

Reconstruct Sharp Features from Intensity-Derived Scan Angles

Optional Subtitle of the Thesis

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Branislav Balvan

Registration Number 12023159

to the Faculty of Informatics

at the TU Wien

Advisor: Mag. Stefan Ohrhallinger, PhD

Vienna, July 23, 2026

Branislav Balvan

Stefan Ohrhallinger

Erklärung zur Verfassung der Arbeit

Branislav Balvan

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 23. Juli 2026

Branislav Balvan

Acknowledgements

I would like to thank my family for their continuous support during my studies and the writing of this thesis, as well as my supervisor, Stefan Ohrhallinger, for his guidance throughout the project and all the feedback he provided.

Abstract

Surface reconstruction algorithms such as Poisson reconstruction favor global smoothness and tend to lose the sharp edges and corners that characterize man-made and CAD objects. This thesis investigates whether scan angles derived from LiDAR backscatter intensity can be used to improve the reconstruction of such sharp features. We present a pipeline that estimates point normals using Principal Component Analysis, refines them in a second pass using the recorded intensity under a Lambertian reflectance model, reconstructs a surface mesh, and deforms that mesh toward detected sharp-feature chains to recover edges and corners lost during reconstruction. The approach is evaluated on a set of models from the Thingi10K, ABC datasets and on custom-authored CAD shapes, using a simulated LiDAR scan to obtain input point clouds together with a ground-truth mesh for comparison. The intensity-based adjustment reduces the angular error of the estimated normals on every model, by an average of 10.6%, with the largest gains on smooth organic surfaces where the method’s assumptions hold best. For the final reconstructed surface, the deformation visibly restores sharp features and reduces the worst-case reconstruction error on the models where such features dominate, although this improvement is only partially captured by the global Hausdorff distance. The results show that intensity-derived scan angles are useful and inexpensive processing-wise for improving normal estimation, with the benefit concentrated precisely where standard estimation is weakest.

Contents

Abstract	vii
Contents	ix
1 Introduction	1
2 Related work	3
3 Method	7
3.1 Normal Estimation and Adjustment	9
3.2 Initial Surface Reconstruction	12
3.3 Feature Detection and Classification	13
3.4 Feature Clustering and Chain Extraction	17
3.5 Feature Chain Fitting and Projection	21
3.6 Feature-Constrained Mesh Deformation	23
4 Results	27
4.1 Intensity-Based Normals Adjustment	29
4.2 Feature-Constrained Mesh Deformation	30
4.3 Discussion	34
4.4 Limitations	36
5 Conclusion	39
5.1 Future work	40
Overview of Generative AI Tools Used	41
List of Figures	43
List of Tables	45
List of Algorithms	47
Bibliography	49

Introduction

Identifying sharp features in scanned surfaces is a key step for accurate reconstruction. Reconstruction algorithms prioritize global surface continuity over the preservation of local details, which results in over-smoothing of the surface and the loss of some sharp details in the process. Preserving sharp details, such as edges or corners, is necessary in the reverse engineering of computer-aided design (CAD) models, which are often piecewise smooth and designed with many sharp features for assembly or functional requirements. The loss of these sharp features can degrade both the aesthetic and functional integrity of the reconstructed models. An aesthetic and faithful duplicate of a real-world object is also important for the purpose of digital cultural heritage preservation, which also makes use of photogrammetry with LiDAR sensors. This technology is not only valuable for potential reconstruction in the event of damage but also enables the creation of a digital twin—a precise digital replica that monitors critical environmental factors such as temperature, humidity, and other conditions affecting the stability of cultural artifacts. By using digital twins to predict conditions, conservators and curators can take proactive measures, ensuring the long-term preservation of cultural heritage [ECR⁺26, NF24].

Surface reconstruction with preservation of sharp features poses a challenge because many reconstruction algorithms (Poisson, B-spline, NURBS) implicitly create a smooth and continuous surface from the input point cloud data, leading to the loss of sharp edges and corners. There are already existing approaches using a neural network trained on 3D data with labeled sharp features [FSYL23, YLF⁺18, WXX⁺20], but instead of relying on machine learning for sharp feature detection, our goal was to develop a detection method based on a systematic computational approach [MOG11, HWG⁺13]. We presume that a computational approach can better identify sharp feature types that might not have been included in a training dataset and also save the time needed to adequately label shapes for training.

LiDAR sensors provide high-resolution 3D data in the form of point clouds but do not contain any information about point normal vectors, so it is necessary to estimate them.

We first estimate the normal vector of each point using Principal Component Analysis in relation to the known viewpoint position (scanner position) of each scan. PCA estimation is accurate on smooth surfaces but tends to over-smooth normals near sharp edges and corners, where the local neighborhood spans more than one surface patch. To address this, we use a second signal recorded by the scanner: the backscatter intensity. The LiDAR system emits laser beams toward the target and records the returned pulse after it is back-scattered from the surface. Under the Lambertian reflection model, this backscatter intensity is proportional to the cosine of the incidence angle between the surface and the scan direction. Real-world surfaces are made of varying materials with complex reflection properties and rarely behave like an ideal Lambertian surface, so it is not feasible to rely on the incidence angle alone. Instead, we use it in a second pass that adjusts the PCA-estimated normals based on the recorded intensity and the average orientation of neighboring points, improving normal accuracy particularly in the regions where PCA estimation is weakest.

Building on these refined normals, we reconstruct a surface mesh and deform it toward detected sharp-feature chains in order to recover the edges and corners that the initial reconstruction smooths out. We evaluate the approach on a set of models from the Thingi10K [ZJ16], ABC datasets [KMJ⁺19] and on custom-authored CAD shapes, using a simulated LiDAR scan [GKU11] to obtain input point clouds together with a ground-truth mesh for comparison.

Related work

LiDAR Backscatter Intensity and the Lambertian Assumption

The input data for our pipeline comes from LiDAR laser scans of either whole scenes or individual objects. Beyond recording the distance at which the laser intersects a surface, the scanner also measures the ratio of light reflected back to the sensor — the backscatter intensity [Jel92]. The relationship between the returned intensity and the incidence angle — the angle between the incoming laser beam and the local surface normal — can be described by Lambert’s cosine law. Pfeifer et al. [PDHF07] showed that this model holds well for terrestrial laser scanners, and in a follow-up study [PHB⁺08] demonstrated that target reflectivity can be recovered from intensity data with a standard deviation of about 5–6%. The Lambertian model is especially accurate for diffuse materials, where incoming light is scattered equally in all directions regardless of the viewing angle — examples include concrete, brick, and wood. However, for surfaces with higher specularity, the Lambertian assumption becomes less reliable because a significant portion of the reflected light is concentrated around the specular direction rather than being scattered uniformly. Tian et al. [TTC⁺21] proposed a combined Lambertian-Beckmann model that accounts for both diffuse and specular reflection components, improving calibration accuracy by an average of about 23% across their test samples. Other alternative models discussed in the literature include the Oren-Nayar model and the Phong model. Since our pipeline targets outdoor urban data, scanning predominantly matte, rough industrial or architectural surfaces, the simplified Lambertian assumption is a reasonable choice. A further limitation worth noting is that our intensity-based normal adjustment assumes constant surface albedo across each local point neighborhood. At material boundaries where reflectance changes abruptly, intensity variations caused by albedo differences can be misinterpreted as incidence angle changes, potentially degrading the adjusted normals [TTC⁺21].

Initial Surface Reconstruction from Point Cloud

For the initial mesh generation from the input point cloud we employ Poisson Surface Reconstruction [KBH06], which solves a global Poisson equation to recover a watertight surface from oriented point samples. Kazhdan & Hoppe [KH13] later introduced the improved Screened Poisson Reconstruction, which incorporates positional point constraints to better preserve fine detail. However, since CGAL provides a robust implementation of the original reconstruction algorithm that suffices for our initialization purpose, we use that variant. An alternative class of methods are Moving Least Squares (MLS) surfaces [Lev04, ABCO⁺03], which fit local polynomial approximations to the point cloud but do not inherently guarantee a watertight output, making them less suitable as a deformation base.

Sharp Feature Detection and Classification

Detecting sharp features in unstructured 3D data has been addressed by a wide range of approaches. Learning-based methods train neural networks directly on the task: PIE-Net [WXX⁺20] detects edge and corner points and fits parametric curves to them, while EC-Net [YLF⁺18] learns to identify sharp features and densify the point cloud along them. These methods require annotated training data, typically from CAD datasets such as ABC [KMJ⁺19], and their generalization is bounded by the feature geometries encountered during the training phase.

