# $MLS^2$: Sharpness Field Extraction Using CNN for Surface Reconstruction

Prashant Raina* Sudhir Mudur Tiberiu Popa

Department of Computer Science and Software Engineering
Concordia University

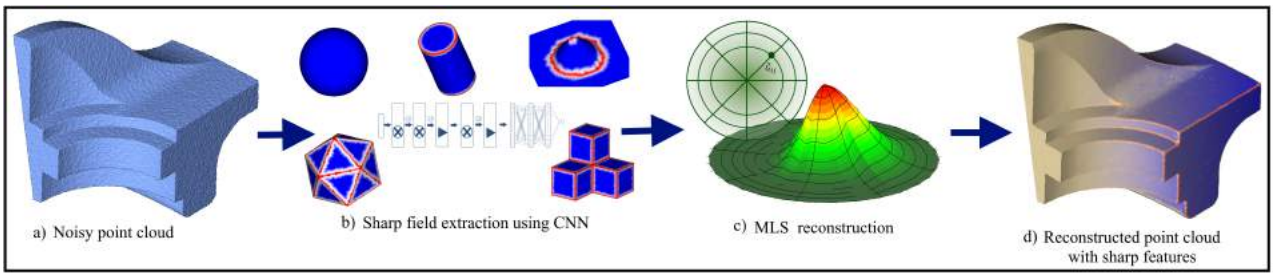


Figure 1: High level view of our method. A sharpness field is extracted from a noisy point cloud (a) using a convolutional neural network (b). An anisotropic kernel (c) is computed that conforms to the sharpness field resulting in a final surface (d) that conforms to the sharpness field.

**ABSTRACT**

We address the challenging problem of reconstructing surfaces with sharp features from unstructured and noisy point clouds. For smooth surfaces, moving least squares (MLS) has been a popular method. MLS variants for dealing with sharp features have been proposed, though they have not been as successful. Our take on this problem is very different. By training a convolutional neural network (CNN), we first derive a sharpness field parametrized over the underlying smooth proxy MLS surface. This field provides us two benefits - (i) it enables us to both detect and reconstruct sharp features, this time using an anisotropic MLS kernel, while preserving most of the MLS reconstruction method's properties, and (ii) unlike classification based methods, it does not require that sharp features be present only at input points. With just a small amount of training data, we demonstrate our results on a set of illustrative test cases and compare qualitatively and quantatively with results from MLS variants and the more recent PointNet deep learning network.

**Index Terms:** Computing methodologies—Computer graphics—Shape modeling—Point-based models; Computing methodologies—Machine learning—Machine learning approaches—Neural networks

## 1 INTRODUCTION

Points are one of the most common acquisition primitives in geometric scanning, making point clouds one of the most popular representations for scanned 3D models [1, 7, 43]. However, such point clouds are not structured (they have no connectivity information like meshes), and typically contain noise and outliers specific to the acquisition apparatus and to the surface being acquired. The problem of extracting the underlying accurate surface from such a point cloud is a fundamental problem in digital geometric modeling and has received a lot of attention over the past 30 years. While there have been a very large number of methods and approaches proposed, it is generally agreed that most of the proposed methods perform well only when the underlying surface is smooth. The main challenge arises in the treatment of sharp features. While there are many methods for the detection and handling of sharp features, (reviewed briefly in the next section), the problem of reconstructing sharp features in point clouds, particularly in the presence of noise, remains a difficult one.

There are two main approaches to this problem, explicit and implicit. Explicit methods [17, 56] typically encode a sharpness value at a given vertex and incorporate this information in the reconstruction formulation. In contrast, implicit methods [36] typically employ robust statistics to treat sharp features as outliers, thus avoiding smoothing over such features.

Explicit detection of sharp features is typically done procedurally [35, 39, 55, 56], often using covariance analysis. For instance, [55] correlates the sharpness value of a point with the distance to the best fitting plane, [35] uses a similar idea looking at the distance traveled while smoothing a point and [5, 56] use Gaussian map clustering to find the sharp points. There are two important limitations with procedural methods: (1) sharp features are difficult to reliably detect in the presence of noise, requiring careful tuning of the parameters [31]; and (2) sharpness is typically defined only at the discrete sampled points and cannot be generalized to resample a more accurate sharp skeleton (i.e. the set of sharp features of a given surface).

Implicit methods also suffer some fundamental shortcomings: (1) there is no way to guarantee that the reconstructions contain points on the sharp skeleton, (2) one cannot explicitly partition the surface into smooth patches (faces), an operation important in CAD/CAM applications, and (3) may suffer from the *step* artifact as illustrated in Figure 2, which is discussed in detail in Section 2.3.

Rather than relying on a one-size-fits-all procedural approach, in this work we propose a data-driven supervised learning solution using convolutional neural networks (CNN) (Figure 1). We have observed that with a small set of annotated data, the CNN is able to reliably learn the local sharp features even in the presence of very large noise. We build this approach within a Moving Least Squares (MLS) framework as follows: (1) we use a proxy MLS surface which allows us to define a sharpness field over a continuous domain; we handcrafted a few representative CAD models with this sharpness field and this forms our training data for the CNN, and (2) we extend the standard MLS reconstruction to use the sharpness field output by the trained CNN to extract sharp geometric features. It is important to note that the only purpose served by the proxy MLS is to provide a parameterized domain for the sharpness field.

*e-mail: p_raina@encs.concordia.ca

Graphics Interface Conference 2018
8-11 May, Toronto, Ontario, Canada

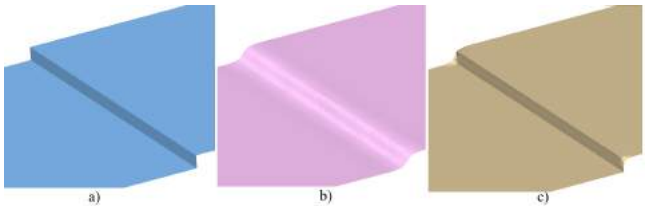


Figure 2: Step artifact of implicit methods. a) Original geometry. b) Implicit MLS formulation [36]. c) Our formulation.

## 2 Related Work

### 2.1 Surface Reconstruction

Prior art on surface reconstruction is very extensive. For a comprehensive review on surface reconstruction we refer the reader to two comprehensive surveys [7, 45]. Below we only provide a brief synopsis, and focus more on the methods closest to ours.

As mentioned before, reconstruction methods are explicit [18, 24, 54, 60] or implicit [2–4, 10, 28, 31, 36, 47]. Explicit methods typically create an initial triangulation from the point cloud [13], thus bringing structure into the data and simplifying the problem. They then apply a filtering operator onto the mesh vertices, using the topology induced by the triangulation. However, this initial triangulation step is notoriously sensitive to noise and outliers and hence very difficult to do correctly using only combinatorial reasoning. As a result, the triangulated model obtained often contains undesirable features such as holes, tunnels or non-manifold vertices [3]. Moreover, the topology of the object's surface is completely determined by the mesh created. So if it is incorrectly constructed it cannot be subsequently fixed.

Implicit methods, as the term implies, express the surface implicitly as a level set either using distance functions [10], indicator functions [28], radial basis functions [12] or moving least squares (MLS) [3, 30, 40]. Radial basis function methods [12] interpolate the data, thus carrying the noise present in the data onto the final surface. Distance functions can be unstable and noisy around sharp features, and in general it is difficult to retrieve the sharpness value explicitly. The same holds for Poisson surface reconstruction. Sharp features are difficult to reconstruct as these methods create an over-smoothed surface lacking in details [11]. MLS methods are discussed in more detail below.

### 2.2 MLS Surfaces

Moving Least Squares techniques [3, 17, 21, 36] define the surface either as the stationary point of a projection operator or the level set of an implicit function [48]. Regardless of the exact definition, the key feature of the MLS approach is the projection operator [30] which provides a simple, elegant and efficient way to move a point close to the surface onto the surface. The MLS method has gained a lot of popularity in the last 15 years as an invaluable tool for surface reconstruction and modeling, spawning a large body of work [14]. MLS methods overcome many of the limitations of both the explicit methods and of the global implicit methods. Being locally defined and coupled with an efficient projection operator they are fast as well as robust. Since they were discovered as a tool for surface reconstruction, over 30 variants have been developed.

