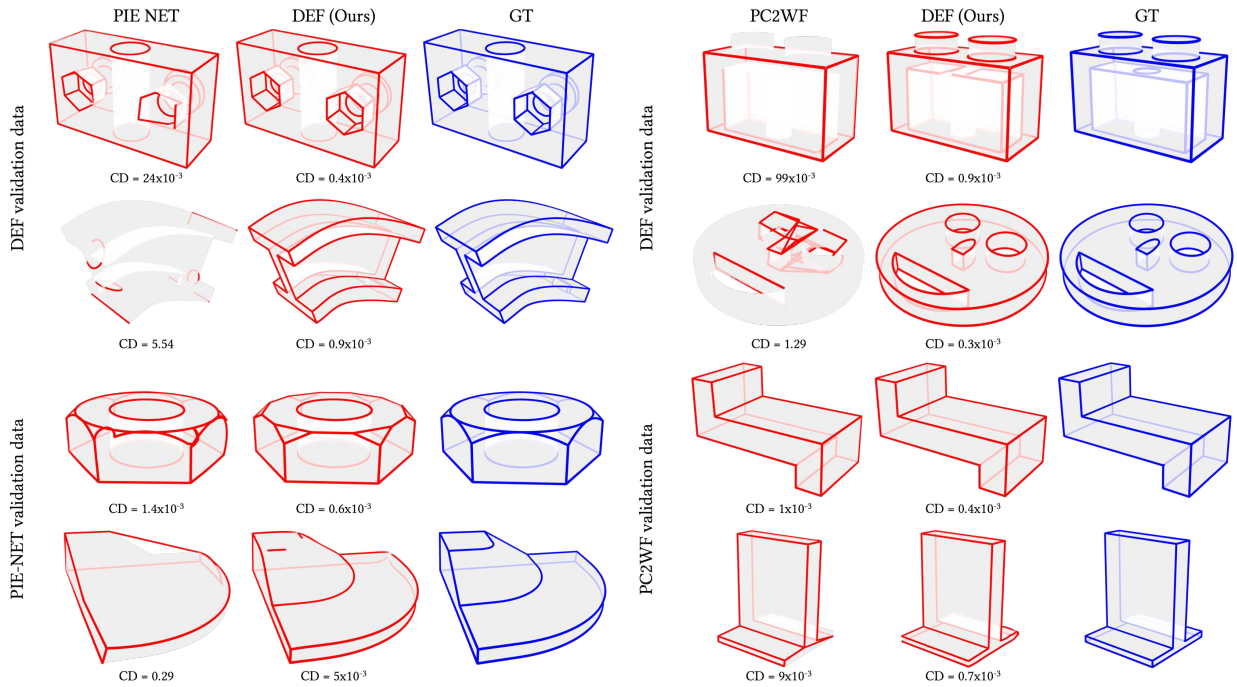

Fig. 16. We use distance field estimates obtained by our method for complete, large sampled shapes (up to 10^7 points) to reconstruct full parametrizations of their feature curves. We compare our inference results to PIE-NET (a) and PC2WF (b) using our validation set (rows 1–2) and on validation shapes from the corresponding papers (rows 3–4).

margin (*e.g.*, DEF achieves 3× higher Recall compared to the best-performing competitor method, *VCM*); the methods *ShF* and *PIE-NET* produced little to no sharpness detections for all shapes that we have used. These observations are also reflected in qualitative results in Figure 14 (b).

7.3 Extracting Parametric Curves

We run our vectorization method on the complete 3D shapes sampled using $n_v = 128$ views, where predictions have been computed by the DEF network and a complete object-level distance field has been obtained in the previous steps (Section 7.2). After setting parameters, we run our method without manual intervention. The output consists of (1) spline curve parameters and (2) endpoint coordinates for straight lines, readily available for further processing.

PIE-NET [Wang et al. 2020] requires subsampling our point clouds to 8,096 points. We applied the farthest point sampling technique to reduce the size of the point clouds. *PIE-NET* parametric curves extraction stage produces a set of points sampled along the curves.

Table 5. Compared to *PIE-NET* parametric feature curve extraction stage, DEF achieves an order of magnitude more accurate reconstruction.

Method	CD ↓	HD ↓	SD ↓
PIE-NET [Wang et al. 2020]	0.97	2.19	0.84
DEF (Ours)	**0.04**	**0.55**	**0.05**

PC2WF [Liu et al. 2021] is essentially free of point cloud size; however, to reduce the computation time and make sampling density closer to the point clouds of the original paper, we subsampled our shapes to 200,000 points each. *PC2WF* outputs pairs of endpoint coordinates that represent a straight line wireframe.

Wireframes [Matveev et al. 2021] has the same input and output as our method.

To assess the wireframe quality, we ran our pipeline on the validation set of 68 complete 3D models (DEF-Sim) along with *PIE-NET* and compared the obtained results to the ground truth parametric curves. To compute the metrics, we sampled all the predicted curves and lines along with the ground truth set of curves into point sets and derived distances between the closest points to calculate CD, HD, and SD. The aggregated statistical estimation of metrics for our method and *PIE-NET* are reported in Table 5. We observed a significant difference between one-sided CD's for *PIE-NET* predictions. Specifically, the average distance from ground truth to prediction is 0.9, the average distance from prediction to ground truth is 0.064. That implies that *PIE-NET* misses many curve instances, but it outputs relatively accurate reconstructions for the detected ones. In turn, the one-sided CD of our method is 0.024 from ground truth to prediction and 0.02 for distance in the opposite direction. We refer the reader to Figure 16 for the qualitative results.

Since *PC2WF* outputs straight lines only, we did not run it on the whole set of validation shapes and report no statistical performance; instead, we provide qualitative results for their method only on the small subset of shapes presented in Figure 16.

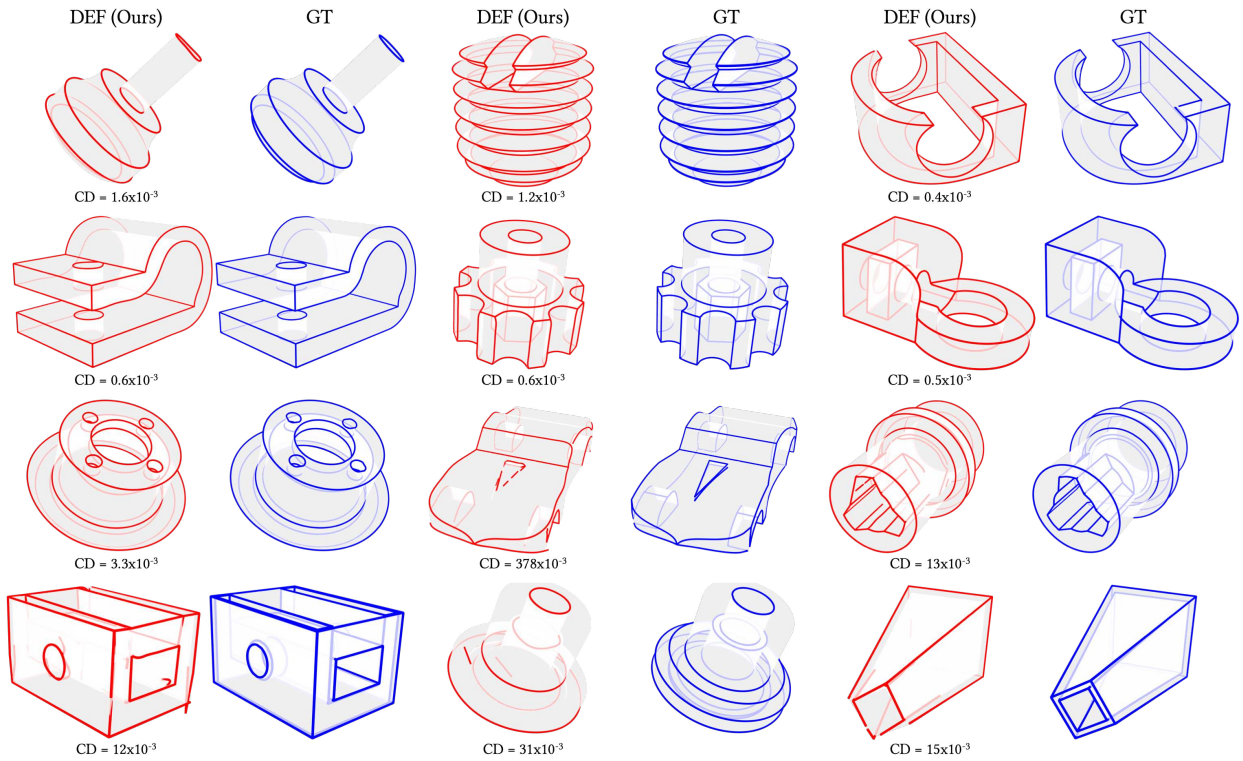

Fig. 17. We showcase twelve additional examples of extracted parametric representations next to the ground truth sets of curves. Row 4 includes visually inferior examples where our method struggles to output clean and complete parametric representation.