Alternatively, computational approaches analyze local point neighborhood eigenstructure for high surface variation [PKG03] or try to detect the sharp features from the distribution of surface normals in a local neighborhood [KBSS01]. The latter paper is the one we also built our detection stage upon (Section 3.3), and although the main goal of that paper is an extension of the marching cubes algorithm, their sharp features detection method is directly applicable to our problem context with a points cloud as an input. Kobbelt et al. detect and classify sharp features from the distribution of surface normals in a local neighborhood, using two computationally inexpensive geometric tests: the opening angle of the normal cone (θ -test) to detect sharpness, and the deviation from the best-fit plane through the normals (ϕ -test) to distinguish edges from corners. This distinction between edge and corner simplifies the subsequent stage of our pipeline, where feature points are clustered and chained into feature curves: corner points naturally delimit the endpoints of edge chains, providing a structural anchor for the chaining process. Additionally, their tangent-plane intersection scheme, solved via SVD, provides a principled way to refine noisy feature positions, which improves the accuracy of further processing stages [KBSS01].

Geometric Primitive Fitting and Extracting

Fitting geometric primitives such as lines and planes to unstructured 3D data is a problem made difficult by the presence of noise and outliers inherent to real-world LiDAR scans. The RANSAC algorithm [FB81] addresses this by repeatedly sampling minimal point subsets, fitting a candidate model, and selecting the model supported by the largest inlier

set. Its tolerance to a high outlier ratio makes it useful for primitive fitting in point-cloud processing, and implementations for various geometries are available in libraries such as PCL [Poi25c]. A known limitation of RANSAC is that a single run detects only the dominant model, which is insufficient when the data contains several instances of the same primitive type, for example multiple coplanar line segments in a shared neighborhood. Multi-model extensions address this by recovering successive models from the same point set. The simplest such extension, Sequential RANSAC, fits the dominant model, removes its inliers, and repeats on the set of remaining points. There exist multiple multi-model fitting frameworks based on RANSAC as described by Fouhey et al. [Fou13], but the Sequential RANSAC is concluded as being one of the most accurate approaches, while also staying computationally reasonable.

Sharp-Feature-Preserving Mesh Reconstruction

Several pipelines have been proposed that take a point cloud as input and produce a triangle mesh in which sharp edges and corners are preserved, rather than smoothed out by an implicit reconstruction step such as Poisson surface reconstruction [KH13]. These methods differ primarily in how they detect sharp features and how they couple the detection step to the meshing step. Two representative pipelines, one based on traditional geometric analysis and one based on supervised learning, illustrate the design space.

Voronoi-based feature detection with feature-preserving Delaunay refinement.

Salman et al. [SYM10] propose a pipeline that extracts sharp features and generates a surface mesh in parallel. They identify feature points using the Voronoi-covariance measure (VCM) [MOG11], trace them into a feature graph, and reconstruct a smooth implicit surface via Poisson reconstruction. Finally, feature-preserving Delaunay refinement [BO05, CDR07] is used to generate a watertight mesh that explicitly contains the extracted feature polylines as edges.

Compared to our approach, the pipeline proposed by Salman et al. shares the high-level structure: extracting features, reconstructing a smooth surface, and refining it, but they differ in two key aspects. First, we detect features using point normals refined by LiDAR backscatter intensity rather than relying solely on raw positions, which mitigates normal-estimation errors near edges. Second, instead of re-meshing the entire surface via Delaunay refinement, we deform an initial Poisson reconstruction toward fitted feature chains. This preserves the sharp features while avoiding the need to re-mesh smooth regions.

Learning-based primitive detection with mesh splitting and selection.

On the other hand, Liu et al. [LXXW23] use a supervised PointNeXt-based neural network [QLP⁺22] to segment point clouds into geometric primitives. These primitives are meshed independently and intersected, using global optimization to assemble a watertight, lightweight mesh with sharp inter-primitive boundaries.

While this yields high-quality reconstructions for CAD models, it differs from our setting again in two distinct ways. First, it relies on supervised training [KMJ⁺19] and generalizes poorly outside of CAD-style distributions. Second, it assumes the object can be decomposed into standard primitives, limiting its application on organic or non-standard shapes. In contrast, our pipeline is entirely unsupervised and makes no primitive assumptions, but the trade-off is that we do not explicitly recover analytical surface representations for smooth regions.

Position in the design space. These two methods bracket the design choices for sharp-feature reconstruction: Salman et al. rely entirely on geometric position analysis, whereas Liu et al. depend on supervised learning. Our work sits between these extremes. We adopt a geometric, unsupervised approach for feature detection and meshing [SYM10], but supplement the standard position-only data with LiDAR backscatter intensity. This is a readily available signal, produced by LiDAR scans, that has not previously been exploited for sharp-feature reconstruction.

Method

This chapter serves as an overview of the computational pipeline for reconstructing surfaces with sharp features from unorganized point cloud scans. The stages are described in the same order as the execution order of the pipeline, from the initial set of point cloud scans to reconstructed mesh as final output. Each section is one stage of the pipeline, with its most important processing steps described as subsections and a list of configurable parameters introduced at the beginning of each section, that allow the user to tweak thresholds and other computational parameters to make the reconstruction and detection of the sharp features more tailored to a specific dataset with better results.

Overview of the Pipeline

The pipeline is composed of 6 main stages: (1) Normal estimation and refinement using LiDAR intensity values, (2) Generation of the initial smooth mesh with Poisson surface reconstruction, (3) Sharp feature detection and classification, (4) Clustering of the detected sharp features and extracting feature chains, (5) Fitting of the feature chains to the best selected model (*line*, *curve*, *circle*) and (6) Mesh deformation adapting to the detected sharp features.

Input Data. The input to our system is a set of one or multiple point clouds $\mathcal{P}_i = \{p_1, \dots, p_N\}$ obtained from LiDAR scans, also called *cloudviews* internally. The scanned geometry should be watertight, so that an initial mesh can be constructed by Poisson surface reconstruction properly. In case of a set of multiple point clouds, it is assumed that the point clouds are at least partially overlapping and each has a 3D position of the scanner's viewpoint $\mathbf{v}_{pos}$ from where the scan have been taken.

Output Data. The output is a reconstructed triangle mesh $\mathcal{M}$ where edges and corners are aligned with the detected sharp features. The processing steps are illustrated in Figure 3.1.