The major drawback of the MLS surfaces, like other implicit methods, is that they are by definition smooth and thus sharp details are lost. However, for many applications, explicit detection and handling of sharp features is important [57]. Within the MLS framework this problem has been addressed to some extent by implicitly restricting the MLS kernel only to the smooth parts of the model. Specifically, robust statistics are employed to fit the MLS kernel to the local smooth patch, treating the points across depth discontinuities as outliers and thus removing them from the computation [17, 36]. For example, some of the methods proposed earlier assign a sharpness coefficient to each vertex [26, 31, 57], however the resulting sharp feature surfaces are not satisfactory, particularly in the presence of noise.

### 2.3 Sharp Feature Surface Reconstruction

In principle, implicit methods like MLS are appealing because they handle both the detection and handling of sharp features in one shot. However, they have some shortcomings. First, they typically result in points not necessarily on the sharp skeleton. Second, they may suffer from the *step* artifact as illustrated in Figure 2. Given a geometry where two patches with similar normals are separated by a discontinuity thinner than the kernel radius (Figure 2a), the implicit MLS kernel will likely not be able to differentiate between the normals of the two patches and will tend to group them together resulting in a smoothing artifact (Figure 2b).

Explicit methods detect the sharp regions either by classifying the vertices as sharp [5, 56] or by creating a singularity or sharpness indicator at each vertex [26, 31] and then use it to fit a curve with the sharp points. Sharp feature indicator is defined at points in the point cloud and computed by detecting discontinuities on a function fit to the points. This procedure is typically sensitive to noise in the data.

Our solution differs from the above. As mentioned earlier, our method defines and outputs a continuous sharpness field parameterized over the underlying smooth proxy MLS surface. In a subsequent step we make use of this field to detect and reconstruct sharp features and smooth faces. As we can see (Figure 2c) our method has preserved the sharp edges.

### 2.4 Convolutional Neural Networks (CNN)

In recent years, CNNs have been used extensively for feature detection in image processing, computer vision and medical imaging [34]. One of the reasons why CNNs work so well on images and voxel data is the inherent grid structure of the input data. This structure lends itself naturally to convolution operators that peel out features as the data passes through the layers in the network. Point based geometric data is unstructured, and does not lend itself easily to grid structure without a suitable transformation. Additionally, geometric features are not invariant to some of the affine transformations such as scale, rotations and translations, which makes it even more challenging as the network would need to be trained with all these transformed geometry. Nevertheless, we have seen that in recent years CNNs are being used more and more for geometric problems [22, 27, 41, 42, 59], although they have been applied primarily to object detection and classification, alignment and segmentation flavored problems.

As mentioned before, to suit the CNN model, geometric data has to be represented in a grid structure. We present here a few of the previous approaches reported. Some methods simply used shape descriptors defined on the point clouds and treat them like a 1D structure [22]. Some other recent methods represent data in a volumetric fashion, as a signed distance function defined on a 3D grid [42, 58]. There has also been recent work on reconstructing a volumetric representation of a 3D model, either from a 2D image [51] or from an incomplete voxel grid [23]. Recent depth cameras such as the Kinect device output the geometry basically as a monocular image, making it suitable for CNN processing [52]. Other methods take an arbitrary point cloud and project it in multiple views for a sequence of depth images represented on a regular grid [53]. Still other methods parameterize the data onto a sphere or cylinder that can be unfolded in a grid structure [49, 50].

Geodesic convolution methods and variants [8, 9, 33, 58] use a polar grid in the vicinity of a point to define features in a structured way. We adopt this layout, but with a fundamentally different formulation. We define an MLS based radial grid parameterization that is used to tailor an invariant sharpness feature. This is described in more detail next.

## 3 Overview

The input is typical: a noisy point cloud $PC = \{P_k\}$ with consistently oriented possibly noisy normals $\{N_k\}$. Our first step is to construct a sharpness field defined not only on the point cloud, but also in its vicinity, thus allowing surface resampling as may be needed. The second step is to define an MLS-based surface construction method that conforms to the explicit sharpness field derived in the previous step. The first output is thus a function $SH : \mathbb{R}^3 \rightarrow [0,1]$ that returns a sharpness indicator or a sharpness probability value for any 3D point in the vicinity of the point cloud. The second output is a surface $\hat{S}_R$ computed using a new projection operator $\widehat{PR}_R(P) = \hat{P}$. To facilitate our training of a CNN, we define a planar radial grid $\tilde{G}_{ij}$ and an intermediate proxy surface $\bar{S}_r$, which is a standard smooth MLS surface with a small radius $r$ and used only for representing the sharpness field on a continuous domain. In our prototype we chose the MLS formulation as defined by Levin et al. [2].

Before we describe the pipeline for our reconstruction process, we would like to present some important notations we have used.

- $\{P_k\}$ denotes the points of the original point cloud $PC$.
- $R$ denotes the kernel radius used for the final output surface.
- $r$ denotes the kernel radius used for the proxy surface, typically much smaller than $R$.

Variables annotated with the $\widehat{hat}$ symbol refer to elements of the final output surface $\hat{S}_R$:

- $\widehat{PR}_R$ denotes the projection operator for the final reconstructed surface, which we define later.
- $\hat{P}$ denotes a point on the final reconstructed surface obtained by applying the projection operator $\widehat{PR}_R$ on a point $P$.

Variables annotated with the $\overline{bar}$ symbol refer to elements of the proxy surface $\bar{S}_r$:

- $\overline{PR}_r$ denotes the projection operator [2] for the proxy surface, which is the standard MLS operation.
- $\bar{P}$ denotes a point on this surface obtained by applying the projection operator $\overline{PR}_r$ on a point $P$.

Variables annotated with the $\tilde{tilde}$ symbol refer to elements of the grid. The MLS projection operators mentioned here compute both a 3D point and the surface normal at that point. When not specified, it is assumed that the output of the projection operation is the 3D point. When the normal from the projection is to be used, we will append `.normal` at the end of the variable.

Our pipeline is shown in Figure 3. An MLS construction with a small kernel radius is applied to the original point cloud (Figure 3a) resulting in a proxy surface as illustrated in Figure 3b. A local radial grid is unfolded on this surface (Figure 3c) so as to facilitate creation of the input for the CNN (Figure 3d). The output of the trained CNN is a sharpness field (Figure 3e). An MLS construction with a larger anisotropic kernel uses the sharpness field and the local parameterization to generate the final surface (Figure 3g).

The following sections describe in detail the main steps of the pipeline: Section 4 presents the local MLS parameterization and unfolding which is at the core of all our other operations; Section 5 presents the sharpness field computation; Section 6 presents our final surface reconstruction method; Section 7 discusses how our method can be used to extract the sharp featured surface; Section 8 describes some implementation details and shows our results; Section 9 provides details of comparative evaluation with other machine learning methods and also Pointnet; Section 10 presents some limitations of our method and some directions for future research; Section 11 gives our conclusions.

**Algorithm 1** Spoke unfolding algorithm

**procedure** UNFOLD($\tilde{G}^{3D}_{i*}$) ▷ Input: 3D points of a spoke
  $\bar{G}_{i0} \leftarrow \overline{PR}_r(\tilde{G}^{3D}_{ij})$
  $T_0 \leftarrow I$
  **for** $j = 1..n_j$ **do**
    $\bar{G}_{ij} \leftarrow \overline{PR}_r(T_{j-1} \cdot \tilde{G}^{3D}_{ij})$
    $Q \leftarrow T_{j-1} \cdot \tilde{G}^{3D}_{ij}$
    $N_1 \leftarrow \bar{G}_{i,j-1}$`.normal`
    $N_2 \leftarrow \bar{G}_{ij}$`.normal`
    $T_j = T_{j-1} \cdot$`AlignNormals`$(Q, N_1, N_2)$
  **end for**
  **return** $\bar{G}_{i*}$ ▷ Output: grid points on $\bar{S}_r$
**end procedure**

## 4 Local Parametrization

At the foundation of our method is the local parameterization around a point $P_k$, which is obtained by mapping points from the point cloud within radius $R$ of $P_k$ onto the 2D domain shown in (Figure 4a), what we refer to as unfolding. This parameterization provides a local structure enabling more advanced queries similar to the ones found in meshes and grids, and is key to both our operations: computing the sharpness and computing the MLS projection.