For both *PIE-NET* and *PC2WF*, qualitative results depict the shapes from our validation set and figures from the respective papers that were used to evaluate the quality of the corresponding methods.

Results indicate that our method is more flexible and robust with respect to the shape sampling variation and geometric complexity.

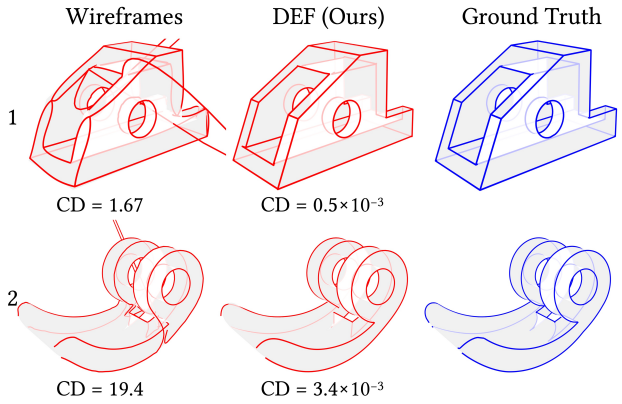

Fig. 18. Our current pipeline improves corner detection (row 1) and is able to resolve complex curves (row 2), whereas *Wireframes* outputs imprecise curve graphs that lead to outlier curves with extreme variation.

Compared to *PIE-NET*, DEF detects more curve instances, and due to the predicted distance field, the fitting procedure does not rely solely on the point positions and is free of sampling issues. Our pipeline can fit curves of different types when *PC2WF* has been designed for straight lines. On the other hand, the performance of our method is strongly conditioned by the choice of parameters when both *PIE-NET* and *PC2WF*, as learning-based methods, are almost free of parameter tuning. We described a simple tuning procedure that only exploits the distance field estimation to mitigate that.

Additionally, we demonstrate how our current vectorization pipeline compares to the previous version (*Wireframes*). We compare the two methods in Figure 18. The improved corner detection and kNN-based polyline construction enable our method to resolve cases of close corners and complex curves. Curve graph topology guides the curve fitting stage and, if imprecise, may lead to outlier curves as it is seen in the *Wireframes* output.

7.4 Ablation Studies

We conducted a large number of computational experiments to determine the optimal parameters of our method; our main conclusions were outlined in Section 5; here, we summarize the results of the studies supporting these conclusions. We present a separate stress-test to explore the robustness of our approach in Section 7.5.

Learning Architectures. In this paper, our focus is on 3D data represented as a collection of depth images, one of the most common

Table 6. Compared to point-based DGCNN [Wang et al. 2019], our CNN-based learning method more efficiently regresses distance-to-feature values. For image-sampled patches that tend to be non-uniform, adding prior sharpness estimates from *VCM* yields no advantage to either method.

Dataset	Method	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
Regular images (no bg, reprojected to points)	DGCNN + Histogram loss	11.3	55.5	80.9	3.7×10^{-2}
Regular images (no bg, reprojected to points)	DGCNN + Histogram loss + VCM	13.6	70.0	68.8	4.8×10^{-2}
Regular images (no bg)	CNN + Histogram loss (DEF)	**9.7**	**32.5**	**84.6**	$\mathbf{3 \times 10^{-2}}$
Regular images (no bg)	CNN + Histogram loss + VCM	10.9	36.8	80.4	3.7×10^{-2}
Regular images (with bg, DEF-Sim)	CNN + Histogram loss (DEF)	11.1	42.5	80.0	2.2×10^{-2}

types of scanned 3D data. this allows us to use standard convolutional networks that take advantage of the regular sampling pattern in the data. To quantify the advantage obtained from using this additional regularity of sampling, we consider an alternative approach, ignoring depth image structure, and viewing the collection of images as an unstructured point set. As standard CNNs can no longer be applied to this type of data, we use the DGCNN network [Wang et al. 2019]; we set depth $D = 6$ and width $W = 64 \times 1.35^{D-3} \approx 150$. Similarly to the CNN version, we trained the network using the Histogram loss, studying various modifications, most importantly, training the DGCNN using the ground-truth distances $d(p)$ *and* VCM sharpness labels as an additional input.

For highly non-uniform image-sampled patches (*e.g.*, rays passing nearly in parallel to parts of the surface), VCM struggles to extract feature-related information. Thus, adding VCM labels yields no advantage for range-scan data for both the DGCNN and the CNN DEF models. Generally, we observe DEF networks to outperform point-based models (DGCNN trained with Histogram loss supervised by $d(p)$ *and* VCM) on regularly sampled range-scan data, see Table 6, middle rows. CNN DEF models additionally demonstrate better noise-resistance compared to the point-based alternative, as can be seen in Figure 13. In this experiment, we train CNN DEF and DGCNN models on noisy sampled data, and find that the latter yields lower Recall and higher FPR values across noise magnitudes.

Data Generation. We mention an additional configuration of interest, obtained by considering two versions of the range-scan data: a filtered version that excludes patches with depth discontinuities or background pixels (we refer to it as *no bg*), and a dataset including all types of patches (referred to as *with bg*); we train models separately on either data variety. DEF models trained on patch datasets without background pixels perform quantitatively better for similar testing data, see Table 6, bottom rows; however, as shown in Figure 19, networks trained on data with background pixels yield more stable predictions, particularly on near-boundary pixels.

Loss Type. (Section 5.1). The results of our study of loss functions lead us to find the Histogram loss [Imani and White 2018] to perform favorably compared to L_1/L_2 losses (see Table 1).

Reconstruction on Complete 3D Models. We investigate the two crucial factors in the reconstruction of distance-to-feature fields om complete sampled 3D shapes: the number of views n_v and the inference function applied over the set D_p of interpolated predictions

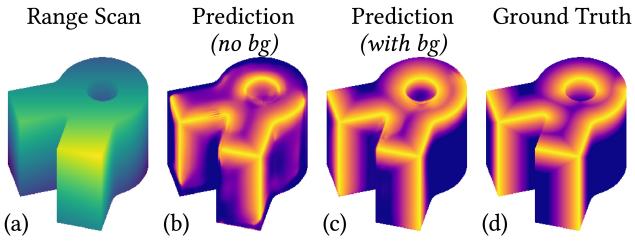

Fig. 19. We opt for training on instances with background and depth discontinuities (*with bg*, (c)); excluding these (*no bg*, (b)) yields suboptimal predictions, particularly near patch boundaries.

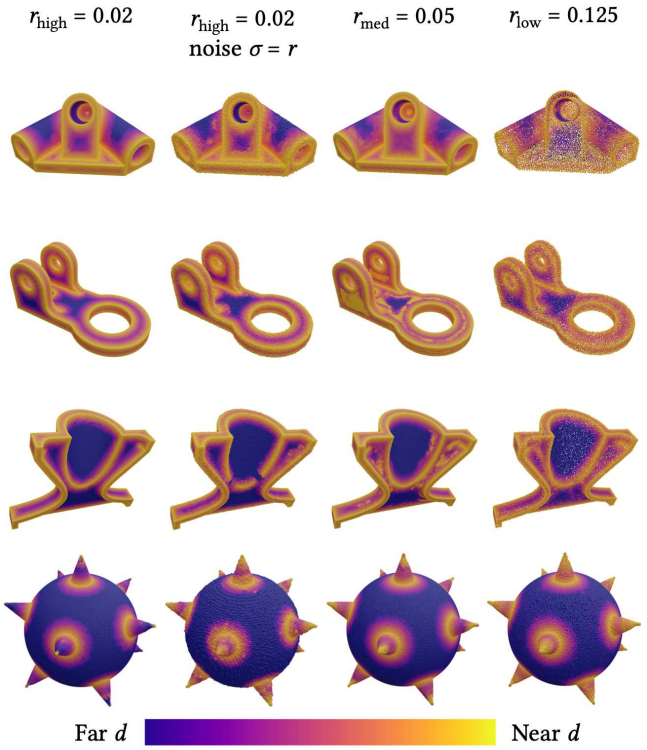

Fig. 20. Our approach is able to withstand (b) high noise magnitudes and (a), (c), (d) large variations in sampling density.