3. METHOD

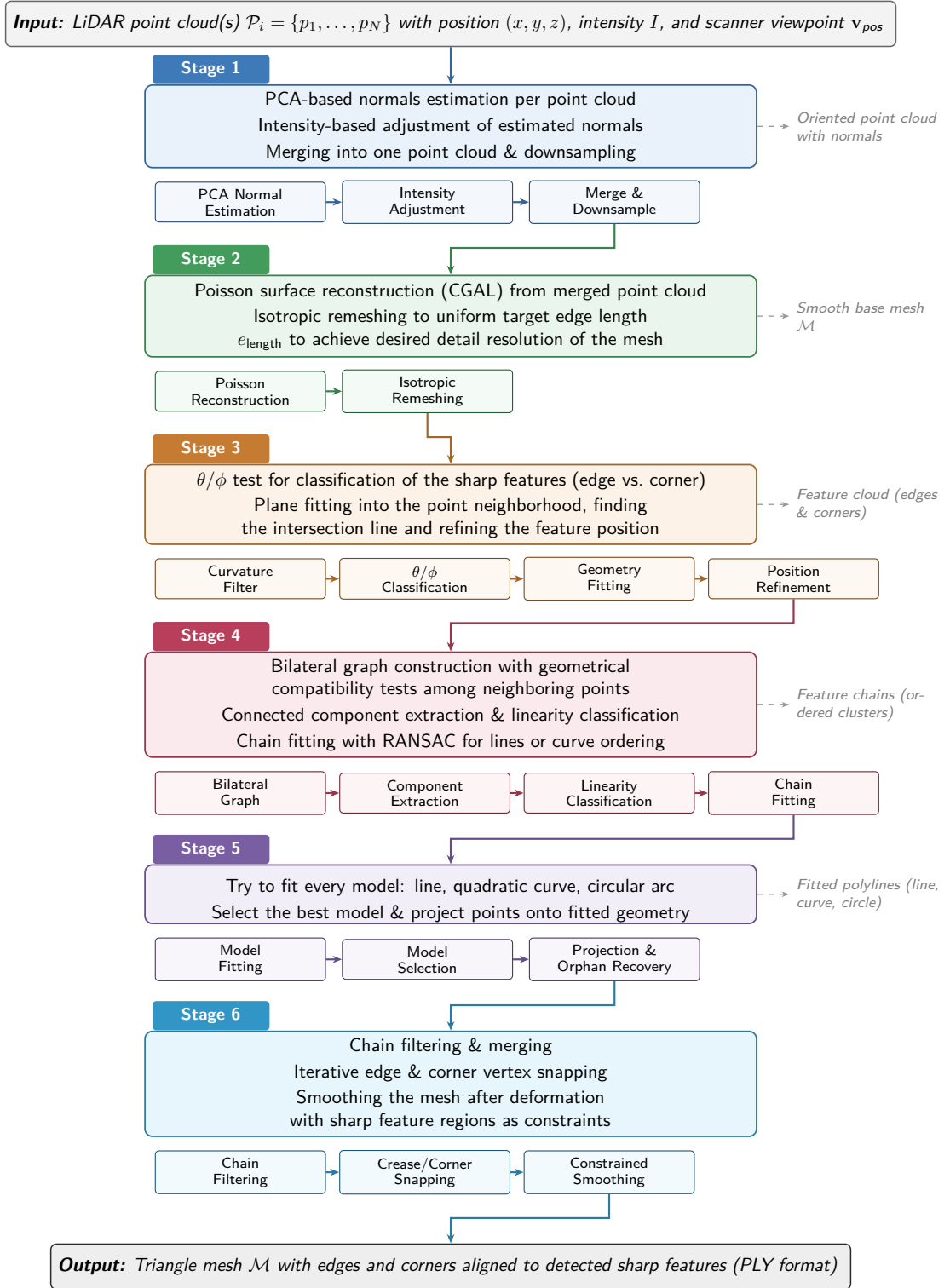


Figure 3.1: Overview of the whole computational pipeline.

3.1 Normal Estimation and Adjustment

The goal of the first stage of the pipeline is to calculate normals for each point of the input point cloud $\mathcal{P} = \{p_1, \dots, p_N\}$. Existing surface normals are an important prerequisite for both the surface reconstruction process as well as the detection of sharp features, but the point cloud obtained from the LiDAR scans contains only 3D coordinates (x, y, z) and backscatter intensity values I for each point. After the initial normal estimation, we run an adjustment pass on the estimated normals to further improve the estimation. Additionally, each point cloud in the set must carry the position of the scanner’s viewpoint $\mathbf{v}_{pos}$ relative to the point cloud origin, as required for the initial normal estimation (especially with multiple scans in the set).

3.1.1 Configurable Parameters

Parameter	Default Value	Description
k_{adj}	20	Neighbourhood size for intensity-based adjustment
L_{pre}	0.01	Voxel edge length for optional pre-merge downsampling
L_{post}	0.01	Voxel edge length for optional post-merge downsampling

Table 3.1: Normal estimation and adjustment parameters.

3.1.2 Geometric Estimation (PCA)

The input data are provided mainly as scans of the same object from different viewpoints. A normal estimated by PCA over a local neighborhood is only defined up to sign, so a reference is needed to orient all normals consistently; in this case the sensor viewpoint of the scan. Since this viewpoint is well-defined only per individual scan and is lost once all scans are merged into a single cloud, it makes sense to estimate the normals for each *cloudview* separately with its viewpoint included and only merge the clouds afterwards. We use PCL’s `NormalEstimation` class [Poi25b], which internally performs the Principal Component Analysis over local neighborhoods to estimate surface normals. For every point p_i and its k_{est} nearest neighbors a covariance matrix gets constructed, with $k_{est} = k_{adj}$:

$$\mathbf{C} = \frac{1}{k_{est}} \sum_{j \in \mathcal{N}_i} (p_j - \bar{p})(p_j - \bar{p})^\top \quad (3.1)$$

where $\bar{p}$ is the geometric center of the neighborhood $\mathcal{N}_i$. We get the estimated normal direction $\mathbf{n}_{PCA}$ from the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$.

The normals estimated by PCA can have ambiguous orientation (they are flipped). This is the reason why the scanner viewpoint $\mathbf{v}_{pos}$ is necessary, because we can pass it to the estimator, which then orients each normal towards the given scanner viewpoint position[Poi25a].

3.1.3 Intensity-Based Adjustment

The PCA estimation produces more accurate normals on smooth, mostly uniform surfaces and tends to produce over-smoothed normals on surface regions with sharp edges and corners. The reason is that the k_{adj} -nearest neighborhood contains two or more significantly different surface patches, causing the estimated normal to represent an average of these instead of the true surface direction at the queried point (which would be identical with one of the surface patches). Each point in the point cloud has a backscatter intensity recorded from the scanner. When we establish the relation between the intensity value and the angle, under which a ray coming from the scanner view position is falling on the surface plane with the point, we can further adjust the normals after the initial estimation with PCA.

Using the Backscatter Intensity

To improve the accuracy of the estimation, we adjust the PCA normals by using the LiDAR return intensity. Following the Lambertian reflectance model (see relevant Chapter 2), we assume the recorded backscatter intensity I is proportional to the cosine of the incidence angle between the surface normal and the (scanner) view direction [PHB⁺08]:

$$I \propto \rho (\mathbf{n} \cdot \hat{\mathbf{v}}) \quad (3.2)$$

where $\hat{\mathbf{v}}$ is the unit vector from the point towards the scanner viewpoint and ρ is the local surface albedo value, assumed as a constant for the whole point neighborhood for simplification.

This assumption makes the adjustment reliable for objects of a single material. However, at the boundary between two surfaces of differing reflectance, a change in I can stem either from the differing albedo or from the surface angle, and the model cannot separate these two causes. Because ρ is assumed constant, our method attributes the entire intensity change to the incidence angle and consequently adjusts the normal as if the surface had tilted, producing an inaccurately oriented normal at such boundaries. The difference in reflection across different materials is also demonstrated by Tian et al. [TTC⁺21].

Neighborhood Statistics. For each point p_i we take its k_{adj} nearest neighbors $\mathcal{N}_i = \{p_j\}$ and compute the unit view direction $\hat{\mathbf{v}}_j = (\mathbf{v}_{\text{pos}} - p_j) / \|\mathbf{v}_{\text{pos}} - p_j\|$ and the geometric alignment $\mathbf{n}_j \cdot \hat{\mathbf{v}}_j$ (flipping $\mathbf{n}_j$ if it faces away from the scanner). The mean intensity $\bar{I}$ and the mean cosine $\overline{\cos \theta}$ are summed like this:

$$\overline{\cos \theta} = \frac{1}{k_{\text{adj}}} \sum_{j \in \mathcal{N}_i} (\mathbf{n}_j \cdot \hat{\mathbf{v}}_j) \quad (3.3)$$

$$\bar{I} = \frac{1}{k_{\text{adj}}} \sum_{j \in \mathcal{N}_i} \quad (3.4)$$

Target Angle Prediction. Because we assume the local surface albedo value ρ is constant, the ratio of intensity to incidence cosine should be similar for the whole point neighborhood. We estimate the target cosine for point p_i as:

$$\alpha_{\text{target}} = I_i \cdot \frac{\overline{\cos \theta}}{\bar{I}} \quad (3.5)$$

clamped between 0 and 1. This value represents the incidence angle the normal should have, given the point’s observed intensity relative to its neighbors.

Normal Rotation. The original PCA normal $\mathbf{n}_{\text{PCA}}$ is decomposed into 2 components - parallel and perpendicular to the view direction $\hat{\mathbf{v}}_i$:

$$\mathbf{n}_{\perp} = \mathbf{n}_{\text{PCA}} - (\mathbf{n}_{\text{PCA}} \cdot \hat{\mathbf{v}}_i) \hat{\mathbf{v}}_i \quad (3.6)$$

A refined unit normal is created, so it achieves the desired incidence cosine α_{target} to match the observed intensity while preserving the in-plane orientation estimated by PCA:

$$\mathbf{n}_{\text{ref}} = \alpha_{\text{target}} \hat{\mathbf{v}}_i + \sqrt{1 - \alpha_{\text{target}}^2} \frac{\mathbf{n}_{\perp}}{\|\mathbf{n}_{\perp}\|} \quad (3.7)$$

In case that the point cloud contains points with either invalid incident direction (point is the same as viewpoint), with n_{PCA} that has zero length or accumulated neighborhood intensity $\bar{I}$ zero, these points are left unadjusted.