It is important to note that a simple Euclidean projection on the plane centered at a point $P_k$ with the normal $N_k$ would not work for the unfolding operation as some cells of the grid would be flipped due to bending of the radial spoke, as can be seen in Figure 4d.

We use the proxy surface $\bar{S}_r$. We compute the parameterization around a point $P_k$ by first computing a radial grid of points $\tilde{G}_{ij}$ in 2D centered at the origin as shown in Figure 4a. Index $i$ is the angular parameter and index $j$ is the radial parameter. $n_i$ and $n_j$ are user-defined and denote the number of angular and radial divisions, respectively.

We align this radial grid such that the center 3D point, $\tilde{G}_{k00}$, is at the origin of the grid and the normal of the grid's plane is aligned with the normal at that point. Now the points $\tilde{G}_{ij}$ have a canonical embedding in 3D and are denoted by $\tilde{G}^{3D}_{kij}$ (Figure 4b).

The points $\tilde{G}^{3D}_{ij}$ close to the origin (i.e. the first ring) can be projected directly onto $\bar{S}_r$ using the $\overline{PR}_r$ operator, but points farther away cannot as they are too far away from $\bar{S}_r$ for $\overline{PR}_r$ to work (Figure 4c, d). Therefore, we carefully unfold all points $\tilde{G}^{3D}_{i*}$ on each spoke $i$ one by one by applying a cumulative transformation to bring it back closer to the surface (Algorithm 1 ).

The function `AlignNormals`$(Q, N_1, N_2)$ outputs a transformation matrix corresponding to the shortest rotation around the point $Q$ that overlaps $N_1$ onto $N_2$. Computing this transformation is a well known procedure. Because the rotations are not around the origin, our transformations $T_i$ are technically combinations of translations and rotations. So, all the matrix-point and matrix-vector operations are performed in homogeneous coordinates. The embedding of the points of the original point cloud in this grid is done by projecting them onto $\bar{S}_r$ and detecting which grid cell they landed on. We evaluated this for increasing noise and since it depends mainly on the MLS projection operator $\bar{S}_r$, we found that it breaks only for higher noise levels, with noise amplitudes exceeding a quarter of the entire edge length (Figure 5).

Our radial grid bears a superficial similarity to [32, 46] where a similar structure was used. However, the way we define the inputs and perform the unfolding procedure are fundamentally different.

## 5 Sharpness Field Computation and Extraction

As sharp features are discontinuities in the normal field, we devise a scheme where the inputs to the CNN are angle differences between

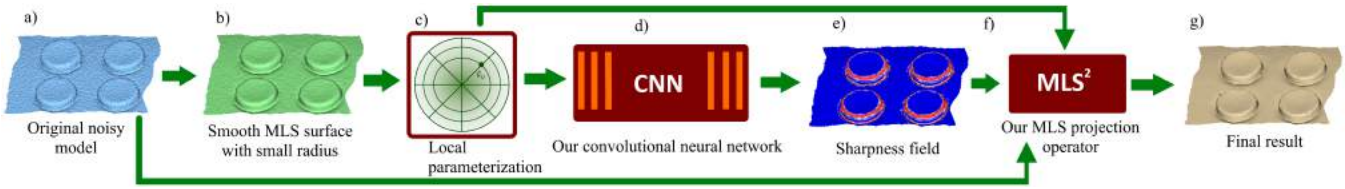


Figure 3: Our pipeline: a) Original noisy model. b) Smooth MLS proxy surface with small radius. c) Per point local radial grid. d) CNN for outputting the sharpness field. e) Sharpness field. f) Our MLS formulation that uses the sharpness field, the local radial grids and the original point cloud to generate g) the final surface.

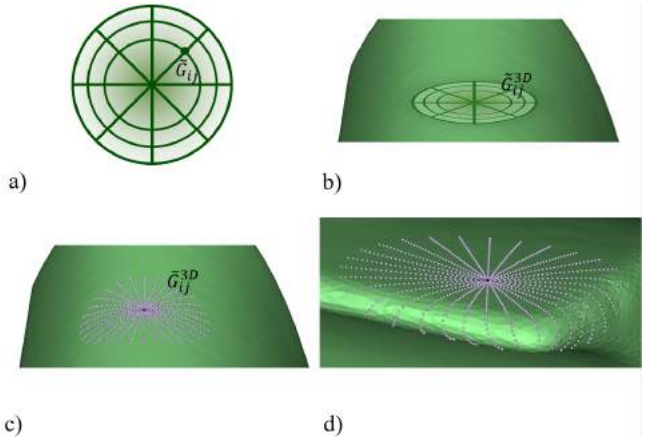


Figure 4: Local Parameterization:$i$ is the angular parameter and $j$ the radial parameter. a) Planar radial grid b) Planar radial grid aligned with the source point and normal. c,d) Planar radial grid unfolded onto the surface. d) shows that the unfolding procedure is very robust even in extreme cases.

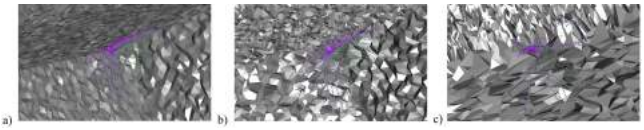

Figure 5: Unfolding breaks only for higher noise levels exceeding 5%: (a) 0.5%, (b)1.5%, (c) 5%

the normals at the queried point $P_k$ and the normals at a set of its neighbours. The output is a scalar value, the probability of the point belonging to a sharp feature.

Before finalizing on the use of a CNN [34], we experimented with several classical learning methods such as linear regression [34], linear SVM [16], random forests [25] and gradient boosting [19].

The points on the radial grid $\tilde{G}_{ij}$ are projected onto the MLS surface $\bar{S}$. The resulting points $\bar{G}_{kij}$ are on a regular grid and also have normals. The inputs to the CNN are the angles $\alpha_{kij}$ between $\bar{G}_{kij}$ and the origin $\bar{G}_{k00}$. These are angle differences and are scale, rotation and translation invariant. Note that the input angles come from $\bar{S}_r$, which is a smooth version of the point cloud, however the small radius filters only the very high frequency noise without losing the salient details, as can be seen from the results. To make it invariant to the choice of the local coordinate frame, we sort the angles with the same radius and we group the inputs by radii ( [44]). More formally, the input to the CNN for a point $P_k$ of the point cloud is $I(kij) = \alpha_{kij}$ where $i = \texttt{sortperm}_j(k \bmod n_i)$, $\texttt{sortperm}_j$ is a sorting permutation of the values on radius $j$ and $j = k \texttt{ div } n_i$. In all our examples we used $n_i = n_j = 30$. Sample inputs for the CNN are shown in Figure 6.

For choosing the CNN architecture, we tried out different network configurations of convolutional layers, max pooling layers and fully connected layers using a ReLU activation function and a Glorot

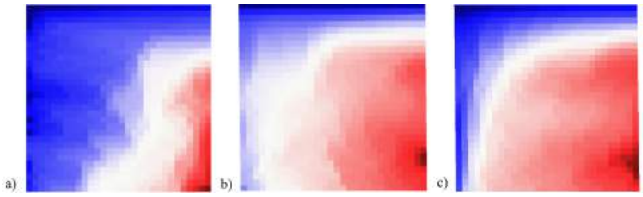


Figure 6: 2D Input to our CNN for a single point; Inner to outer rings go from top to bottom. For better visualization we show it on a tri-scale of low (blue), medium (white) and large (red) angles. a) A point on a smooth patch. b) A point on a corner. c) A point on a sharp edge.