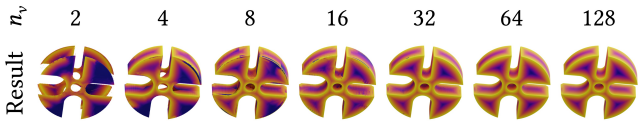

Fig. 21. We experimentally observe our method to benefit from increasing the number of views used during fusion. For this synthetic shape, $n_v = 18$ projections give an approximate Recall of 90%.

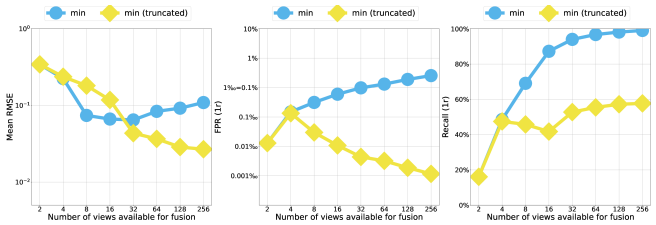

Fig. 22. Qualitatively, reconstructing distance-to-feature field on a complete 3D shape is able to detect the vast majority of features with around $n_v = 16$ views; increasing the number of views to $n_v = 32, 64$, or 128 refines and stabilizes these detections.

in (4). To this end, we consider an order of magnitude fewer set of $n_v = 18$ views and two additional inference functions: *truncated min* and *linear fit*, as well as compare against an aggregation method applied on top of DGCNN predictions. *Truncated min* is computed by removing 20% of smallest values in D_p and taking min; *linear fit* fits a robust version of local linear regression [Huber et al. 1973] to $d(p)$ in each sampled point p by extracting local patches of Euclidean neighbors of size 50, and computes the final estimate as a fitted value in p.

Statistical results for our sets of 68 synthetic and 84 real scanned models are presented in Table 7. We focus our attention on the Recall and RMSE measures and conclude that having a sufficient number of views is crucial to the successful reconstruction of our distance function. Comparisons of inference functions generally lead to *truncated min* improving over RMSE but not Recall measure compared to *min*, with *linear fit* being inferior to both these approaches.

7.5 Robustness Study

Noise and Sampling Sensitivity. We examine the noise sensitivity of our method by training DEF CNNs on datasets with increasing noise levels and coarse sampling, and using these in reconstructing distance fields on complete 3D models. We vary the noise magnitude from 0 up to $2r$, where r is sampling distance. Performance of the networks in isolation drops moderately as noise magnitude rises, as seen in Figure 13; the models show particular robustness to sampling distance variations, indicating weak influence of sampling on performance. Figure 20 demonstrates qualitative reconstruction results for a number of 3D shapes sampled in a variety of ways; note that overall prediction stays stable across various setups.

Sensitivity to Number of Views. We investigate how the performance depends on the number of available views; for this experiment, we take 1024 views following a geodesic spiral around the object, and perform fusion using $n_v = 2, 4, 8, 16, 64, 128, 256$ views. We present qualitative reconstruction results in Figure 21 and demonstrate performance dynamics in Figure 22. We observe a clear benefit from increasing the number of views, and achieve Recall of approximately 90% with 16 views. The dynamics of RMSE and Recall/FPR measures indicate different statistical effects for *min* vs. *truncated min* inference function in (4). More specifically, while *min* provides superior Recall, it stagnates on RMSE as more data are added, not representing correctly the true distance-to-feature field. In contrast, *truncated min* is able to continue improving both RMSE and FPR measures, but shows saturation of Recall as smallest values are being cutoff from the set D_p in (4).

8 CONCLUSIONS

We presented a new learning-based pipeline for automatic sharp feature detection from sampled 3D data. Our approach is based on training and comparing different methods on a dataset annotated with distance-to-feature information derived from the ABC dataset of 3D CAD models. Our method works on patches sampled from the input shape, with predictions combined in a postprocessing step.

We demonstrate that the CNN-based model operating on regularly sampled range images, when such images are available as an input or via resampling the input, is an efficient predictor for distance-to-feature fields. The image-based CNN model is also the most robust to input noise in our experiments. A somewhat surprising observation is that training a regression model benefits from using a histogram loss. At the same time, providing additional inputs, or including additional outputs in training, did not lead to significant improvements in accuracy either for image- or for point-based networks, except adding VCM as input to DGCNN.

We compared our results to recent learning-based methods and a representative high-quality traditional method, demonstrating quantitative and qualitative improvements over these approaches. For instance, the proposed DEF outperforms the best-performing approach by 4% in terms of Recall measure while offering an order of magnitude improvement in false positives rate (from 0.3% to 0.03%). Our method generalizes to real data after fine-tuning; we are not aware of any other feature estimation approach tested on a large collection of real data with manually annotated ground truth. Our approach also scales to orders of magnitude larger point clouds, which has not been successfully shown before.

We make publicly available the two collections of datasets, the benchmarks, the implementation of all baselines, the reference implementation of our method, and our trained models to foster additional work in this direction.

9 LIMITATIONS AND FUTURE WORK

Limitations of our approach to feature estimation include

(1) *Feature Definition.* Our definition of sharp geometric features depends on a relatively large 18° normals angle threshold (normals inner product $\approx$ 0.95). However, for arbitrarily-oriented normals (*e.g.*, the original ABC data [Koch et al. 2019]), we use the absolute of the inner product, and our annotations do not reflect very sharp edges (*i.e.*, those having

Table 7. We demonstrate quantitative results of reconstructing distance-to-feature fields on complete 3D models using variations of our approach. For both DEF-Sim and DEF-Scan collections, we find a significantly better Recall being achieved by *min* fusion, while RMSE favors *truncated min*.

Dataset	Method	RMSE, ↓ $\times 10^{-3}$	RMSE-q_{95}, ↓ $\times 10^{-3}$	Recall (T), %↑		FPR (T), %↓	
				$T = 1r$	$T = 4r$	$T = 1r$	$T = 4r$
DEF-Sim (crops)	DGCNN + Histogram loss ($n_v = 18$, min)	247.6	287.9	52.4	92.3	0.2	2
DEF-Sim	DEF ($n_v = 18$, linear fit)	255.1	351.6	0	3.1	0	0
DEF-Sim	DEF ($n_v = 18$, truncated min)	120.8	227.4	12.5	74.9	0	0.7
DEF-Sim	DEF ($n_v = 18$, min)	100.2	214.1	47.9	92.3	0.2	2
DEF-Sim	DEF ($n_v = 128$, truncated min)	**62.4**	**157.1**	31.8	90.9	0	1
DEF-Sim	DEF ($n_v = 128$, min)	115.1	200.1	**79**	**98**	0.5	5.3
Dataset	Method	RMSE, mm ↓	RMSE-q_{95}, mm ↓	$T = 0.5$ mm	$T = 2$ mm	$T = 0.5$ mm	$T = 2$ mm
DEF-Scan	DEF ($n_v = 12$, linear fit)	1.27	2.36	—	70.1	—	**7.9**
DEF-Scan	DEF ($n_v = 12$, truncated min)	**1.25**	**2.3**	—	80.9	—	9.5
DEF-Scan	DEF ($n_v = 12$, min)	1.54	2.85	—	**91.7**	—	20.1

normals whose inner product is larger than 0.95); this special case remains an open issue.

(2) *Data Annotation Procedure.* For complex geometry (*e.g.*, folded shapes, shapes with rich geometric detail in internal cavities), our distance-to-feature annotations may produce spurious signal on flat surfaces due to feature curves that are close in Euclidean (but not geodesic) sense; we exclude such data from training. In such instances, using *geodesic* instead of *local Euclidean* distances is more appropriate.

(3) *Visibility and Cross-View Consistency.* Dependence on feature visibility can be viewed as a limitation of our approach; however, for common real data acquired by scanners, only visible features are present. We eliminate inconsistency in per-view predictions in each 3D surface point by obtaining multiple likely distance-to-feature values, then statistically inferring a final value (*e.g.*, by taking min).

(4) *Feature Ambiguity.* Sufficiently dense sampling of nearby features is a crucial requirement for our algorithm to accurately distinguish individual features. In instances where having enough (*e.g.*, 8 or more) samples between feature curves is possible, our method efficiently relates samples to respective closest feature lines; otherwise, close feature curves may cause incorrect clustering of points.

(5) *Parametric Curve Extraction.* Limitations of our vectorization method mainly stem from the quality of the extracted distance-to-feature field. For instances with varying sampling density or unstable distance values, our method may struggle with distinguishing close curves or concentric circles (see, *e.g.*, Figure 17, row 4). A partly related effect is gluing together two close corners (see, *e.g.*, Figure 16, row 4).