3.1.4 Merging and Downsampling

After normal estimation and adjustment, the individual cloudview point clouds are joined into a single merged cloud. Because overlapping scan regions introduce redundant points and non-uniform density across the point cloud, a voxel grid downsampling step provided by PCL’s `VoxelGrid` can optionally be applied [Poi25e]. We can downsample already before the merge to improve the performance of the whole estimation and adjustment step, or afterwards on the merged cloud. The voxel filter subdivides the space with the point cloud(s) into voxels with configurable edge length (L_{pre} resp. L_{post}) and replaces all points within each voxel by their geometric center with an averaged normal.

3.2 Initial Surface Reconstruction

After obtaining a merged point cloud with estimated and intensity-adjusted normals, the next step is to reconstruct a continuous, watertight surface mesh that will serve as the base geometry for the final deformation stage. We use the Poisson surface reconstruction algorithm [KBH06], as implemented in the Computational Geometry Algorithms Library (CGAL) [The25a].

3.2.1 Configurable Parameters

For the initial mesh reconstruction stage the only configurable property is the target edge length of the reconstructed mesh (used during the isotropic remeshing):

Parameter	Default Value	Description
e_{length}	0.01	Target edge length after isotropic remeshing

Table 3.2: Surface reconstruction parameters.

3.2.2 Poisson Surface Reconstruction

We use Poisson surface reconstruction [KBH06] to obtain a watertight mesh from the oriented point cloud. The core idea is to treat the input normals as samples of a gradient field and recover a scalar indicator function $\tilde{\chi}$ whose gradient best matches the input normals. This amounts to solving the Poisson equation:

$$\Delta \tilde{\chi} = \nabla \cdot \vec{V} \quad (3.8)$$

where $\vec{V}$ is the vector field built from the oriented samples and Δ is the Laplacian. The reconstructed surface corresponds to the zero-level set of $\tilde{\chi}$. We rely on CGAL’s Poisson reconstruction pipeline, which extracts the iso-surface via adaptive marching cubes. We enable the `manifold_with_boundary` constraint during mesh generation to guarantee manifold output with consistently oriented normals.

3.2.3 Mesh Quality and Remeshing

The mesh generation is controlled by three quality parameters scaled by the average point spacing $\bar{d}$ (computed from the 6 nearest neighbors of each point), a minimum facet angle of 20° to prevent invalid triangles, a maximum facet size of $10 \cdot \bar{d}$, and a maximum surface approximation distance of $0.15 \cdot \bar{d}$. After reconstruction, the mesh is uniformly remeshed using CGAL’s isotropic remeshing [The25b] to achieve a consistent target edge length e_{length} , thus being more suited for the subsequent deformation.

3.2.4 Motivation for Subsequent Deformation

Poisson reconstruction produces a smoothed approximation of the given geometry (in our case the provided point cloud). Because the algorithm solves a global Laplacian equation, fine geometric detail like sharp edges and corners are attenuated in the resulting reconstructed surface. This smooth mesh therefore serves just as a clean starting point that must be further refined in the deformation stage (Section 3.6) to recreate a mesh topologically more similar to the scanned object.

3.3 Feature Detection and Classification

After the base mesh has been created, we return back to the (merged) point cloud where the pipeline must identify the sharp edges and corners that were smoothed away during Poisson reconstruction. Rather than relying on learned approaches, we use the geometric detection strategy proposed by Kobbelt et al. [KBSS01], that not only detects if there is a sharp feature, but it also classifies them into edge and corner features. The final output of this pipeline stage is a set of points with refined positions and classification type (edge, corner) that mark the surface regions with sharp features.

3.3.1 Configurable Parameters

The feature detection stage uses the following parameters with default values based on empirical evaluation and the recommended values from Kobbelt et al. paper [KBSS01]:

Parameter	Default Value	Description
<i>Classification</i>		
k_{test}	15	Neighborhood size for θ/ϕ tests
κ_{min}	0.01	Minimum curvature for pre-filtering
θ_{sharp}	70°	Normal cone opening angle threshold
ϕ_{corner}	0.7	Out-of-plane deviation threshold for corners
<i>Geometry Fitting</i>		
d_{RANSAC}	0.05	RANSAC inlier distance for plane fitting
d_{max}	0.1	Maximum distance from a point to the feature edge
p_{cluster}	0.25	Minimum neighborhood cluster partition ratio
<i>Position Refinement</i>		
k_{refine}	10	Neighbors count for tangent plane collection

Table 3.3: Feature detection and refinement parameters.

3.3.2 Curvature-Based Pre-Filtering

The normal estimation stage in (Section 3.1) provides besides surface normals also an estimated curvature κ_i for each point. Since the vast majority of points lie on smooth surfaces, we apply a basic filter that immediately rejects any point with $\kappa_i < \kappa_{\text{min}}$ (set

by default to 0.01). This simple step eliminates flat parts of the point cloud from further processing, reducing the computational cost of the following geometric tests.

3.3.3 Geometric Classification via θ/ϕ Tests

For each remaining point p_i , we retrieve its k_{test} nearest neighbors and collect their normal vectors $\{\mathbf{n}_j\}$. The normals estimated in (Section 3.1) might still have inconsistent orientation, which could affect the further detection process, so they need to be aligned to a common hemisphere by flipping any normal that has negative dot product with the reference direction [HDD⁺92]. We then classify the local surface with following two tests from Kobbelt et al [KBSS01]:

θ -Test for Sharp Feature Detection. The opening angle of the local normal cone is measured by computing the minimum pairwise dot product among all aligned neighborhood normals:

$$\theta_{\cos} = \min_{a,b} (\tilde{\mathbf{n}}_a \cdot \tilde{\mathbf{n}}_b) \quad (3.9)$$

If $\theta_{\cos} \geq \theta_{\text{sharp}}$ (where $\theta_{\text{sharp}} = \cos(70^\circ) \approx 0.342$), the normals are sufficiently consistent and the point is classified as smooth. A value below this threshold indicates that two or more distinct surface orientations are present in the neighborhood, marking the point as a sharp feature candidate.

ϕ -Test for Edge vs. Corner Discrimination. All points that pass the θ -test are sharp features, but we need to distinguish for each if it's an edge (2 surfaces meeting) or a corner (3 and more surfaces meeting). Using PCA on the set of aligned normals and take the eigenvector corresponding to the smallest eigenvalue, $\mathbf{n}^*$, which is the normal to the best-fit plane through the normal vectors. The ϕ -test then measures the maximum deviation:

$$\phi_{\cos} = \max_j |\tilde{\mathbf{n}}_j \cdot \mathbf{n}^*| \quad (3.10)$$

If all normals are close to a single plane the $\phi_{\cos}$ is small and the point is classified as an edge, but in case the threshold is exceeded $\phi_{\cos} \geq \phi_{\text{corner}}$ it is reasonable to assume three or more surfaces are meeting at this point, so it should be classified as a corner. We can see the result of this classification in Figure 3.2.

3.3.4 Geometry Fitting

The classification tests identify where the sharp features are, but we also need the geometry fitting step that determines the 2 planes at whose intersection each edge feature point is located.

Plane Fitting and Intersection for Edge Features If a point is classified as an edge, the neighborhood normals are partitioned into two clusters C_1 and C_2 using K-means clustering [Llo82], corresponding to the two faces meeting at the edge. We check if the partition of the clusters is reasonably balanced in the neighborhood:

$$\frac{\min(|C_1|, |C_2|)}{\max(|C_1|, |C_2|)} \geq p_{\text{cluster}} \quad (3.11)$$

A plane $\Pi_i : \mathbf{n}_i \cdot \mathbf{x} + d_i = 0$ is then fitted to each cluster using RANSAC (with inlier threshold $d_{\text{RANSAC}} = 0.05$) with the PCL's `RandomSampleConsensus` [Poi25c]. The intersection line of the two fitted planes defines the local edge direction:

$$\ell : \mathbf{x}(t) = \mathbf{p}_0 + t \cdot (\mathbf{n}_1 \times \mathbf{n}_2) \quad (3.12)$$

The candidate is accepted as an edge feature only if the distance from p_i to ℓ is below a threshold d_{max} . The fitted plane normals $\mathbf{n}_1, \mathbf{n}_2$ and the edge tangent $\mathbf{t} = \mathbf{n}_1 \times \mathbf{n}_2$ are stored as geometric descriptors of the feature.