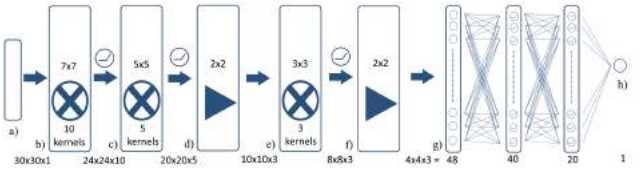


Figure 7: Architecture of our CNN network. a) 2D Input b,c,e) Convolutional layers. d,f) Max pooling layers. g) A set of two fully connected layers. h) Output.

weight initialization scheme [20]. The best results were obtained for the configuration with 3 convolutional layers, 2 max-pooling layers and 2 fully connected layers. Figure 7 illustrates schematically the CNN we use in our method.

We trained the network on the set of representative models shown in Figure 8. To create the training models, we painted sharpness values manually with 1 on the edges and 0 in the smooth regions. We then applied a sharp decreasing kernel around the edges to ensure continuity of the training sharpness field. Since the training models had to be hand-crafted for the sharpness field values, we chose the model set carefully to cover most edge vertex configurations encountered in CAD models. We created 2 more noisy instances of each model. For all models the added noise is defined by a percentage (between 1% to 5%) of the length of the diagonal of the bounding box. While each of the models has only around $2000-3000$ points, there is no restriction on the test model size; it can be orders of magnitude larger. Some of our test models have over 100K points. The results and commentary is provided in section 8.

One additional problem is the choice of the radii, both for the trained models and for the query models. We found experimentally that the best results we obtained were for a radius 8 times the median length between all pairs of closest points. In Figure 8 we show this radius on a model.

## 6 Surface Reconstruction

Although our work is not tied to a specific MLS formulation, for our proof of concept prototype implementation we used one of the early MLS formulations [30]. Given a set of points in 3D space $\{P_k\}$ the basic MLS formulation [30] describes a smooth surface that consists of the stationary points of a projection operator. The challenge in this setting is that it is difficult to know without a connectivity structure if

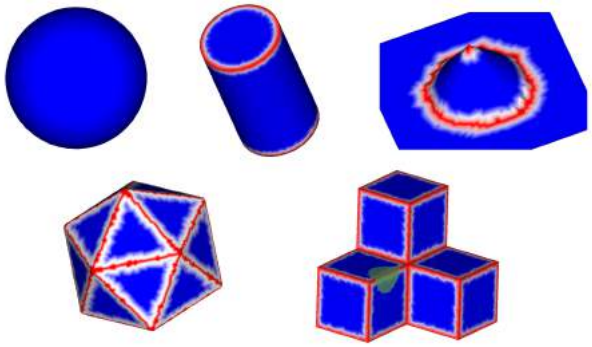

Figure 8: Initial training set for our CNN network rendered with the user defined sharpness value. The small green patch indicates the radius of the grid used to train the network.

a neighboring point is inside the same smooth fragment of the model or if it is across a sharp feature and is located on a different smooth fragment. This is illustrated in Figure 9a: the center point belongs to one patch, but other points within its radius cross several sharp boundaries. Therefore, we have used the following approach. Using the parameterization defined in Section 4 we compute the weights first on the radial grid of the parameterization $\bar{G}_{kij}$ (shown as white points in Figure 10a) and then transfer the weights to the points in the point cloud $P_k$ shown in Figure 10b. More specifically, we accomplish the surface reconstruction for each point $P_k$ in 3 steps:

1. We compute sharpness-dependent MLS weights $\lambda_{kij}$ on the grid points $\bar{G}_{kij}$. (Section 6.1)

2. We transfer the weights $w_{kl}$ of every neighbour $P_l$ of $P_k$ within radius $R$. (Section 6.2)

3. We perform the projection. (Section 6.3)

### 6.1 Weight Computation

Algorithm 2 is used for weight computation for all points, including the point whose neighbourhood crosses sharp boundaries as shown in Figure 9a. It basically consists of two passes. In the first pass, markers are placed along the spokes (Figure 10a, rendered in magenta) signaling the detection of a sharp feature. The distance from the center to this marker is simultaneously computed. The sharpness profile along each spoke is shown in Figure 9. Figure 9c) shows 4 example plots of sharpness values from 4 spokes. The top left plot illustrates a spoke that crosses a sharp feature. The top right plot illustrates a spoke that crosses a feature and continues to another even bigger one. (This situation can happen around corners.) Since these functions are fairly smooth, the sharp feature is simply taken to be the first local maximum with respect to the sharpness at the center point that is larger than 0.5. The bottom right plot shows a spoke that crosses an area with no sharp features.

In the second pass, weights $\lambda_{ij}$ are computed as a Gaussian function with mean 0 and standard deviation $\sigma$. $\sigma$ adapts to the local features and thus its value is set depending on the distance to a sharp feature. Figures 9d and 9e show the profile of the weights $\lambda_j$. It can be observed that spokes which intersect sharp boundaries are shorter, but $\lambda_{ij}$ still varies smoothly towards 0.

### 6.2 Weight Transfer and Final Projection

The weights computed in the previous step have to be transferred to the original point cloud in order to apply the final projection operation. Note the points $\hat{G}_{ij}$ are the grid points $\tilde{G}_{ij}$ mapped onto the $\hat{S}_r$ surface using the unfolding algorithm (Algorithm 1). The points are organized as a regular grid, so we create the triangulation

**Algorithm 2** Computing weights $\lambda_{ij}$ along the spoke $i$ centered around point $P_k$

**procedure** COMPUTEWEIGHTS($\tilde{G}^{3D}_{i*}$, $SH(\cdot)$)
  $d_j \leftarrow 0$
  $d \leftarrow 0$
  $barrier \leftarrow -1$
  **for** $j = 1..n_j - 1$ **do** ▷ First pass
    $d+ = \|\tilde{G}^{3D}_{ij} - \tilde{G}^{3D}_{i,j-1}\|$
    $d_j = d$
    $b_1 \leftarrow SH(\tilde{G}^{3D}_{ij}) > 0.5$
    $b_2 \leftarrow SH(\tilde{G}^{3D}_{ij}) \geqslant SH(\tilde{G}^{3D}_{i,j-1})$
    $b_3 \leftarrow SH(\tilde{G}^{3D}_{ij}) \geqslant SH(\tilde{G}^{3D}_{i,j+1})$
    **if** $b_1$ **and** $b_2$ **and** $b_3$ **then**
      $barrier \leftarrow j$
      **break**
    **end if**
  **end for**
  $\sigma \leftarrow d/3$
  **for** $j = 0..n_j$ **do** ▷ Second Pass
    **if** $j < barrier$ **then**
      $\lambda_{ij} = Gauss_\sigma(d_j)$ ▷ Gaussian kernel
    **else**
      $\lambda_{kij} = 0$
    **end if**
  **end for**
  **return** $\lambda_{ki*}$
**end procedure**

$$T = \{\triangle(\hat{G}_{i,j}, \hat{G}_{i,j+1}, \hat{G}_{i+1,j+1}) \bigcup \triangle(\hat{G}_{i,j}, \hat{G}_{i,j+1}, \hat{G}_{i+1,j})\}$$
$$\forall i \in [1..n_i], j \in [1..n_j]$$

Note that in the above notation $i+1$ and $j+1$ are taken $mod\, n_i$ and $mod\, n_j$ respectively and for the first sector ($j = 0$), we do not add $\triangle(\hat{G}_{i,0}, \hat{G}_{i+1,1}, \hat{G}_{i+1,0})$ as it is degenerate. The weights $\lambda_k$ are normalized to sum up to one and are used in the smooth MLS projection [2].

### 6.3 Projection onto the Sharp Skeleton

Points that are on or very close to the peak of the sharp feature are ambiguous as it is difficult to decide which side of the sharp feature they belong to. These points can be simply detected in Algorithm 2 by placing an upper bound on the minimum $d$ over all the spokes. We devise a simple procedure that moves them onto the sharp skeleton. Our sharp skeleton consists of the set of surface points where the sharpness value is maximum. Using our local parametrization, we iteratively move the points along the surface in the direction of steepest ascent until a local maximum is reached. Additional points can be added to the sharp skeleton using this procedure if needed.