Future Work in the direction of our research may include

(1) *Extending to Features of Multiple Types.* We have used interior curves in all training examples *on patches,* however we hypothesize that training with boundary (contour) curves *on whole shapes* or *patches with boundary, i.e.*, distinguishing different feature types, might be beneficial.

(2) *Reconstruction of a Complete Distance Field.* Our procedure for inferring distance-to-feature fields on complete 3D shapes is agnostic to the type of function that it reconstructs; at the same time, our distance-to-feature is a non-negative, piecewise-linear, bounded function; incorporating such forms of explicit prior knowledge about this function can considerably improve prediction accuracy.

(3) *Real-World Prediction.* We believe that extending our preliminary study of feature estimation in scanned 3D shapes to a full, robust algorithm capable of vectorizing real-world scans represents a promising research direction.

ACKNOWLEDGMENTS

We are grateful to Prof. Dzmitry Tsetserukou (Skoltech) and his laboratory staff for providing the 3D printing device and technical support. We thank Sebastian Koch (Technical University of Berlin), Timofey Glukhikh (Skoltech) and Teseo Schneider (New York University) for providing assistance in data generation. We also thank Maria Taktasheva (Skoltech) for assistance in computational experiments. We acknowledge the use of computational resources of the Skoltech CDISE supercomputer Zhores for obtaining the results presented in this paper [Zacharov et al. 2019]. The work was supported by the Analytical center under the RF Government (subsidy agreement 000000D730321P5Q0002, Grant No. 70-2021-00145 02.11.2021).

DEF: Deep Estimation of Sharp Geometric Features in 3D Shapes Supplementary Material

A DETAILS ON TRAINING AND EVALUATION DATASETS

A.1 Details on Datasets Construction

Choosing Projection Planes. As outlined in the main text, our image-based datasets consist of range-image data obtained using a set of orthogonal projections. Each projection corresponds to a choice of a plane and placement of the image 64×64 grid (a virtual camera sensor) in the plane. The plane orientation is computed by composing three coordinate frame transformations, that help achieve larger degree of diversity in out datasets:

1) We pick a point on a sphere around the object and start with the tangent plane to the sphere;
2) We translate the image in the picked plane, to capture different parts of the object from this view direction, by offsetting camera frame origin by $(s_x \frac{i_x}{n_x}, s_y \frac{i_y}{n_y})$, where a (s_x, s_y) is the object's bounding-box extent, as seen from picked view direction, n_x, n_y are number of translations performed along camera x- and y-axes, respectively, and $i_{x,y} = -n_{x,y}/2, \ldots, n_{x,y}/2$;
3) We rotate the sample grid orientation in the plane by choosing an uniformly distributed angle of rotation around the z-axis of the camera.

Forming Mesh Patches. We form mesh patches and select feature curves for each patch by extracting entire surface spline regions that are found by association to any of the sampled points, along with their adjacent curves, removing boundary curves (see Section 4.1 in the main text). This helps to ensure that the mesh patch does not have holes consisting of separate triangles not being encountered by raycasting.

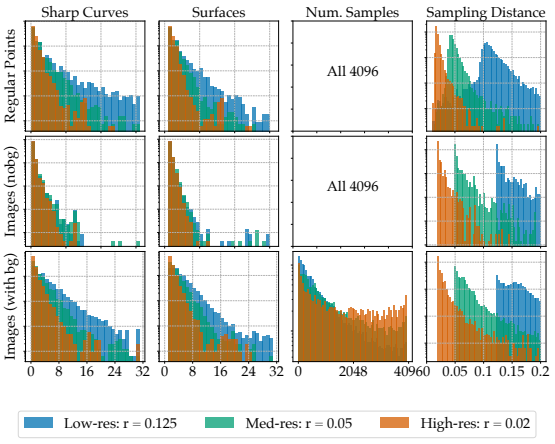

Fig. 23. Statistically, our local patch-based datasets differ substantially with respect to both sampling distance r, sampling pattern (regular and irregular), and data flags. We opt for range-images (with bg, lower row), as it statistically is able to include a wider variety of sampled geometry.

Computing Annotations. We compute distance-to-feature annotations between points and sharp edges in extracted mesh patches using a fast implementation of KD-tree over axis-aligned bounding-boxes enclosing sharp edges, enabling us to compute annotations for millions of point samples quickly.

Data Flagging. The extremely high variability of geometry in our datasets suggests additional data labeling using a number of *data flags*, providing indicators of specific traits encountered in the data. We used the following Boolean data flags:

- *Coarse surfaces (by the number of edges):* spline patches for which triangulated versions have less than 8 edges along any side.
- *Coarse surfaces (by mesh angles):* spline patches for which triangulated versions have a median difference in angles of adjacent faces exceeding 10 degrees.
- *Deviating resolution:* point patches where the average distance between samples deviates by more than $r/2$ from the specified sampling distance.
- *Sharpness discontinuities:* point patches for which difference in distance annotation in any two neighboring points exceeds the Euclidean distance between the two.
- *Bad face sampling:* point patches for which the average number of point sampled on each face is not in the range $[r, 100r]$.
- *Raycasting background:* set to *true* for images where at least one pixel contains background values.
- *Depth discontinuities:* set to *true* for images where depth changes by more than $T = 0.5$ units in neighboring pixels.

Our final datasets (DEF-Sim) are formed so that all flags are required to be false, except for *Depth discontinuities*, and *Raycasting background*, that we allow to take arbitrary values when forming *with bg* versions of our data.

A.2 Summary and Statistics of Our Datasets

We have computed a number of statistical quantities to better understand and characterize our data collections. Table 8 presents an overview of core statistics for datasets used in this work, and Figures 23–24 represent patch and complete model statistics for DEF-Sim and DEF-Scan, respectively. We confirm that we have developed

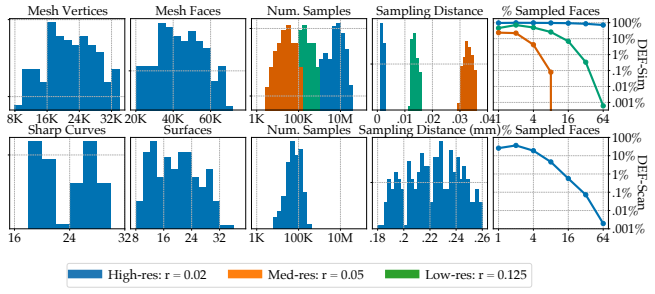

Fig. 24. Statistically, our simulated (top row) and scanned (bottom row) complete 3D shape datasets vary with respect to sampling distance r, and DEF-Scan is similar to a medium-resolution version of DEF-Sim in terms of sampling density per feature.

Table 8. Overview of all data collections used within this work. For complete models and DEF-Scan, we provide estimates of the percentage of sampled faces, the sampling distance, and the number of samples. We use the following shorthands for Patch Selection and Annotation: SR: patch selection and annotation based on local surface regions; FM: patch selection and annotation based on full 3D mesh model. We use the following shorthands for Sampling: RC: range-images sampling obtained using raycasting; RC*: for full models, we concatenate range scans sampled using raycasting; S: sampling pattern emerging for real-world scanning. We use the following shorthands for Noise: $\Sigma_3 = \{0.005, 0.02, 0.08\}$: the set of three noise magnitudes used for complete 3D shapes. $\Sigma_6 = \{0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$: the set of six noise magnitudes used for complete 3D shapes. * designates an estimate computed over the concatenated scans; z: adding noise in the direction of z-axis of the virtual camera; z^*: for full models, we concatenate noisy range scans sampled using raycasting; S: noise pattern emerging for real-world scanning.