Corner Features. At corners, where three or more surfaces meet, reliable multi-plane fitting becomes impractical. Rather than attempting to recover all incident planes, we store the corner as an corner feature (`END_FEATURE`) carrying the two most divergent normals in the neighborhood, i.e. the pair achieving the minimum dot product in the θ -test. These two normals are not intended to fully characterize the corner geometry, they only serve as representative directions that anchor the corner in the same data layout as edge features (which carry two plane normals by definition) and provide enough information for the further chain extraction and deformation stages, where corners only need to act as pinned constraints, they are not included in the sharp feature chain extraction and fitting.

3.3.5 Position Refinement

When an object is scanned by a LiDAR scanner, usually some amount of noise is introduced so the original point positions p_i do not lie exactly on the true sharp features. With the help of Kobbelt et al.'s tangent plane intersection approach, we can compute refined positions p'_i for the points that approximate the actual feature geometry better [KBSS01].

Edge Point Refinement. We get the refined position for each edge point if we find a point that minimizes the total squared distance to a set of tangent plane constraints $\{(\mathbf{N}_i, \mathbf{P}_i)\}$:

$$p' = \arg \min_{\mathbf{x}} \sum_i (\mathbf{N}_i \cdot (\mathbf{x} - \mathbf{P}_i))^2 \quad (3.13)$$

The constraint set consists of the two tangent planes at p_i and then additional planes from nearby feature points. For each neighbor p_j that is also an edge, we pick whichever of its two plane normals best aligns with the $\mathbf{n}_1$ and $\mathbf{n}_2$ of p_i .

We solve the minimization problem via SVD, but with the smallest singular value zeroed out to enforce a rank-2 constraint, otherwise the solution would collapse to a single point in space (as opposed to staying as a line) [KBSS01].

Corner Point Refinement. For corner feature points we use the same SVD-based formulation as in the edge case but drop the rank-2 constraint, since a corner is the intersection of three or more surfaces rather than two. The system collects plane equations from the corner’s own two stored normals, anchored at its position, together with both plane normals of every neighboring feature in the corner’s local neighborhood, each anchored at that neighbor’s position. The SVD solution is the point that best satisfies all collected plane constraints simultaneously, locating the corner where multiple surfaces meet.

3.3.6 Feature Cloud Construction

For the next steps in the pipeline we only need to work with the sharp feature points and don’t need the whole point cloud anymore, so we copy only the features in a new point cloud called *feature cloud*. Each feature point holds its type (edge or corner), position p'_i , the fitted plane normals $\mathbf{n}_1, \mathbf{n}_2$, the edge tangent direction $\mathbf{t}$, and a reference back to its index in the original point cloud.

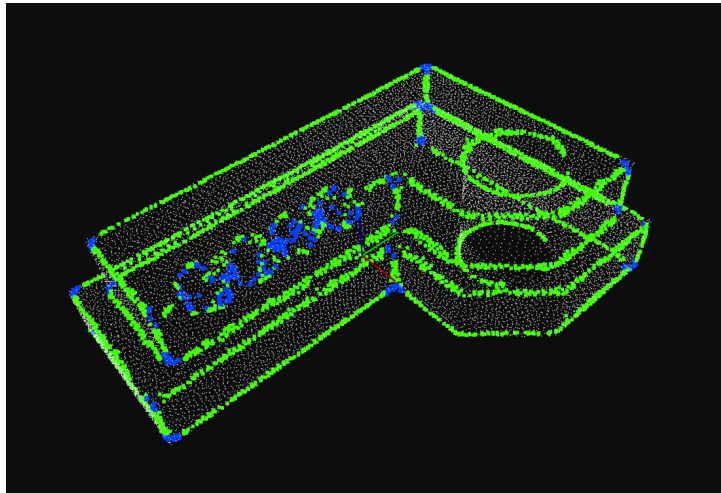


Figure 3.2: Detection of sharp features, classified either as edge feature (green) or corner (blue).

3.4 Feature Clustering and Chain Extraction

The previous stage produces an unstructured set of feature points. To be useful for mesh deformation, these points must be organized into continuous geometric structures representing individual sharp edges. This stage groups spatially connected feature points into *feature chains* using a connectivity graph, then fits appropriate geometric primitives to each chain.

3.4.1 Configurable Parameters

The clustering stage uses the following parameters:

Parameter	Default Value	Description
<i>Graph Construction</i>		
k_{cand}	15	KNN search size
r	0.1	Search radius
τ_{tan}	0.95	Tangent alignment threshold
τ_{dir}	0.80	Bilateral directional consistency threshold
τ_{norm}	0.90	Normal consistency threshold
σ_{min}	0.5	Minimum composite edge score for second-pass acceptance
<i>Linearity Classification</i>		
τ_{res}	0.08	Maximum non-primary variance ratio
τ_{lin}	15.0	Minimum dominant eigenvalue ratio
<i>Chain Extraction</i>		
τ_{merge}	0.996	Minimum parallelism $ \mathbf{t}_1 \cdot \mathbf{t}_2 $ for segment merging
Circle RMS	0.05	Maximum residual for circle acceptance

Table 3.4: Feature clustering and chain extraction parameters.

3.4.2 Bilateral Graph Construction

The Parallel Edge Problem

In previous experiments naive proximity-based grouping failed when two parallel edges lie close together: points on one edge may be nearer to points on the adjacent edge than to their own continuation, causing distinct features to merge into a single cluster. To prevent this, we use a **bilateral connectivity criterion** that requires mutual geometric alignment between any two connected points.

Geometric Compatibility Tests

For each edge feature point p_i , the k_{cand} nearest neighbors from the set of detected features within a search radius r are retrieved. A candidate connection to neighbor p_j must pass four tests:

Tangent Alignment. The computed tangent vectors $\mathbf{t}_i$ and $\mathbf{t}_j$ for sharp edge feature points p_i, p_j must be nearly parallel:

$$|\mathbf{t}_i \cdot \mathbf{t}_j| \geq \tau_{\text{tan}} \quad (3.14)$$

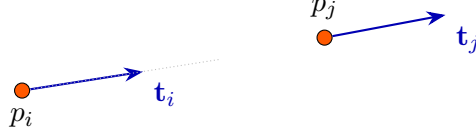


Figure 3.3: The tangents $\mathbf{t}_i$ and $\mathbf{t}_j$ at the two edge feature points must be nearly parallel, i.e. their absolute dot product must exceed τ_{tan} .

Bilateral Directional Consistency. The connection direction $\mathbf{d}_{ij} = (p_j - p_i) / \|p_j - p_i\|$ must align with the tangent at *both* sharp edge feature points:

$$|\mathbf{d}_{ij} \cdot \mathbf{t}_i| \geq \tau_{\text{dir}} \quad \text{and} \quad |\mathbf{d}_{ij} \cdot \mathbf{t}_j| \geq \tau_{\text{dir}} \quad (3.15)$$

With this bilateral check a point on edge E_A looking perpendicularly toward edge E_B will fail this test because the connection direction is orthogonal to both edges tangents, thus keeping parallel edges separated.

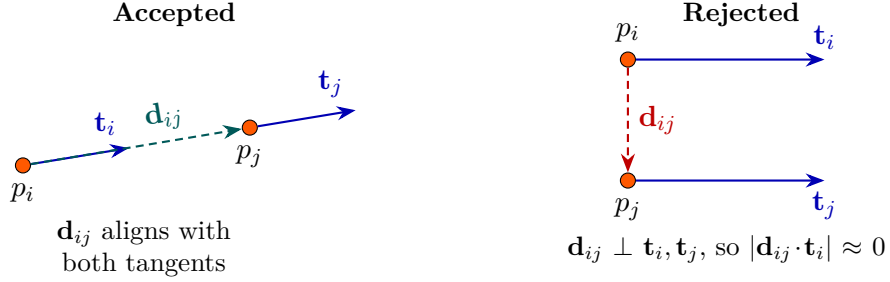


Figure 3.4: Left: a valid connection along a single edge, where the connection direction $\mathbf{d}_{ij}$ aligns with both tangents. Right: two parallel edges facing each other, where $\mathbf{d}_{ij}$ is orthogonal to both tangents and the test $|\mathbf{d}_{ij} \cdot \mathbf{t}_i| \geq \tau_{\text{dir}}$ and $|\mathbf{d}_{ij} \cdot \mathbf{t}_j| \geq \tau_{\text{dir}}$ fails, keeping the edges separate.