## 7 Reverse Engineering to CAD/CAM Models

Reverse engineering scanned point clouds into 3D geometric models (not just dense triangle meshes) is important and essential particularly on point-based surfaces if these scans have to find use in CAD/CAM applications [6]. Presently, this is mostly a manual process with the user interactively segmenting the points into patches/faces and their bounding edges. An important advantage of our $MLS^2$ formulation is that it can be used to automatically derive additional information about the geometric structure of the scanned object's surface. Some applications would be: (1) segmenting the model into smooth patches and (2) explicitly extracting the sharp

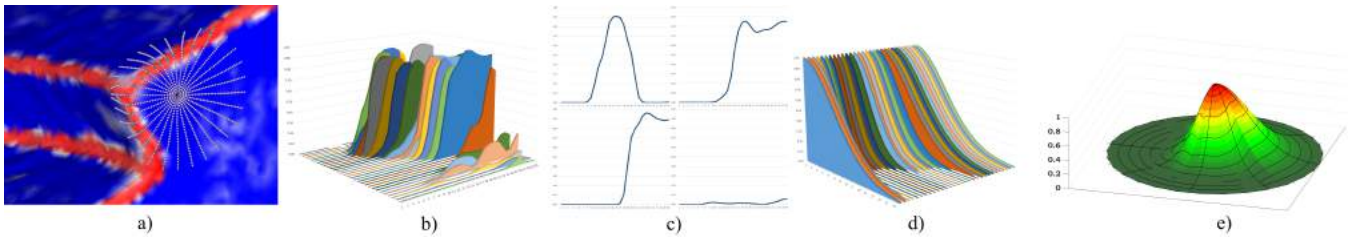


Figure 9: Weight computation based on sharpness indicator field. a) A point whose neighbour points cross a sharp feature. b) Sharpness value profile for each individual spoke. c) 4 example plots of sharpness values from 4 spokes showing various possible configurations. d) Weights profile $\lambda_{kij}$ for each individual spoke. e) Weights profile in a radial graph that outlines the anisotropic kernel.

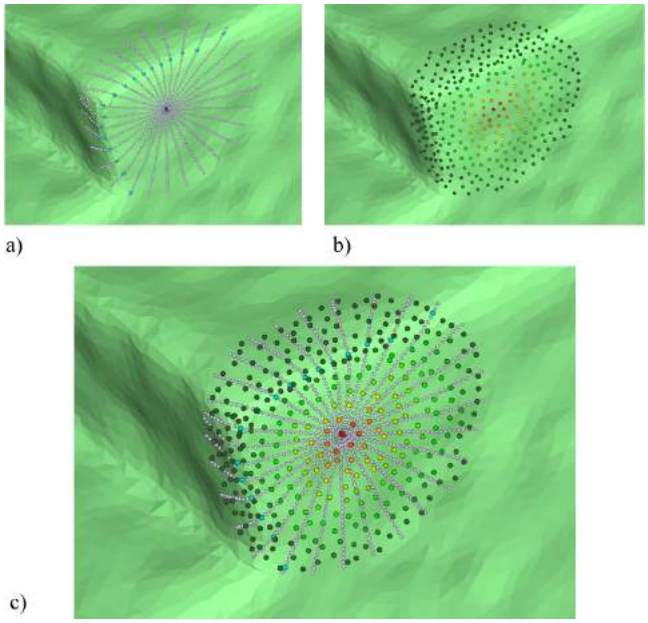


Figure 10: Unfolding Illustrated: a) The center point and the radial grid embedding showing the barrier points rendered in cyan. b) The local neighbourhood with MLS weights color coded on a logarithmic scale: red corresponds to high weight and green corresponds to low weight. Note that the weights of points across the discontinuity have 0 weight. c) Juxtaposition of the points in a) and b).

skeleton with edges and vertices. Our results for both operations are shown in Figure 11.

### 7.1 Surface Segmentation

The surface segmentation is done by selecting a random seed point and performing a flood-fill operation on the graph induced by the $MLS^2$ kernel where two points belong to the same patch if their mutual $MLS^2$ weights are non-zero or in other words, if they are within one radius distance from each other and they are on the same side of all nearby sharp features. The new $MLS^2$ kernel presented in section 6 has by construction the property that if two vertices, however close to one another, are on different sides of a sharp feature, their $MLS^2$ weights with respect to each other will be 0.

### 7.2 Sharp Skeleton Extraction

In the sharpness field provided by the CNN, sharp features correspond to ridges and peaks. This structure of the sharpness field can naturally be used to construct the sharp skeleton. We proceed iteratively as follows. We start with a vertex close to the sharp skeleton. We move it onto the sharp skeleton using the projection onto the sharp skeleton procedure outlined in Section 6.3. From here we construct the sharp skeleton path by walking on the proxy surface

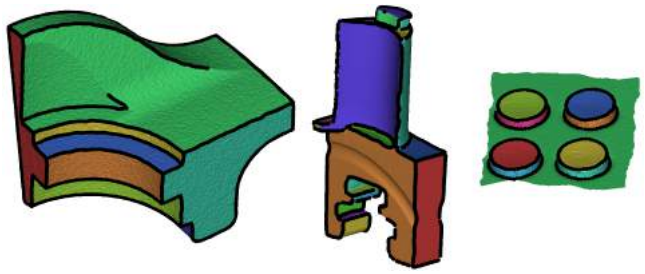

Figure 11: Reverse engineering of CAD models: smooth patch and sharp skeleton extraction shown on the original point cloud.

along the ridge of our sharp skeleton function. We stop when either: (1) we reach the same point (loop), (2) the sharp skeleton vanishes (the fan disk shows an example of this situation) or (3) we connect with another existing sharp skeleton path at a corner. We repeat this procedure until we cover all sharp skeleton paths. It is important to note that this procedure will reveal automatically all the sharp feature types such as corners, darts, creases, etc.

## 8 Discussion

**CNN Architecture:** We used PyTorch [38] to program and run the CNN. The max-pooling layers helped with noise resilience. However, having more than two max-pooling layers reduced the quality of the resulting field, possibly because it prematurely reduced the resolution of the input data. Having more than 2 fully-connected layers led to overfitting. The training of our network and data took around half an hour for 100 epochs, using Adam [29] to train the network with a learning rate of 0.0003 and a minibatch size of 32. Batch normalization was not found to significantly improve results, likely because the input values are all angles normalized to the range [0,1]. Besides the 2D CNN, we also experimented with a different network structure, where the spokes of our radial grid were fed to parallel 1D convolutional layers. This network was significantly harder to train and did not improve results.

**Training Set and Strategies:** In terms of the methodology of the training process there are two dimensions: (a) variety in the local shape and (b) variety in the noise. By variety in the local shape in this context we mean training on adequate set of models in order for the network to handle as general inputs as possible. Also, training on the same data with added noise will make the network more robust.