Dataset	Num. Samples	% Sampled Faces	Sampling Dist. r	Noise std σ	Train	Val	Test	Patch Selection	Sampling	Annotation	Noise	bg
DEF-Sim (patch-high-0-nobg-N)	4096	—	0.02	—	2K-256K	32K	32K	SR	RC	SR	—	✗
DEF-Sim (patch-med-0-nobg-64k)	4096	—	0.05	—	64K	32K	32K	SR	RC	SR	—	✗
DEF-Sim (patch-low-0-nobg-64k)	4096	—	0.125	—	64K	32K	32K	SR	RC	SR	—	✗
DEF-Sim (patch-high-σ-nobg-64K)	4096	—	0.02	Σ_6	64K	32K	32K	SR	RC	SR	z	✗
DEF-Sim (patch-high-0-wbg-N)	2913	—	0.02	—	2K-256K	32K	32K	SR	RC	SR	—	✓
DEF-Sim (patch-med-0-wbg-64k)	1880	—	0.05	—	64K	32K	32K	SR	RC	SR	—	✓
DEF-Sim (patch-low-0-wbg-64k)	1201	—	0.125	—	64K	32K	32K	SR	RC	SR	—	✓
DEF-Sim (patch-high-σ-wbg-64K)	2913	—	0.02	Σ_3	64K	32K	32K	SR	RC	SR	z	✓
DEF-Sim (complete-high-0-68)*	8456K	98%	0.002	—	—	—	68	FM	RC*	SR	—	✓
DEF-Sim (complete-med-0-68)*	225K	71%	0.013	—	—	—	68	FM	RC*	SR	—	✓
DEF-Sim (complete-low-0-68)*	36k	22%	0.033	—	—	—	68	FM	RC*	SR	—	✓
DEF-Sim (complete-high-σ-68)*	8456K	98%	0.002	Σ_3	—	—	68	FM	RC*	SR	z^*	✓
DEF-Scan (patches-med)*	6878	—	0.5 mm	—	981	479	468	FM	S	FM	S	✓
DEF-Scan (complete-med-scan)*	83K	36%	0.22 mm	0.328 mm	86	41	39	FM	S	FM	S	✓

a variety of diverse synthetic and real-world datasets suitable for training and testing methods of detection sharp geometric feature curves.

B DETAILS ON RECONSTRUCTION FOR COMPLETE 3D MODELS

Inference Functions. We infer the final distance-to-feature estimate by computing the value of a inference set-function $g(\cdot)$ given a set $D_p = \{\widehat{d}_1(p), \ldots, \widehat{d}_n(p)\}$ of predictions obtained (either directly or by interpolation) for each sampled point p. To process these predictions, we have experimented with the following variants of pointwise aggregation. Basic aggregation methods:

- averaging $g(D_p) = \frac{1}{|D_p|} \sum_{\widehat{d} \in D_p} \widehat{d}$, computing median $g \equiv$ median, and extracting minimum: $g \equiv$ min.
- computing truncated average and minimum, computed by removing the largest and smallest 20% of values, then computing the corresponding quantity;
- to perform inference based on predictions obtained using segmentation methods (e.g., [Mérigot et al. 2010; Raina et al. 2019; Yu et al. 2018]), one can use the following simple scheme. Individual predictions $\widehat{d}_1(p), \ldots, \widehat{d}_n(p)$, with $\widehat{d}_i(p) \in \{0, 1\}$, can be combined using $g(D_p) = \mathbb{I}_{[T,1]}(\frac{1}{|D_p|} \sum_{\widehat{d} \in D_p} \widehat{d})$, i.e. setting the fused prediction to 1 (sharp) when an average predicted value exceeds a threshold T.

Predictions obtained using one of the basic methods can be post-processed to improve smoothness by:

- minimizing L_2 or total-variation (TV) based functionals of the form:

$$\min_{\{\widehat{d}(p)\}} ||\widehat{d}(p) - \widehat{d}^0(p)|| + \alpha \sum_{k=1}^{K} ||\widehat{d}(p) - \widehat{d}(\mathrm{NN}_k(p)||^\gamma,$$

  ($\mathrm{NN}_k(p)$ denotes the kth nearest neighbor of the point p, we used $K = 50$ and $\gamma \in \{1, 2\}$);
- fitting a robust version of local linear regression [Huber et al. 1973] (we extract local point patches of $K = 50$ neighbors of each point, reduce their feature dimensionality to 2, fit a outlier-robust linear regression model [Owen 2006] using the `scikit-learn` implementation (`HuberRegressor`), and extract predictions in the seed point).

Overall, we have found that setting $g \equiv$ min produces the best results for our test samples set.

Details on Transferring Predictions across Image Views. For a 3D point p, we perform interpolation and estimation of visibility $\widehat{v}^{s \to t}(p)$ as indicated below. To interpolate predicted distance values at the

warped point $\widehat{p}$ in the reprojected image I_s, we construct a K-neighborhood $\{\mathrm{NN}_k(\widehat{p})\}_{k=1}^{K}$ (we set $K = 4$) and compute the linear bivariate B-spline representation of a surface [Dierckx 1995] using this neighborhood and respective distance values in $\widehat{d}_s(\widehat{p})$. We have chosen an implementation available in SciPy [Virtanen et al. 2020] and invoke the low-level `scipy.interpolate.bisplrep` over the wrapper `scipy.interpolate.interp2d` as the former offers direct control over the smoothness of the result. We evaluate the fitted B-spline at point $\widehat{p}$ to obtain an interpolated distance value (equivalently to a bilinear interpolation) and set the binary visibility mask $\widehat{v}^{s\to t}(p)$ to 1, or mark an interpolated value as not available when less than K nearest neighbors exist within a Euclidean distance of $6r$, where r is the sampling distance. We do not perform interpolation for points on the patch boundary as we have discovered the corresponding estimates to be unstable. In these instances, we set the binary visibility mask $\widehat{v}^{s\to t}(p)$ to zero. We repeat the described process for all available pairs of images.

C DETAILS ON EXPERIMENTAL EVALUATION

C.1 Experimental Setup

Measures of Quality. For each patch P_i, our computed quality measures are defined by:

$$\mathrm{RMSE}_i = \frac{1}{\sqrt{N_i}}\sqrt{\sum_{p\in P_i}\left(d_i(p) - \widehat{d}_i(p)\right)^2},$$

$$\mathrm{Recall}_i(T) = \frac{\sum\limits_{p\in P_i}\widehat{s}_i(p)s_i(p)}{\sum\limits_{p\in P_i}s_i(p)},$$

$$\mathrm{FPR}_i(T) = \frac{\sum\limits_{p\in P_i}\widehat{s}_i(p)(1-s_i(p))}{\sum\limits_{p\in P_i}(1-s_i(p))},$$

where $d_i(p)$ and $s_i(p)$ are the ground-truth distances and thresholded labels, respectively, $\widehat{d}_i(p), \widehat{s}_i(p)$ their respective estimates, and $N_i = |P_i|$ the number of non-background samples in the patch P_i. For methods producing hard segmentation labels, we directly use their predictions; for methods producing segmentation probability labels, we compute $\widehat{s}_i = \mathbb{1}\left(\widehat{r}_i > 0.5\right)$ where $\widehat{r}_i(p)$ is the estimated probability for p to be a sharp point. We provide RMSE-q_{95} for a collection of patches $\{P_i\}$ by computing the 95% quantile of respective RMSE_i values. We calculate the metrics for a set of patches by averaging metrics obtained for individual patches.

To measure the curve extraction quality, we used metrics defined by:

$$\mathrm{CD}_{P\to Q} = \frac{1}{|P|}\sum_{p\in P}\inf_{q\in Q}\|p-q\|^2,$$

$$\mathrm{CD} = \mathrm{CD}_{P\to Q} + \mathrm{CD}_{Q\to P},$$

$$\mathrm{HD} = \max\{\sup_{p\in P}\inf_{q\in Q}\|p-q\|,\ \sup_{q\in Q}\inf_{p\in P}\|p-q\|\},$$

where P and Q are point clouds that are compared.

Here Chamfer distance CD reflects the average discrepancy in two sets of curves, and Hausdorff distance HD measures the worst-case deviation between the curves.

Our third metric, Sinkhorn distance SD, is an approximation of the Wasserstein optimal transportation. It uses blurring the transport plan through the addition of an entropic penalty to reduce the computational cost. SD is computed as a series of iterative updates, for more details refer to [Feydy et al. 2019].

C.2 Parameter Choices

Voronoi Covariance Measure (VCM) [Mérigot et al. 2010] We ran a direct grid search to obtain the set of parameters with the maximal Recall for each sampling distance and noise level. Each of the parameters was varied over a grid of 11 values: $\{0.01, 0.05\} \bigcup \{0.1i\}_{i=1}^{9}$. For each combination we ran *VCM* inference on the validation set, computed Recall value and determined the set of parameters maximizing the metric. The selected parameters are presented in Table 9.