Normal Consistency. The plane normals $\{\mathbf{n}_1^{(i)}, \mathbf{n}_2^{(i)}\}$ and $\{\mathbf{n}_1^{(j)}, \mathbf{n}_2^{(j)}\}$ must correspond to compatible surface orientations. For each normal of p_i we take its best match against either normal of p_j , and require that both of p_i 's normals are matched, i.e.

$$\min_{k \in \{1,2\}} \max_{l \in \{1,2\}} |\mathbf{n}_k^{(i)} \cdot \mathbf{n}_l^{(j)}| \geq \tau_{\text{norm}}. \quad (3.16)$$

The absolute value makes the test invariant to normal orientation, comparing the underlying planes rather than their signed directions.

Graph Construction and Component Extraction

Connections that pass all tests are scored based on tangent and directional alignment (the higher the better) at short distances:

$$w(i, j) = \frac{|\mathbf{t}_i \cdot \mathbf{t}_j| \cdot |\mathbf{d}_{ij} \cdot \mathbf{t}_i| \cdot |\mathbf{d}_{ij} \cdot \mathbf{t}_j|}{\|p_j - p_i\| + \epsilon} \quad (3.17)$$

The best forward and best backward neighbors (determined by the sign of $\mathbf{d}_{ij} \cdot \mathbf{t}_i$) are identified for each point. An edge is added in the first pass if at least one of its two endpoints regards the other as its best neighbor in either direction. A second pass then runs over the remaining candidate edges and adds an edge when a candidate exists in both directions and its composite score s_{ij} exceeds $\sigma_{\min}$, where s_{ij} combines tangent alignment, directional alignment, and an inverse-distance term. Connected components are then extracted, each forming a candidate cluster.

3.4.3 Linearity Classification

Each cluster needs to be classified either as linear or non-linear to choose the appropriate fitting strategy. PCA is applied to the positions of the cluster's point, yielding eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ [WJHM15]. We then compute:

$$\text{Residual Fraction} = \frac{\lambda_2 + \lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \quad (3.18)$$

$$\text{Strength Ratio} = \frac{\lambda_1}{\lambda_2 + \lambda_3} \quad (3.19)$$

If the residual fraction $\tau_{\text{res}} < 0.08$ and the strength ratio τ_{lin} exceeds 15.0 a cluster is classified as linear (these parameter values can be configured).

3.4.4 Chain Extraction

Linear Clusters

One linear cluster may contain multiple disjoint line segments, which should actually be separate sharp features. We can extract individual segments using a sequential application of RANSAC [FB81, Fou13], where for each iteration, a line model is fitted via RANSAC with an adaptive inlier threshold ϵ derived from the cluster's median point spacing and overall extent, then a starting point (either end of the cluster along the fitted direction) is grown in both directions along the adjacency graph, accepting neighbors that lie close to the line and whose connection direction aligns with the line tangent. Each grown segment is refit using PCA and its points are removed from the remaining points. This repeats until almost no remaining points are left.

Lastly, overlapping or nearby collinear segments produced by this process are considered for merging if they satisfy strict parallelism ($|\mathbf{t}_1 \cdot \mathbf{t}_2| \geq \tau_{\text{merge}}$). The close-distance criterion depends on the segment's projected extents along $\mathbf{t}_1$. Disjoint segments are merged only if the longitudinal gap between them is below a fixed tolerance *and* their perpendicular offset lies within the adaptive RANSAC threshold ε . Overlapping segments are merged on the perpendicular offset alone, using a stricter bound of $\varepsilon/2$. The combined segment is then refit by PCA and its points reordered by projection along the new line direction.

Non-Linear Clusters

For each cluster classified as non-linear, the cluster is marked as a closed loop if more than 80% of points have exactly two internal neighbors. Open curves are ordered by finding the diameter path (longest shortest path) of the cluster's Minimum Spanning Tree, while loops are ordered by traversing the adjacency cycle. The chain is provisionally labeled as a curved feature, but the definitive model selection (circle or quadratic spline) is left to the following stage of the pipeline (Section 3.5) to decide.

3.4.5 Output: Feature Chains

The output is a collection of feature chains, each containing: a list of feature points, a preliminary geometric type (line for linear clusters, (looped) curve for non-linear clusters), fitted line parameters where possible and an unique cluster identifier. To ensure the sequence of points in a feature chain is ordered, each feature point knows its position within the chain.

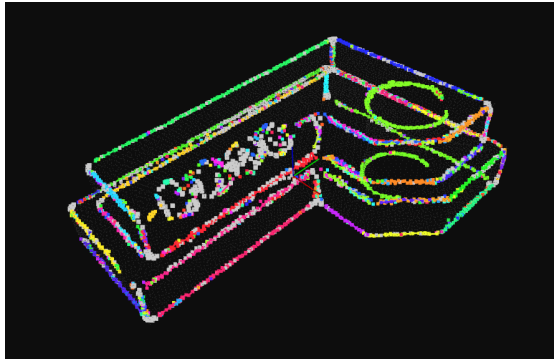


Figure 3.5: Clustered feature edges. Each color represents a distinct feature chain.

3.5 Feature Chain Fitting and Projection

The clustering stage produces ordered feature chains with preliminary geometric type, but the definitive fitting of the chain to a geometric model is performed by this stage: for each chain, three candidate geometric models are evaluated, the best is selected, and all member points are projected onto the chosen curve. The result is a set of noise-free feature representations where every point lies exactly on its fitted geometry.

3.5.1 Configurable Parameters

The feature fitting stage uses the following parameters:

Parameter	Default Value	Description
<i>Model Selection</i>		
Line preference	1.30	Error tolerance multiplier favouring linear fits
Circle improvement	0.80	Required error ratio to accept a circle over alternatives
$\kappa_{\min}$	0.2	Minimum curvature to accept a quadratic fit
<i>Circle Constraints</i>		
$\theta_{\text{span}}^{\text{open}}$	1.2 rad	Minimum angular span for open-chain circle acceptance
$\theta_{\text{span}}^{\text{loop}}$	5.0 rad	Minimum angular span for loop circle acceptance
<i>Orphan Processing</i>		
d_{snap}	0.05	Maximum distance for orphan snapping

Table 3.5: Feature fitting and projection parameters.

3.5.2 Candidate Model Fitting

For each feature chain, three geometric models are fitted to the ordered chain points $\{p_i\}_{i=1}^N$:

Line. A line with center $\mathbf{C}$ and direction $\mathbf{t}$, obtained from PCA on the chain points. Then if we take the sum of the perpendicular distance from the line to each point and compute the root mean square, we can estimate how well the line fits:

$$e_{\text{line}} = \sqrt{\frac{1}{N} \sum_{i=1}^N \|(p_i - \mathbf{C}) - [(p_i - \mathbf{C}) \cdot \mathbf{t}] \mathbf{t}\|^2} \quad (3.20)$$

Quadratic Curve. The coefficients $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$ are fitted separately per coordinate axis by least squares via SVD. To decide whether the chain has meaningful curvature, we evaluate a curvature indicator at the midpoint of the normalized parameter domain. For a general parametric curve the curvature is

$$\kappa(s) = \frac{\|\mathbf{r}'(s) \times \mathbf{r}''(s)\|}{\|\mathbf{r}'(s)\|^3}, \quad (3.21)$$

where the cross product extracts the component of $\mathbf{r}''$ perpendicular to the tangent $\mathbf{r}'$ and the factor $\|\mathbf{r}'\|^3$ removes the dependence on parametrization speed. Since s parametrizes the curve by normalized chord length rather than true arc length, κ serves as a curvature indicator for the thresholded *line* versus *curve* decision rather than as a metrically exact geometric curvature. For the quadratic $\mathbf{r}(s) = \mathbf{a}s^2 + \mathbf{b}s + \mathbf{c}$ the derivatives are $\mathbf{r}'(s) = 2\mathbf{a}s + \mathbf{b}$ and $\mathbf{r}''(s) = 2\mathbf{a}$, the latter being constant. The indicator is obtained by evaluating κ at the parameter midpoint.

Circular Arc. A circle with center $\mathbf{C}$, plane normal $\mathbf{n}$, and radius R is fitted using PCL’s `SampleConsensusModelCircle3D` [Poi25d]. The angular span θ_{span} is computed to distinguish partial arcs from full circles.

3.5.3 Model Selection

We select out of the three presented models the one that aligns with the points of the chain the most. The selection logic is shown in Table 3.6, where each rule is evaluated top to bottom and the first matching rule determines the final model selection.