We systematically explored both these dimensions and we found out that adding additional models beyond these 5 shapes shown in Figure 8 in the training set did not improve the results. This is also because our training is per point and therefore even if we have only 5 models each model has thousands of points. However, training on the models with added noise yielded a cleaner sharpness map, but it resulted in parts of some edges being missed, which is not surprising since training on noisy data makes the detection less sensitive to subtle edges or creases. As illustrated in Figure 12, the results from

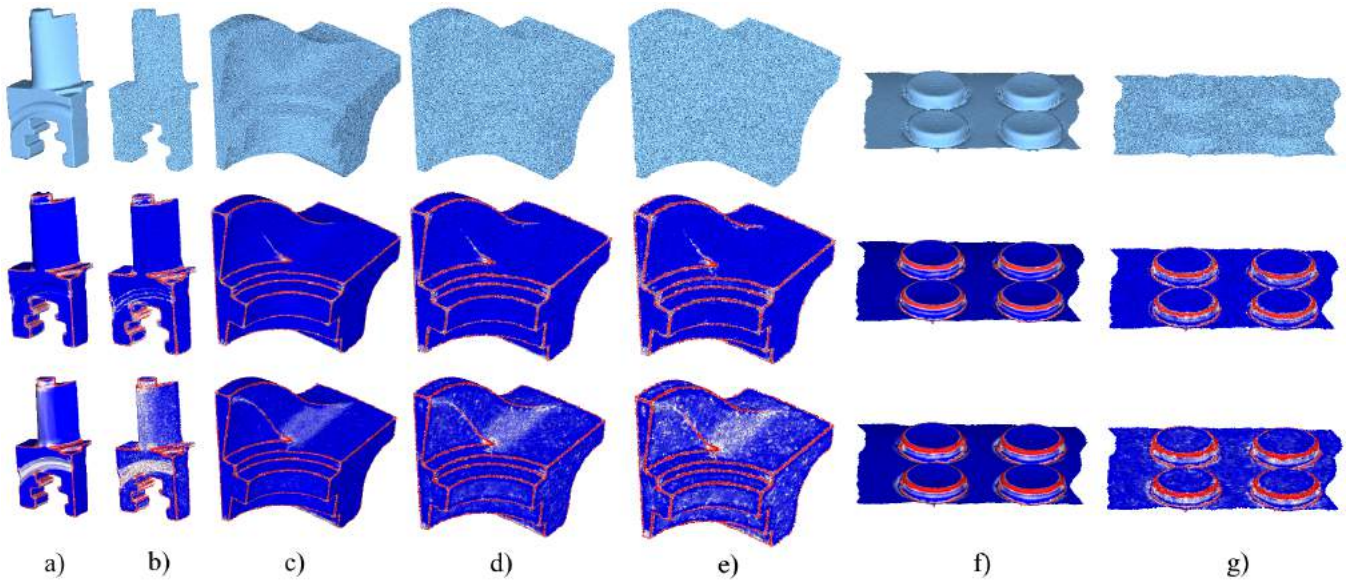


Figure 12: Results of our sharp feature extraction. a) The original noisy blade model. b) The original blade model with added $0.01$ noise. c) The fan disk model with $0.005$ added noise. d) The fan disk model with $0.01$ added noise. e) The fan disk model with $0.015$ noise. f) A phone keypad scan fragment (original) g) Same model as in f) with $0.01$ added noise. The top row contains the original models. The rows below contain the models rendered showing the sharpness value. The middle row was generated by a CNN trained on both clean and noisy data while the CNN used for the bottom row was trained with only clean data.

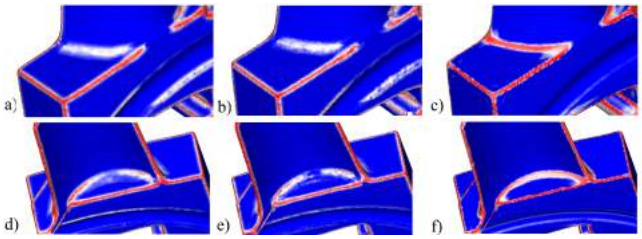


Figure 13: Comparison with gradient boosting and random forests. a,d) Gradient boosting results. b,e) Random forest results. c,f) CNN results.

training only on the clean data are more noisy, but nuanced edges are better preserved.

**Binary Classification vs. Non-binary Reconstruction:** Most existing explicit sharp features detection methods perform a binary classification on the points in the point cloud. The disadvantage of such a classification based approach from the perspective of a surface reconstruction problem is that it makes it difficult to retrieve explicitly the sharp skeleton, since the probability that a point lies exactly on the sharp skeleton is very small. Classification based method do not provide a method to re-sample the surface on or close to a sharp feature. Conceptually, in our approach, the curve that is a sharp feature on an otherwise smooth surface is defined as a maximum ridge of our sharpness field. Even if this is an implicit definition we provide a method to reconstruct these features explicitly as illustrated in Figure 12.

**Initial MLS Surface:** Our choice of using an intermediate smooth MLS surface may seem counter-intuitive and may raise concerns about over smoothing. The main benefit of having this initial surface is that it provides a flexible and scalable mechanism to do local parameterization and surface resampling, which is critical in our goal of defining a continuous smoothness field rather than a per-sampled point binary classification. It is important to repeat here that this proxy surface plays no geometric role in the final surface reconstruction, rather it is just parameterization domain that enables us to sample the sharpness field at an arbitrary location by using the MLS projection operator.

**Parameter Values:** For our experiments, we used small radius $r = 3$ and large radius $R = 8$. The unit is the median length between a point and its closest neighbor. Our radial grid is $30 \times 30$ for all models.

## 9 Evaluation and Results

### 9.1 Comparative Evaluation of Sharp Feature Detection

Our sharp feature extraction results are shown in Figure 12. Wec can see that the sharp features (in red) are extracted reliably even in the presence of extreme noise.

To evaluate our sharpness field, we designed an ideal sharpness field for the fandisk model. We chose this model because it has points clearly located on sharp edges, as well as a variety of different types of sharp features. Inspired by ( [56]:Fig.7), we created 3 wedge models with different sharp feature profiles to add to our test set, Figure 15. We manually marked edges with sharpness value 1 and then diffused the sharpness value to nearby points, with the sharpness value diminishing exponentially with distance. Several different machine learning models were tried with the same input feature, to evaluate the best method to use for sharp feature detection. The quality metric we use for validating the sharpness field is the mean square error of the sharpness value at each point.

Table 1: Comparison of Error in Different ML Techniques

| Model | Fandisk | Wedges |
|---|---|---|
| Linear Regression | 0.015019 | 0.044839 |
| Linear SVM | 0.009747 | 0.017954 |
| Random Forest | 0.006219 | 0.005651 |
| Gradient Boosting | 0.010869 | 0.008251 |
| CNN | **0.005976** | **0.005275** |

We see from Table 1 that CNNs have a slight advantage for noisy data. Further, we see from (Figure 14 of blade model) that qualitatively, the CNN does a better job of capturing creases.

**Comparison with gradient boosting and random forest.** We experimented with various supervised learning techniques. While