Sharpness Fields (ShF) [Raina et al. 2019]

ShF outputs a real-valued field similar to ours, which has value 0 far from feature line and reaching 1 at the feature. In practice we observed that this field is more narrow than ours, meaning that for a fair comparison we needed to find a linear transformation to equalize them. To do that, we implemented the following transformation selection procedure:

$$\min_{\alpha}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left([1-ShF_i] - \max\{d_i/\alpha, 1\}\right)^2},$$

where d_i is our ground truth distance-to-feature field of i-th patch, ShF_i is a prediction by their network on i-th patch, α is an equalizing coefficient, N is the size of validation set. Intuitively, this functional measures RMSE between the predictions by *ShF* and our transformed field. We computed these values for a range of coefficients $\alpha = \{0.01i\}_{i=1}^{10}$ on validation set and selected $\alpha = 0.06$ as the one minimizing this functional.

Other competitors *Edge-Aware Consolidation Network (EC-Net)* [Yu et al. 2018], *PIE-NET* [Wang et al. 2020], *PC2WF* [Liu et al. 2021] have no parameters to tune.

C.3 Parameter Choices for Vectorization Pipeline

The sampling technique in DEF-Sim ensures that the pairwise point distance r_{high} is 0.02 for individual images on average; we choose to relate all parameters to this value. We observed that the parameters with the strongest effect on the final result were the proximal points selection threshold d_{sharp} and the corner detection threshold T_{corner}. To set these parameters, we implemented a parameter sweep over a grid: we varied d_{sharp} in the range $[2r, 4r]$, and T_{corner} in the range $[0.6, 0.85]$. For each set of parameters, we ran the whole

Table 9. Parameters of VCM for different types of data.

Sampling distance r	Noise magnitude σ	R	ρ	T
r_{high}	$0, r/8, r/4, r/2, r$	0.05	0.1	0.3
r_{high}	$2r, 4r$	0.1	0.3	0.3
r_{med}	0	0.1	0.1	0.4
r_{low}	0	0.2	0.1	0.4

vectorization procedure and computed a symmetric CD between the sampled spline curves and a point set P_{sharp} that consists only of points with estimated distance $\widehat{d}$ less than d_{sharp}, thus measuring the goodness of fit. The resulting set of parameters is chosen by the lowest value of CD. We found reasonable default settings for the rest of the tunable parameters that do not affect the result as much.

For the endpoint detection, we choose $R_{\text{endpoint}} = 10r$. The threshold $T_{\text{endpoint}} = 0.6$, which means there should be 60% more points on one side from the query point compared to the other side to consider a ball center to be the curve endpoint. Finally, the choice of splitting threshold is $T_{\text{split}} = 4r$. We want the polyline controlled by this value to accurately reflect the corresponding curve geometry.

Finally, we discuss parameters $\mathcal{N}_i$, T_{variance}, and R_{corner} used in corner detection procedure. It is designed as aggregation of several corner estimates, hence it doesn't require setting the exact parameter values. We vary R_{corner} in the range of $5r, \ldots, 8r$, the number of neighbor sets $\mathcal{N}_i$ in the range 10, 20, 40, and the threshold T_{variance} in the range 5, 10, 15, 20, 25. With this grid of parameters we obtain 60 different corner estimates, for each set of estimates we compute the fraction of cases where a specific set B_i was labeled as a corner and normalize it by 60, eventually obtaining a probability for each set to be a corner.

C.4 More Ablative Experiments

Data volume. As a part of the ablative studies, we conduct training on datasets of increasing size. We performed training for each dataset size (we used noise-free patch datasets with sampling distance $r = r_{\text{high}}$) until convergence. We present results in Table 10, where we observe that metric values stabilize for datasets with around 64k training patches. Not surprisingly, larger training datasets improve performance. The subsequent experiments were performed with 64k training patches.

Model capacity. We performed an additional experiment to identify the optimal configuration of our backbone CNN. We instantiated a series of ResNet [He et al. 2016] backbones with significantly varying number of parameters and trained each until convergence on the validation set. Table 11 presents results, that generally indicate some increase in performance for larger models. We select the ResNet-152 backbone network for all subsequent studies.

Additional Inputs. We evaluated the effect of adding auxiliary inputs by concatenating the *VCM* prior sharpness estimates, normal vectors, and both simultaneously to the raw range images (sampling distance $r = r_{\text{high}}$, no noise). We trained ResNet-152 on depth images with additional input channels; we present statistical results in Table 12, upper rows, where we compare these configurations againt the baseline where an input range-image P is regressed onto a distance labels $d(p)$. Metric values demonstrate that no conclusive gain in performance is observed for regression metrics, compared to such a baseline. Hence, we further train on range-images without additional inputs in all instances.

Additional Outputs. Similarly to the previous experiments, we performed an ablative study to understand how the auxiliary tasks affect feature line estimation performance. We experimented with concatenating direction-to-feature, ground-truth normals, and both simultaneously to the distance labels $d(p)$, and adding additional heads to our network to predict these quantities. We present statistical results of this experiment in Table 12, middle rows. In all cases regressing the normals, directions towards the feature line, or both of them at the same time did not lead to increasing the quality of feature line extraction. Hence, we proceed further without using any additional outputs.

Table 10. For DEF networks trained on datasets of increasing size, performance generally stabilizes for 16K–64K patches (DEF-Sim, no bg, $r = r_{\text{high}}, \sigma = 0$). We opt for 64K patches as this dataset size provide the most diversity for training.

Train Size	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
2k	118.7	545.7	0	0
4k	138.6	609.4	0	0
8k	105.5	581.4	37.65	0.1
16k	57.5	341.8	63.4	0.18
32k	61.4	403.2	70.5	0.22
64k (Ours)	61.5	361.1	57.36	0.06
256k	85	424.9	45.01	0.07

Table 11. As image-based backbone grows in capacity, DEF results generally improve on validation set (DEF-Sim, no bg, $r = r_{\text{high}}, \sigma = 0$). We end up selecting the largest resnet152 backbone for the remaining experiments.

Backbone (# Params)	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
resnet26 (34.4 M)	9.3	37	72.47	0.02
resnet34 (30 M)	9.8	34.7	**83.81**	0.02
resnet50 (44 M)	7.3	24	82.12	0.02
resnet101 (63 M)	8.2	26.5	79.85	0.02
resnet152 (78.6 M)	**7.2**	**23.1**	83.39	0.02

C.5 More Experiments on Complete 3D Models

We have performed more experiments to investigate the limits of robustness of our method to reduction in sampling density and increase in noise strength. To this end, we employed $n_v = 18$ views of the same models in DEF-Sim dataset, but have augmented respective range-images with noise acting in the camera direction, and performed sampling to model decrease in point density as r grows. Table 13 presents quantitative evaluation of our method with such input data. We conclude that sparse data, at least the ones we studied, did not result in a significant degradation of our approach, apart from the large increase in the FPR measure, indicating that more false positives shall be identified. Adding noise, in contrast, significantly impacts results, as the method tends to no longer detect features, instead focusing on averaging predictions across the shape in an attempt to reduce noise. Even so, our method remains generally stable for noise magnitudes of up to r, all with using only

Table 12. We perform experiments to study the effect of introducing additional signals at the input, and additional supervision at the output of our networks (results obtained on DEF-Sim, no bg, $r = r_{\text{high}}$, $\sigma = 0$). As input in addition to depth image P, we supply ground-truth normals $n(p)$, prior sharpness estimates $\widehat{s}_{\text{VCM}}(p)$ obtained by *VCM*, and their combinations. As output in addition to distance estimates $\widehat{d}(p)$, we require our model to predict normals $\widehat{n}(p)$, 3×1 directions $\widehat{r}(p)$ to the closest point on the sharp feature curve, and their combinations. We end up selecting the most basic scheme where we predict distance estimates $\widehat{d}(p)$ from the input depth image P.

Input	Output	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
$P, n(p)$	$\widehat{d}(p)$	7.2	34.1	69.31	0.02
$P, \widehat{s}_{\text{VCM}}(p)$	$\widehat{d}(p)$	8.6	26.8	78.09	0.03
$P, n(p), \widehat{s}_{\text{VCM}}(p)$	$\widehat{d}(p)$	6.2	25.9	76.53	0.02
P	$\widehat{d}(p), \widehat{n}(p)$	8.1	31.8	74.69	0.01
P	$\widehat{d}(p), \widehat{n}(p), \widehat{r}(p)$	8.5	33.2	74.82	0.02
P	$\widehat{d}(p), \widehat{r}(p)$	8.3	33.9	74.09	0.02
P	$\widehat{d}(p)$	**7.2**	**23.1**	**83.39**	**0.02**

18 views for reconstruction, that we have identified is a modest number of views.

We additionally investigated how our method's performance depends on the location of the predictions, relative to a sharp feature curve. As can be seen in Figure 25, our method has a performance peak at around r to $2r$, which indicates that predicting distances in locations exactly on the feature curve or far away from the curve might be more difficult than doing so in some proximity from the curve.