Priority	Model	Condition
1	Circle	Angular span θ_{span} exceeds threshold and $e_{\text{circle}} < 0.80 \cdot e_{\text{best}}$
2	Quadratic	Curvature $\kappa > \kappa_{\text{min}}$ and $e_{\text{quad}} < 0.95 \cdot e_{\text{line}}$
3	Line	$e_{\text{line}} \leq 1.30 \cdot e_{\text{best}}$
4	Best fit	Lowest RMS error among all candidates

Table 3.6: Model selection priority.

3.5.4 Projection and Reordering

With the model selected, every point of the chain is projected onto it’s new fitted geometry. We use orthogonal projection for lines, radial projection in the fitted plane for circles, and Newton-Raphson parameter optimization to fit quadratic splines (minimizing $\|\mathbf{r}(s) - p\|^2$) [Far02]. The projected positions replace the refined positions from the detection stage, ensuring that all feature points lie exactly on the fitted geometry. After the projection, chains are re-sorted again depending on the geometry type, but in this case we decide the ordering of each point by its projection parameter for lines, angle for circles, arc-length parameter for quadratic curves.

3.5.5 Orphan Recovery

Orphan feature points that were excluded during clustering are re-added by finding the nearest fitted chain within the same cluster and projecting the orphan onto it. If the projection distance falls within a maximum snap distance d_{snap} , the point is added to

the chain and the chain is re-sorted. This final step improves the completeness of the feature representation without introducing weird connections.

3.6 Feature-Constrained Mesh Deformation

The Poisson reconstruction produces a smooth, watertight mesh that lacks the sharp edges that might have been present in the original scan data. The final pipeline stage restores these features by deforming the mesh based on the detected and fitted feature chains from the previous stages. The mesh is deformed by combining crease edge sharpening with corner vertex snapping, followed by a smoothing pass to remove possible artifacts in surface regions without detected sharp features.

3.6.1 Configurable Parameters

The mesh deformation stage uses the following parameters:

Parameter	Default Value	Description
<i>Chain Filtering</i>		
d_{merge}	2.0	Perpendicular distance factor for chain merging
c_{min}	5	Minimum points to retain a chain
l_{min}	3.0	Minimum chain length as multiple of point spacing
<i>Deformation</i>		
τ_{crease}	1.5	Distance factor for crease edge identification
α_{snap}	0.9	Initial snap strength
i_{def}	5	Number of deformation iterations
<i>Smoothing</i>		
i_{smooth}	10	Angle-based smoothing iterations
λ	0–0.001	Mean curvature flow strength (optional)

Table 3.7: Mesh deformation and smoothing parameters.

3.6.2 Preparation: Chain Filtering and Merging

Before deformation, the feature chains are converted into simplified polylines. For better results we filter and remove chains that are too short (fewer than c_{min} or total arc length below $3 \times s_{\text{median}}$, where s_{median} is the median point spacing). Circular chains are also polylines, but more dense to reflect the circular shape.

The clustering stage may produce duplicate chains tracing the same geometrical edge, so another filtering and merging step is needed to identify and remove the redundant chains. Two open chains are merged when their overall directions are nearly parallel and their average perpendicular distance is small (below d_{merge}). For loop chains, merging is based on proximity of their respective centers. In all cases longer chain is always favored and retained, while the others get discarded. Last safety measure to minimize the possible

inaccuracies in the reconstruction, caused by wrongly fitted feature chains, if a chain has poor directional alignment with its associated mesh edges, it also gets discarded.

3.6.3 Crease Edge Identification

For each feature chain we identify edges on the mesh that should belong to it. For each mesh edge e connecting vertices v_0 and v_1 , the edge midpoint $\mathbf{m} = (\mathbf{v}_0 + \mathbf{v}_1)/2$ is projected onto every chain polyline. The projection finds the closest point on any segment of the chain:

$$\mathbf{p}_{\text{proj}} = \mathbf{a}_i + t_i(\mathbf{b}_i - \mathbf{a}_i), \quad t_i = \text{clamp}\left(\frac{(\mathbf{m} - \mathbf{a}_i) \cdot (\mathbf{b}_i - \mathbf{a}_i)}{\|\mathbf{b}_i - \mathbf{a}_i\|^2}, 0, 1\right) \quad (3.22)$$

If the distance between an edge and the chain is below $\tau \cdot s_{\text{median}}$ (with $\tau = 1.5$), the edge is marked as a crease edge associated with the chain, and target positions for both endpoints v_0 and v_1 are projected individually onto that chain.

3.6.4 Corner Vertex Identification

Corner features points (as classified during detection stage in Section 3.3) are handled differently from edge chains. For each corner feature with a valid refined position, the nearest mesh vertex is found via a KD-tree search. If the distance is within a factor of the local average edge length, the vertex is locked as a corner vertex with the corner feature’s refined position as its target. When multiple corner features compete for the same mesh vertex, the closest one is picked.

3.6.5 Iterative Vertex Snapping

The deformation proceeds over i_{def} iterations, with a displacement strength decreasing over iterations:

$$\alpha_k = \alpha_{\text{snap}} \left(1 - 0.5 \frac{k}{i_{\text{def}}}\right) \quad (3.23)$$

where $\alpha_{\text{snap}} = 0.9$ as default. At each iteration, crease edges and corner vertices are recomputed based on the current mesh geometry to track the gradually deformed surface.

Crease Vertex Displacement. For each crease vertex v , the target position $\mathbf{q}_v$ is determined by majority vote: the chain contributing the most crease edges incident to v defines the target, computed as the average of its per-edge projections onto the chain polyline. This avoids conflicting displacements when multiple chains compete for the same vertex. The displacement $\mathbf{d}_v = \mathbf{q}_v - \mathbf{x}_v$ is clamped to half the local average edge length $\bar{\ell}_v$ to prevent excessive single-step deformation, giving $\tilde{\mathbf{d}}_v = \mathbf{d}_v \cdot \min(1, 0.5 \bar{\ell}_v / \|\mathbf{d}_v\|)$. The clamped displacement is then scaled by the iteration strength and applied, $\mathbf{x}'_v = \mathbf{x}_v + \alpha_k \tilde{\mathbf{d}}_v$.

Corner Vertex Displacement. Corner vertices are snapped toward their pinned target positions using the same strength schedule and displacement clamping.

3.6.6 Selective Post-Deformation Smoothing

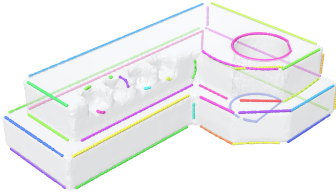
In order to improve the visual appearance of the resulting mesh, we can optionally apply a smoothing pass, but with all the already deformed sharp edges and corners marked as immovable constraints:

1. **Angle-based smoothing:** CGAL's `angle_and_area_smoothing` [The25b]
2. **Mean curvature flow** (optional): CGAL's `smooth_shape` [The25b] - ideally with a small setting ($\lambda \leq 0.001$), or else the smoothing will be excessively aggressive.

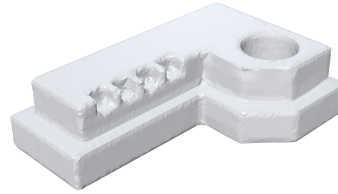
Using the sharp features as constraints ensures that the deformation that has been done will not be undone during smoothing and the reconstructed geometric details will be preserved.

3.6.7 Output

The final output is a triangle mesh in PLY format.



(a) Feature chains overlaid on the reconstructed surface mesh.



(b) Final deformed mesh with reconstructed sharp features.

Figure 3.6: Results after the deformation stage.

Results

In this chapter we first show how the estimation of the normals changes when we make use of the discussed intensity-based adjustment as compared to the normals on the ground truth model. Secondly, we show how does the subsequent deformation differ with or without the prior normals adjustment and also comparison to the Poisson surface reconstructed mesh alone.

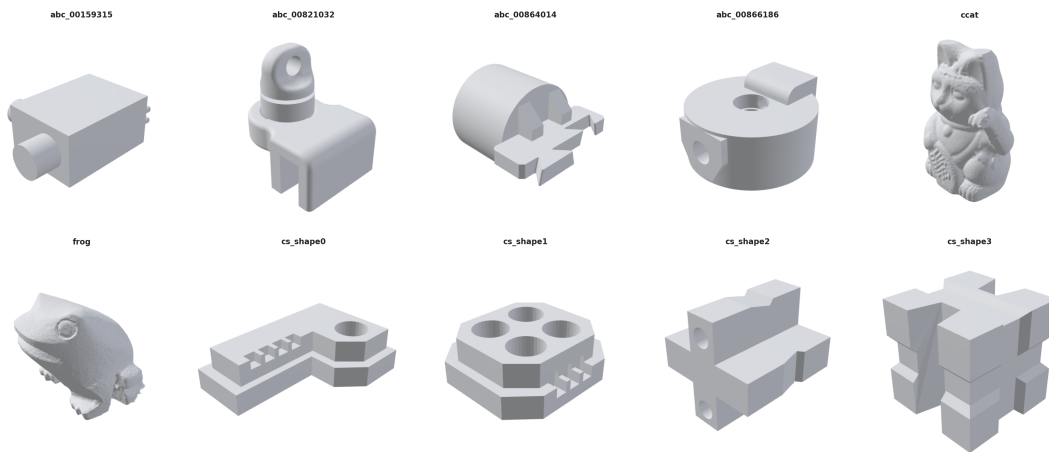


Figure 4.1: 3D objects used for evaluation.