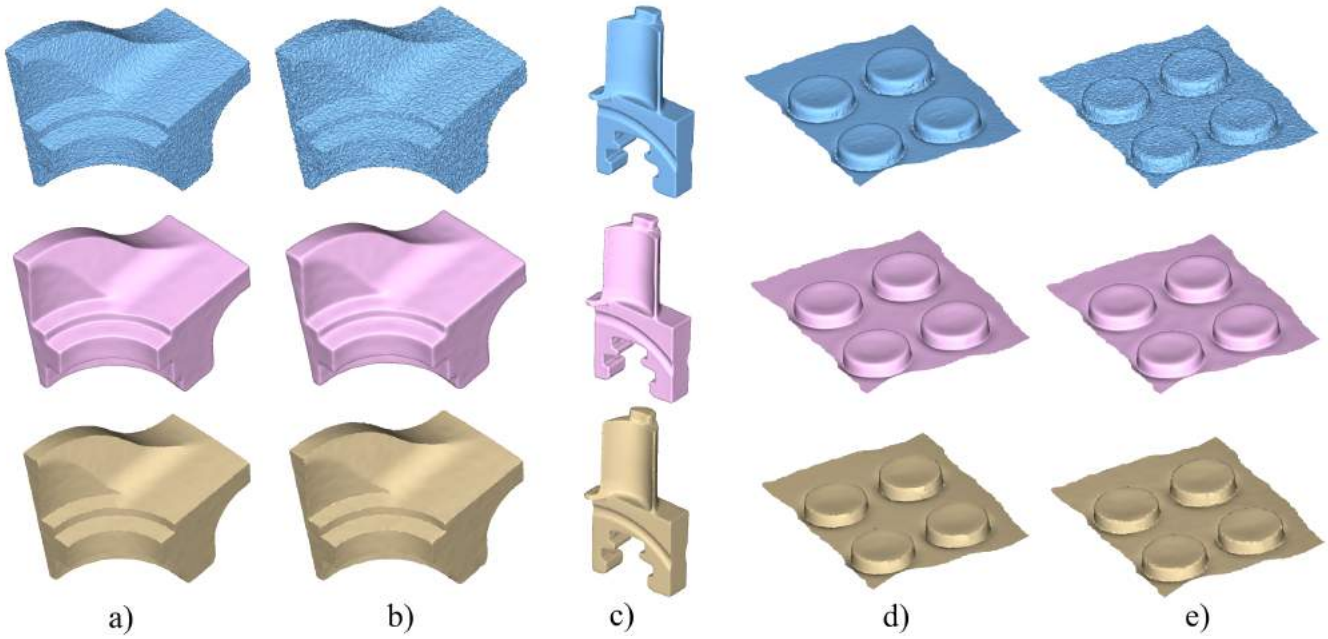


Figure 14: Reconstruction comparisons and results. a) Fan disk with $0.01$ noise added. b) Fan disk with $0.015$ noise added. c) Blade (original model). d) Phone keypad scan fragment (original). e) Phone keypad scan fragment with $0.01$ noise added. Top row: Original model. Middle row:

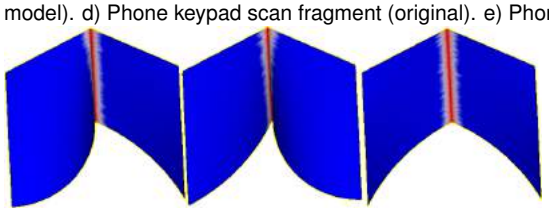

Figure 15: Wedge models with different sharp feature profiles for testing inspired by [56]:Fig. 7

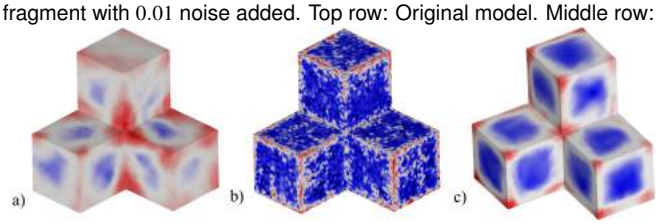


Figure 16: PointNet Results. a) Using transfer learning. b) Training PointNet from scratch. c) Using a PointNet-based autoencoder.

CNN was the overall winner, gradient boosting and random forest came a close. For gradient boosting we used the same input structure as for CNN and we used the open source LightGBM library by Microsoft. They tend to miss fuzzier edges as shown in Figure 13 allowing patches to bleed to a neighbouring patch. Second, while the sharpness field is always high on the points close to the sharp skeleton, it is not maximum on the points corresponding to the sharp edge, thus creating problems in the reconstruction.

**Comparison with Pointnet for Sharp Feature Detection** We had initially experimented with using automatically learned input features for sharp feature detection instead of hand-crafting a feature. The recently published PointNet [41] architecture seemed to be a promising deep learning approach for feature learning on point clouds, due to its previous success in point cloud classification and segmentation. This approach allows feeding raw Cartesian coordinates of points as input to the network. However, using this architecture carries a severe restriction, namely that the number of input points for the neural network is fixed at training time. Therefore, in order to use PointNet, we must sample a fixed number of nearest neighbors around each point being tested, and use this neighborhood as the input. It also seems unlikely that PointNet would perform well on point clouds with a different scale from those seen in the training set, so we normalized the coordinates of our training data to the range [-1, 1].

We tried three approaches for using the PointNet architecture for sharp feature detection:

1. Using a Pointnet pre-trained for point cloud classification, and fine-tuned for sharp feature detection by replacing and re-training the last two fully-connected layers, Figure 16 left.

2. Training a Pointnet from scratch for sharp feature detection instead of classification, Figure 16 middle.

3. Training a Pointnet-based deep single-class autoencoder on neighborhoods of sharp edges using the architecture given in [37], and feeding the encoder portion as input to two fully connected layers which are then trained to detect sharp features, Figure 16 right.

In each case, the network failed to learn sharp feature detection, even on the training data. Figure 16 shows the sharp feature detection result obtained on one training shape. The authors of [41] mention that PointNet learns to summarize a shape by a sparse set of key points. As can be seen from Figure 16, the embedded representation learned by the PointNet architecture, when fine-tuned to sharp feature detection, only learns to detect corners and not edges. We speculate that this is because the sparse set of key points learned by the network only provides enough context for corner detection.

By contrast, we show that with our hand-crafted feature, the CNN can learn to detect edges not only on the training set but also on complex, previously unseen shapes.

### 9.2 Surface Reconstruction

Our reconstruction results are shown in Figure 14 (bottom row). The reconstruction completely preserves the sharpness of the sharp edges, including some of the more subtle ones. This is partly because the

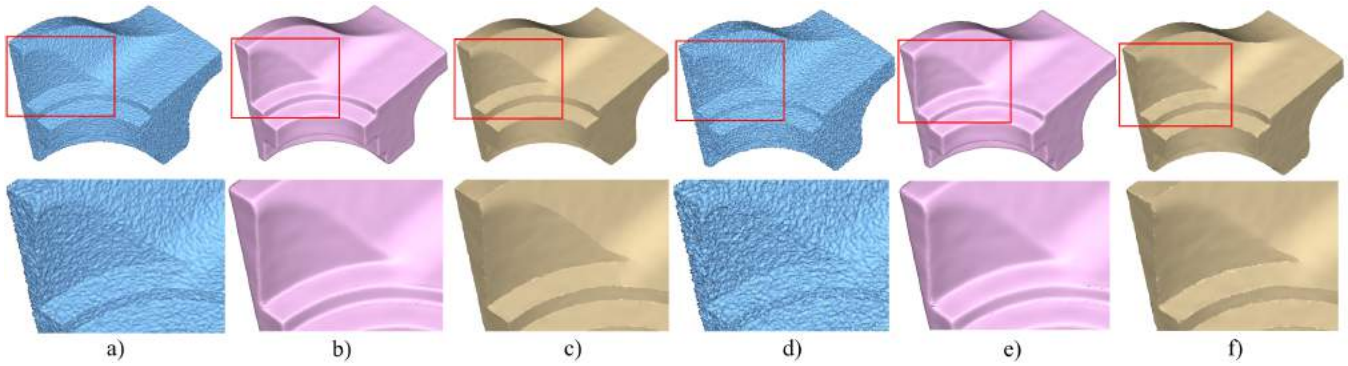


Figure 17: Reconstruction comparisons details. a) Fan disk with 0.01 noise added b) RIMLS result. c) Our result. d) Fan disk with 0.015 noise added. e) RIMLS result. f) Our result.

projection operator is combined with the accurate explicit detection of the sharp features, and more importantly due to the fact that our field is not restricted to just the points in the point cloud, thus allowing re-sampling of the sharp edge in the vicinity of the points in the point cloud. The reconstruction is stable with varying levels of noise, although understandably some features may be lost depending on the level of noise.

We performed direct comparisons with RIMLS, one of the state of the art MLS-based reconstruction algorithms [36]. We used their implementation available in Meshlab [15], with a kernel having the same size as ours and the default sharpness value. On the fan disk model, our algorithm seems to preserve better the sharpness of the edges including the more subtle edges as illustrated in Figure 17. On the blade model, the results are somewhat comparable. On the phone keypad, our algorithm preserves the sharpness better. In addition, unlike RIMLS, our method does not suffer from the *step* artifact. These results can be seen in Figure 2.

The fan disk and the blade models have also been used as example models in previous work. Avron et al. [4] provide visual comparisons (Figures 6 and 7 in their paper) using the same blade model as ours in Figure14c), as well as a fan disk with 0.01 noise corresponding to our result from Figure 14a). On the blade model, their method, as well as the others that they are comparing with, have difficulties dealing with the left side of the blade, which is narrow and sharp. We do not experience these problems and we show in Figure 4d) that our method works correctly even in such a challenging case.