D ALTERNATIVE POINT-BASED PIPELINE

D.1 Dataset Construction

We describe an alternative procedure to obtain point-sampled patches P with $N = |P| = 4096$ points with distance-to-feature annotations $d(p), p \in P$.

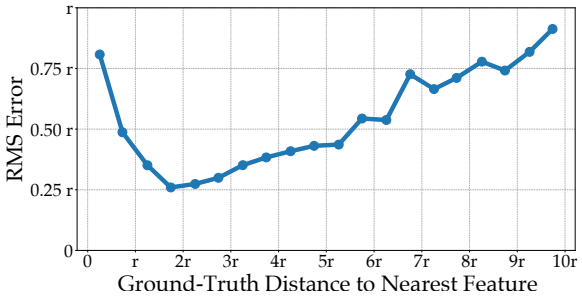

Fig. 25. Statistically, our method has the lowest RMSE in locations spaced around r–$2r$ from the sharp feature curves. This observation explains why in particular instances our method demonstrates performance drops in Recall ($1r$) while remaining robust according to Recall ($4r$).

Dataset Design. We follow exactly the same procedure for feature definition, feature selection, distance-to-feature computation, deciding on feature size, and computing sampling density, as described in the main text.

Patch and Feature Selection. We extract local patches from triangulated 3D shapes by selecting all mesh faces inside or intersecting with a sphere of radius $\sqrt{N}r/2$ ($N = 64^2$), centered at 128 uniformly distributed (using Poisson Disk Sampling [Bowers et al. 2010]) points on the model surface. Among all connected parts of the mesh inside the sphere, if any, the largest one is selected.

Shape Sampling. We obtain point clouds using Poisson Disk Sampling [Bowers et al. 2010], similar to [Wang et al. 2020] and unlike [Yu et al. 2018] that use raycasting similarly to our image-based datasets. If the number of samples generated on this patch with Poisson disk sampling is larger than N, N points closest to the center are retained; if the number of sampled points is less than N, this particular patch is discarded.

Patch-Based Datasets. We have constructed a dataset of 65,536 patches for training, 32,000 patches for validation, and 32,000 for testing our model.

Complete 3D Model Datasets. To construct a sampled and annotated version for a complete 3D model, we first compute a Poisson Disk Sampling of the complete 3D mesh. Next, to compute distance-to-feature annotations over the complete 3D shape, we extract overlapping local regions in the mesh as mentioned above, associate the sampled points to each local mesh region, and annotate these points using our normal procedure; this results in multiple annotations available for each point as local regions overlap. We compute a minimum over the available annotations in each point to produce the final complete annotations.

D.2 Methods

Learning Architecture. We use the DGCNN architecture [Wang et al. 2019] and systematically vary the size of the base network, by simultaneously increasing both width W and depth D according to the relations $W = 64 \times 1.35^k$, $D = 3 + k$, varying k from -2 to 3. The quantitative results suggest that for $k \geqslant 1$ the gains in performance stabilize; we end up choosing $k = 3$, DGCNN with depth $D = 6$, width $W = 158$. While the DGCNN model was trained using the Histogram loss using the supervision from ground-truth distances $d(p)$ only, we discovered that adding prior sharpness estimates from *VCM* has the potential improve performance considerably; this is in contrast with the effect *VCM* has on image-based data. However, adding *VCM* labels requires an additional effort to compute these scores before running the model on the new shapes.

Reconstruction for Complete 3D Models. To compute a distance-to-feature field for an input complete 3D shape P, we first extract point patches P_i with 4096 points. We use an adjacency graph of the points based on their k nearest neighbors (we use $k = 5$), extracting the largest connected component of this graph. Each patch is obtained by a breadth-first search from a vertex, and we add patches until each point is covered by at least 10 patches. For each of these local patches

Table 13. Results of reconstructing object-level distance-to-feature field (DEF-Sim, 68 shapes) indicate that DEF is able to perform robustly w.r.t. sampling distance, with only FPR indicating performance degradation for lower resolution datasets. DEF is additionally resilient to noise with signal-to-noise ratios of up to 1:1, as indicated by Recall($4r$); for larger noise magnitudes, performance inevitably degrades.

Sampling distance r	Noise magnitude σ	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓	Recall ($4r$), %↑	FPR ($4r$), %↓
r_{high}	0	100.2	214.1	47.9	0.2	92.3	2
r_{med}	0	88.4	197	38.4	0.2	94.1	4.5
r_{low}	0	134.7	272.1	47.3	2.1	95.5	22.7
r_{high}	$r/4$	658.5	817	56.7	2.6	96.1	16.7
r_{high}	r	651.4	786.2	6	0.3	71.8	10.5
r_{high}	$4r$	541.5	730.8	0	0	37.9	4.9

Table 14. Experiments using point-based DGCNN [Wang et al. 2019] demonstrate promising results for unstructured sampling patterns with uniform sampling; however, image-sampled patches tend to be significantly non-uniform, impairing DGCNN performance; adding prior sharpness estimates from *VCM* yields no advantage for this method.

Dataset	Method	RMSE ↓ $\times 10^{-3}$	RMSE-q_{95} ↓ $\times 10^{-3}$	Recall ($1r$), %↑	FPR ($1r$), %↓
Unstructured points	DGCNN + Histogram loss	10.0	38.1	89.5	7×10^{-2}
Unstructured points	DGCNN + Histogram loss + VCM	7.8	25.6	90.0	8×10^{-2}
Regular images (no bg, reprojected to points)	DGCNN + Histogram loss	11.3	55.5	80.9	3.7×10^{-2}
Regular images (no bg, reprojected to points)	DGCNN + Histogram loss + VCM	13.6	70.0	68.8	4.8×10^{-2}

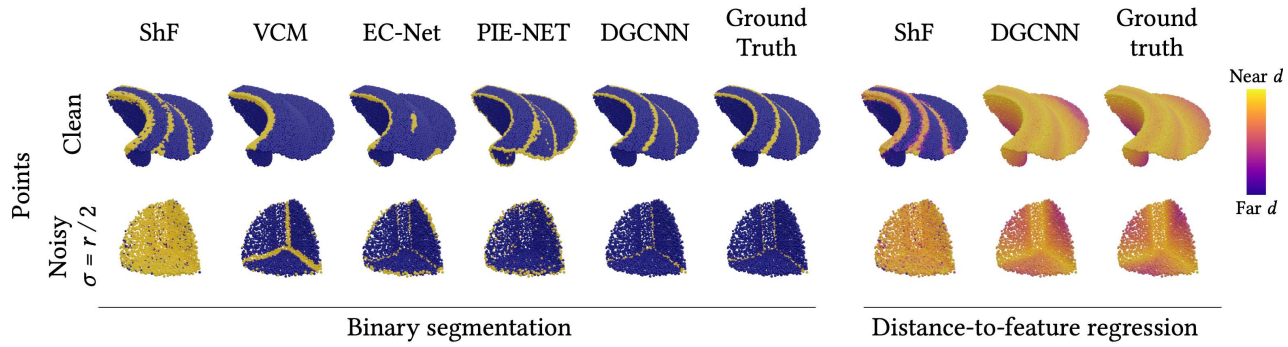

Fig. 26. Comparative prediction results for a DGCNN model pre-trained on a point-based collection vs. the competitor approaches *ShF*, *VCM*, *EC-Net*, and *PIE-NET*. The DGCNN model trained on the datasets we use is able to perform competitively on sampled data.

P_i, we predict a distance-to-feature field $\widehat{d}_i(p)$ using a DGCNN model, resulting in a set of predictions

$$D_p = \left\{ d_p \mid d_p = \widehat{d}_i(p) = \text{DGCNN}(p|P_i), p \in P_i \right\}_{p=1}^{|P|} \qquad (12)$$

for each point p in the input point cloud (here DGCNN$(p|P_i)$ denotes DGCNN prediction in the same point p given the context point patch P_i). The set D_p of predictions for all patches containing p is filtered by excluding predictions from 20% of points in the patch furthest away from its center of mass. Finally, we compute a minimum over all predictions of the distance $\widehat{d}(p) = \min_{d \in D_p} d$. Other possibilities described in Section B of this document can also be applied.

D.3 Experimental Results

Training Details. All training patches consist of 4,096 points; we applied a random 3D rotation to each patch as an augmentation.