### 9.3 Reverse Engineering of CAD/CAM Models

Our method can be used naturally for this application as was shown in Figure 11. Our method extracts both a segmentation of the patches as well as the explicit sharp skeleton graph. It works well on models with sharp edges even in the presence of noise but the results are not as good when the edges are more fuzzy (Figure 11, middle).

## 10 Limitations and Future Work

Even though our sharpness indicator captures sharp features well, our surface reconstruction only takes into account the points with high sharpness values. However, features that are neither completely sharp nor completely smooth also need to be reconstructed properly. Currently, the sharpness values for many of these features are not high enough, as illustrated by some of the white regions in the second row of Figure 12. We would like to address this by adapting the kernel to use the integral of the sharpness values across a spoke. This way, even if the sharpness values are small locally, they accumulate, thus reducing the kernel size in that direction. Another issue is the need to directly synthesize points which are exactly on the sharp edges. Our current method projects nearby points onto the sharp edges, but suitable nearby points may not always be available. Since this would require a high quality explicit representation of the sharp skeleton, it is more challenging to achieve. We would like to investigate such problems further.

## 11 Conclusion

We present a new algorithm within the MLS methodology to reconstruct point cloud surfaces while preserving sharp feature. Explicit representation of sharp features is important in many CAD/CAM applications. The distinctness in our method arises from the fact that rather than associating a sharpness attribute to just the points in the point cloud, we define a sharpness field whose value can be queried for any point in space within the vicinity of the point cloud surface. The sharpness field is derived by training a CNN with suitably created synthetic data and is parameterized by the underlying smooth MLS surface. In a second step the field is used to detect and recover the surface including sharp features by formulating a new MLS projection operator. We show comparative results for a number of models with sharp features and also show that our method works well even in the presence of high noise in the point cloud. Clearly this is a first step in this direction and the method could be improved to yield even better results.

## References

[1] A. Adamson and M. Alexa. Point-sampled cell complexes. In *ACM TOG*, vol. 25, pp. 671–680. ACM, 2006.

[2] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Proceedings of the First Eurographics conference on Point-Based Graphics*, pp. 149–155, 2004.

[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, 9(1):3–15, 2003.

[4] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or. l1-sparse reconstruction of sharp point set surfaces. *ACM TOG*, 29(5):135, 2010.

[5] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *Digital Image Computing: Techniques and Applications (DICTA), 2015*, pp. 1–8. IEEE, 2015.

[6] P. Benkő, R. R. Martin, and T. Várady. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839–851, 2001.

[7] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva. State of the art in surface reconstruction from point clouds. In *EUROGRAPHICS star reports*, vol. 1, pp. 161–185, 2014.

[8] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 3189–3197, 2016.

[9] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers. Anisotropic diffusion descriptors. In *CGF*, vol. 35, pp. 431–441. Wiley Online Library, 2016.

[10] F. Calakli and G. Taubin. Ssd: Smooth signed distance surface reconstruction. In *CGF*, vol. 30, pp. 1993–2002. Wiley Online Library, 2011.

[11] R. Campos, R. Garcia, and T. Nicosevici. Surface reconstruction methods for the recovery of 3d models from underwater interest areas. In *OCEANS 2011 IEEE - Spain*, pp. 1–10, June 2011. doi: 10.1109/Oceans-Spain.2011.6003633

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 67–76. ACM, 2001.

[13] F. Cazals and J. Giesen. Delaunay triangulation based surface reconstruction. In *Effective computational geometry for curves and surfaces*, pp. 231–276. Springer, 2006.

[14] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, and S.-Y. Jin. A survey of methods for moving least squares surfaces. In *Volume Graphics*, pp. 9–23, 2008.

[15] P. Cignoni, M. Corsini, and G. Ranzuglia. Meshlab: an open-source 3d mesh processing system. 2008.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM TOG*, 24(3):544–552, July 2005. doi: 10.1145/1073204.1073227

[18] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *ACM TOG*, vol. 22, pp. 950–953. ACM, 2003.

[19] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

[20] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, vol. 9, pp. 249–256, 2010.

[21] G. Guennebaud and M. Gross. Algebraic point set surfaces. In *ACM TOG*, vol. 26, p. 23. ACM, 2007.

[22] K. Guo, D. Zou, and X. Chen. 3d mesh labeling via deep convolutional neural networks. *ACM TOG*.

[23] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. *arXiv preprint arXiv:1709.07599*, 2017.

[24] K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. In *CGF*, vol. 23, pp. 391–400. Wiley Online Library, 2004.

[25] T. K. Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 278–282. IEEE, 1995.

[26] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang. Edge-aware point set resampling. *ACM TOG*, 32(1):9, 2013.

[27] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[28] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP*, vol. 7, 2006.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] D. Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*. Springer, 2004.

[31] Y. Lipman, D. Cohen-Or, and D. Levin. Data-dependent mls for faithful surface approximation. In *SGP*, SGP '07, pp. 59–67. Eurographics Association, Aire-la-Ville, Switzerland, 2007.

[32] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45, 2015.

[33] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. Technical report, 2015.

[34] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[35] J. Nie. Extracting feature lines from point clouds based on smooth shrink and iterative thinning. *Graphical Models*, 84, 2016.

[36] A. C. Oztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *CGF*, vol. 28. Wiley Online Library, 2009.

[37] I. M. L. G. Panos Achlioptas, Olga Diamanti. Learning representations and generative models for 3d point clouds, 2018.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. *NIPS 2017 Workshops*, 2017.

[39] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. In *CGF*, vol. 22, pp. 281–289. Wiley Online Library, 2003.

[40] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM TOG*, 22(3):641–650, 2003.

[41] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[42] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[43] P. Reuter, P. Joyot, J. Trunzler, T. Boubekeur, and C. Schlick. Surface reconstruction with enriched reproducing kernel particle approximation. In *Point-Based Graphics, 2005*, pp. 79–87. IEEE, 2005.

[44] K. M. Saipullah, D.-H. Kim, and S.-L. Lee. Rotation invariant texture feature extraction based on sorted neighborhood differences. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pp. 1–6. IEEE, 2011.

[45] O. Schall and M. Samozino. Surface from scattered points. In *a Brief Survey of Recent Developments. 1st International Workshop on Semantic Virtual Environments, Page (s)*, pp. 138–147, 2005.

[46] R. Schmidt, C. Grimm, and B. Wyvill. Interactive decal compositing with discrete exponential maps. *ACM TOG*, 25(3):605–613, 2006.

[47] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo, and D. Cohen-Or. Interactive topology-aware surface reconstruction. *ACM TOG*, 26(3):43, 2007.

[48] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Siggraph 2005 Courses*, p. 204. ACM, 2005.

[49] B. Shi, S. Bai, Z. Zhou, and X. Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

[50] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pp. 223–240. Springer, 2016.

[51] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proc. CVPR*, 2017.

[52] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[53] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. IEEE ICCV*, pp. 945–953, 2015.

[54] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM TOG*, 22(4):1012–1033, Oct. 2003.

[55] T.-T. Tran, V.-T. Cao, S. Ali, D. Laurendeau, et al. Automatic method for sharp feature extraction from 3d data of man-made objects. In *Computer Graphics Theory and Applications (GRAPP), 2014 International Conference on*, pp. 1–8. IEEE, 2014.

[56] C. Weber, S. Hahmann, and H. Hagen. Sharp feature detection in point clouds. In *Shape Modeling International Conference (SMI), 2010*, pp. 175–186. IEEE, 2010.

[57] C. Weber, S. Hahmann, H. Hagen, and G.-P. Bonneau. Sharp feature preserving mls surface reconstruction based on local feature line approximations. *Graph. Models*, 74(6):335–345, 2012.

[58] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015.

[59] L. Yi, H. Su, X. Guo, and L. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. *arXiv:1612.00606*, 2016.

[60] Y. Zheng, H. Fu, O. K.-C. Au, and C.-L. Tai. Bilateral normal filtering for mesh denoising. *IEEE TVCG*, 17(10):1521–1530, 2011.