Patch-Based Results. Table 14 contains statistical results of the influence of sampling pattern and prior sharpness estimates from *VCM* on performance. We note that the two datasets are not directly comparable, even though they represent point-sampled geometry with the same feature size distribution (sampling distance $r = r_{\text{high}}$). Specifically, while the point-sampled geometry does contain similar geometric patterns, the sampling pattern is more regular, which is ensured by the Poisson Disk Sampling; in contrast, range-scans produced by ray-casting have significantly non-uniform sampling,

where density may vary significantly for surfaces on either side of a sharp feature curve. We conclude that training a model from point-based data benefits from adding prior hard sharpness estimates from the *VCM* method, likely due to benefits offered by sampling; this is not the case for image-sampled data.

We demonstrate a qualitative comparison of patch-based feature estimation performance in the same fashion as for image-based datasets in the main text. For this experiment, competitor approaches were optimized according to the same procedure as for the image-based datasets (see Section C.1 of this document). Figure 26 displays a comparison of point patches where competitor approaches are compared against a DGCNN model trained on a patch-based dataset (see Section D.1 above). We conclude that a point-based network pre-trained on the kind of datasets we use can generalize well to unseen instances and present a viable alternative to competitor approaches.

Results on Complete 3D Models. We make an effort to compare the DGCNN-based method for reconstructing distance-to-feature fields for complete 3D models (see Section D.2) on the same data collection of 68 shapes as our method, DEF, was tested on. The results in Table 7 of the main text indicate that DGCNN-based method is capable of producing nearly the same Recall and FPR values, however it is outperformed by a large margin (2×) according to RMSE measure. As we require our distance field to be as accurate as possible (e.g., for the reconstruction of the set of parametric representations of sharp feature curves), we made an eventual choice in favor of the image-based method.

REFERENCES

D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo. 2015. Fast and Robust Edge Extraction in Unorganized Point Clouds. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 1–8. https://doi.org/10.1109/DICTA.2015.7371262

D. Bazazian and ME. Parés. 2021. EDC-Net: Edge Detection Capsule Network for 3D Point Clouds. *Applied Sciences* 11, 4: 1833 (2021), 1–16. https://doi.org/10.3390/app11041833

John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. 2010. Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–10.

Yuanhao Cao, Liangliang Nan, and Peter Wonka. 2016. Curve networks for surface reconstruction. *arXiv preprint arXiv:1603.08753* (2016).

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. Meshlab: an open-source mesh processing tool.. In *Eurographics Italian chapter conference*, Vol. 2008. Salerno, Italy, 129–136.

Joel II Daniels, Linh K Ha, Tilo Ochotta, and Claudio T Silva. 2007. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*. IEEE, 123–136.

Joel Daniels Ii, Tilo Ochotta, Linh K Ha, and Cláudio T Silva. 2008. Spline-based feature curves from point-sampled geometry. *The Visual Computer* 24, 6 (2008), 449–462.

Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design* 39, 4 (2007), 276–283.

Paul Dierckx. 1995. *Curve and surface fitting with splines*. Oxford University Press.

WA Falcon. 2019. PyTorch Lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. 2019. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2681–2690.

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. 2005. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)* 24, 3 (2005), 544–552.

Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. 2016. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4340–4349.

T. Hackel, J. D. Wegner, and K. Schindler. 2016. Contour Detection in Unstructured 3D Point Clouds. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1610–1618. https://doi.org/10.1109/CVPR.2016.178

Timo Hackel, Jan D. Wegner, and Konrad Schindler. 2017. Joint classification and contour extraction of large 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 130 (2017), 231 – 245. https://doi.org/10.1016/j.isprsjprs.2017.05.012

Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. 2016. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4077–4085.

JH Hannay and JF Nye. 2004. Fibonacci numerical integration on a sphere. *Journal of Physics A: Mathematical and General* 37, 48 (2004), 11591.

Richard Hartley and Andrew Zisserman. 2004. *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511811685

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research* 21, 248 (2020), 1–43.

Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. 2021. PCEDNet: A Lightweight Neural Network for Fast and Interactive Edge Detection in 3D Point Clouds. *ACM Transactions on Graphics (TOG)* 41, 1 (2021), 1–21.

Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Richard Zhang. 2013. Edge-aware point set resampling. *ACM transactions on graphics (TOG)* 32, 1 (2013), 9.

Peter J Huber et al. 1973. Robust regression: asymptotics, conjectures and Monte Carlo. *The annals of statistics* 1, 5 (1973), 799–821.

Ehsan Imani and Martha White. 2018. Improving Regression Performance with Distributional Losses *(Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 2157–2166. http://proceedings.mlr.press/v80/imani18a.html

Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. 2019. Learning Unsupervised Multi-View Stereopsis via Robust Photometric Consistency. arXiv:1905.02706 [cs.CV]

Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. 2020. A Large-scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks. In *Proceedings of 16th European Conference on Computer Vision (ECCV)*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9601–9611.

Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J Mitra. 2021. CPFN: Cascaded Primitive Fitting Networks for High-Resolution Point Clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7457–7466.

Kai Wah Lee and Pengbo Bo. 2016. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering* 3, 2 (2016), 102 – 111. https://doi.org/10.1016/j.jcde.2015.07.001

Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. 2019. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2652–2660.

Y. Lin, C. Wang, B. Chen, D. Zai, and J. Li. 2017. Facet Segmentation-Based Line Segment Extraction for Large-Scale Point Clouds. *IEEE Transactions on Geoscience and Remote Sensing* 55, 9 (2017), 4839–4854. https://doi.org/10.1109/TGRS.2016.2639025

Yangbin Lin, Cheng Wang, Jun Cheng, Bili Chen, Fukai Jia, Zhonggui Chen, and Jonathan Li. 2015. Line segment extraction for large scale unorganized point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 102 (2015), 172 – 183. https://doi.org/10.1016/j.isprsjprs.2014.12.027

Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. 2021. PC2WF: 3D Wireframe Reconstruction from Raw Point Clouds. *CoRR* abs/2103.02766 (2021). arXiv:2103.02766 https://arxiv.org/abs/2103.02766

Albert Matveev, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. 2021. 3D Parametric Wireframe Extraction Based on Distance Fields. In *2021 4th International Conference on Artificial Intelligence and Pattern Recognition* (Xiamen, China) *(AIPR 2021)*. Association for Computing Machinery, New York, NY, USA, 316–322. https://doi.org/10.1145/3488933.3488982

Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas. 2010. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics* 17, 6 (2010), 743–756.

Open CASCADE Technology OCCT 2021. Open CASCADE Technology OCCT. https://www.opencascade.com/. Accessed: 2021-06-01.

AB Owen. 2006. A robust hybrid of lasso and ridge regression Technical Report.

Parasolid: 3D Geometric Modeling Engine 2021. Parasolid: 3D Geometric Modeling Engine. https://www.plm.automation.siemens.com/global/en/products/plm-components/parasolid.html. Accessed: 2021-06-01.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

Prashant Raina, Sudhir Mudur, and Tiberiu Popa. 2019. Sharpness fields in point clouds using deep learning. *Computers & Graphics* 78 (2019), 37–53.

RangeVision Spectrum 2021. RangeVision Spectrum - a new 3D high-resolution scanner. https://rangevision.com/en/products/spectrum/. Accessed: 2021-06-01.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. 2020. Parsenet: A parametric surface fitting network for 3d point clouds. In *European Conference on Computer Vision*. Springer, 261–276.

Maria-Laura Torrente, Silvia Biasotti, and Bianca Falcidieno. 2018. Recognition of feature curves on 3D shapes using an algebraic approach to Hough transforms. *Pattern Recognition* 73 (2018), 111–130.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* 17, 3 (2020), 261–272.

Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. 2020. PIE-NET: Parametric Inference of Point Cloud Edges. *Advances in Neural Information Processing Systems* 33 (2020).

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–12.

Christopher Weber, Stefanie Hahmann, and Hans Hagen. 2010. Sharp feature detection in point clouds. In *2010 Shape Modeling International Conference*. IEEE, 175–186.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2020. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Reconstruction. *arXiv preprint arXiv:2010.02392* (2020).

Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. 2019. Learning attraction field representation for robust line segment detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1595–1603.

Omry Yadan. 2019. Hydra - A framework for elegantly configuring complex applications. Github. https://github.com/facebookresearch/hydra

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. EC-Net: an Edge-aware Point set Consolidation Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 386–402.

Igor Zacharov, Rinat Arslanov, Maksim Gunin, Daniil Stefonishin, Andrey Bykov, Sergey Pavlov, Oleg Panarin, Anton Maliutin, Sergey Rykovanov, and Maxim Fedorov. 2019. "Zhores"-Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *Open Engineering* 9, 1 (2019), 512–520.