In order to measure the accuracy of the deformation we need to supply a ground truth object for comparison with the resulting object and their normals. With help of a Blender plugin [GKU11] we have simulated a LiDAR scanner and scanned a virtual 3D object from multiple viewpoints to better mimic real-life LiDAR scene data. The selected test samples are 2 objects from Thingi10K-Dataset [ZJ16] with more smooth and organic features, 4 CAD models from ABC-Dataset [KMJ⁺19] and 4 custom shapes that also contain higher count of sharp features, similar to the ABC-Dataset.

Usage with Real-World Scan

For demonstration purposes we used a scan of an urban scene in Vienna (Mozartplatz) made with a LiDAR scanner as an input for our pipeline.¹ Because there is no ground-truth for the real-world scan, we cannot measure the accuracy of the reconstruction, but we showcase the resulting detected sharp features and the points on mesh where there is a significant difference between the base mesh and the resulting deformed mesh for both adjusted and non-adjusted estimated normals.

It is worth mentioning that Poisson reconstruction that we are using in our pipeline works optimally on watertight objects and this is an outdoor scene, so we opted to use only a cutout of the fountain and its surroundings from the scene in order to have a suitable input point cloud. The merged scan consists of 6 scans, each from different viewpoint.



Figure 4.2: Detected sharp features on reconstructed mesh.

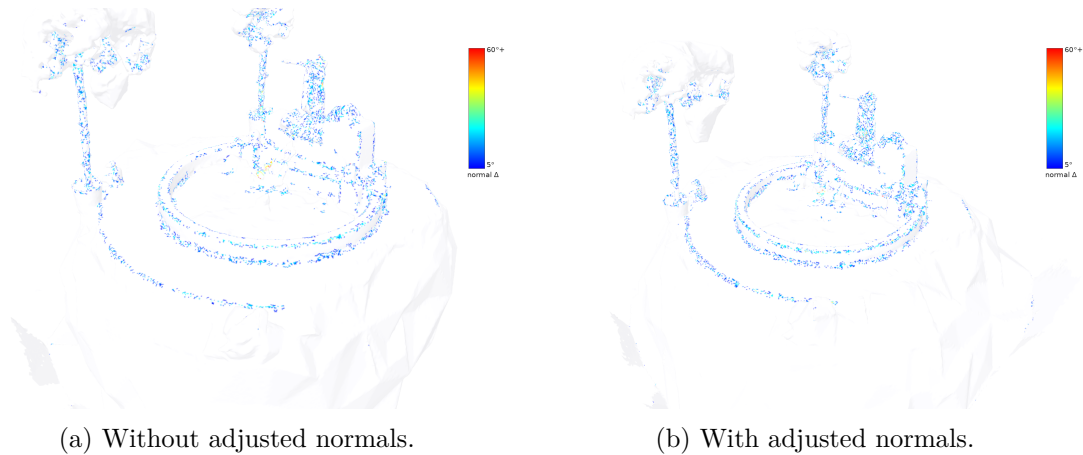


Figure 4.3: Normals orientation difference on reconstructed mesh.

¹Scan data courtesy of TU Wien, Research Unit of Computer Graphics.

4.1 Intensity-Based Normals Adjustment

To measure the accuracy of the normal estimation from Section 3.1, we take each point cloud point p_i with estimated normal n_i and find the closest point q_i on the surface of the ground-truth mesh. We then take the normal vector m_i of the face to which q_i belongs to and calculate the unsigned angular error θ_i between n_i and m_i . We obtain the root-mean-square error over all valid points:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N \theta_i^2}$$

The difference between the normals is computed twice, before and after the final downsampling of the merged cloud, to see how the downsampling influences the resulting adjustment. To obtain the following results we used $k_{adj} = 20$. Compared to $k_{adj} = 10$, this value gave a small but consistent improvement on the more organic models from Thingi10K, in the range of 0.1-0.2° RMSE, while increasing the runtime of the adjustment step by less than 2% (measured on an AMD Ryzen 5 7600X). The improvements for each model are reported below in absolute degrees and as a relative percentage.

Model	Pre-downsample RMSE [°]				Post-downsample RMSE [°]			
	No adj.	Adj.	Δ	Rel. [%]	No adj.	Adj.	Δ	Rel. [%]
abc_00159315	10.74	9.85	0.89	8.3	11.26	10.57	0.70	6.2
abc_00821032	10.50	9.45	1.05	10.0	11.10	10.29	0.81	7.3
abc_00864014	10.48	9.38	1.11	10.5	10.29	9.19	1.10	10.7
abc_00866186	9.62	8.60	1.02	10.6	10.79	9.79	0.99	9.2
ccat	9.04	7.37	1.67	18.5	8.01	6.51	1.50	18.7
frog	9.57	7.95	1.62	16.9	9.31	7.69	1.62	17.4
cshape0	16.12	15.38	0.74	4.6	15.99	15.32	0.67	4.2
cshape1	11.45	10.04	1.41	12.3	11.73	10.68	1.05	8.9
cshape2	11.77	10.73	1.03	8.8	11.68	10.62	1.06	9.1
cshape3	11.14	10.01	1.13	10.2	10.75	9.52	1.23	11.5
<i>Mean</i>	11.04	9.88	1.17	10.6	11.09	10.02	1.07	9.7

Table 4.1: Normal estimation accuracy with and without intensity-based adjustment, using $k_{adj} = 20$.

4.1.1 Interpretation of Results

The intensity-based adjustment of normals reduces RMSE on every model, with a mean improvement of 1.17° (10.6%) pre-downsample and 1.07° (9.7%) post-downsample. The relative improvement is consistently larger on the organic models (*frog* and *ccat*, 17–19%) than on the objects with more sharp features (4–12%). Combined with downsampling the estimation accuracy changes depending on the geometry. The downsampling

further reduces the RMSE on organic models and our custom models with more smooth features, but increases or doesn't change it for CAD models with higher density of sharp features.

4.2 Feature-Constrained Mesh Deformation

For the measurement of the accuracy of the reconstructed mesh we use Hausdorff distance, computed by sampling 100,000 face points on the reconstruction and querying their closest point on the ground-truth mesh. We do not include the Hausdorff distance in reverse direction (sampling on the ground truth) because ground-truth meshes from the ABC-dataset contain internal geometry that is not externally visible and therefore not captured by the simulated LiDAR scan that produces our input point clouds. This would penalize the results of the final deformation for missing surfaces that were never part of the input data.

We compare three variants for each model: the Poisson reconstruction before deformation (*Initial*), the final deformed output with intensity-based normal adjustment enabled (*Deformed*), and the final deformed output with adjustment disabled (*No adj.*). All values are reported relative to the bounding-box diagonal of the ground-truth mesh, in percent.

Model	Mean distance [% diag]			Max distance [% diag]		
	Initial	Deformed	No adj.	Initial	Deformed	No adj.
abc_00159315	0.044	0.050	0.050	1.180	1.200	1.284
abc_00821032	0.156	0.172	0.161	4.873	4.877	4.865
abc_00864014	0.036	0.038	0.039	0.688	0.711	0.703
abc_00866186	0.098	0.104	0.103	4.593	4.612	4.680
ccat	0.046	0.048	0.049	0.415	0.404	0.480
frog	0.038	0.042	0.040	0.475	0.537	0.490
cshape0	0.064	0.082	0.081	2.107	2.132	2.047
cshape1	0.140	0.147	0.149	3.290	3.298	3.354
cshape2	0.134	0.144	0.139	2.976	2.955	2.942
cshape3	0.068	0.072	0.073	0.854	0.726	0.729
<i>Mean</i>	0.083	0.090	0.088	2.145	2.145	2.158

Table 4.2: Surface accuracy of the reconstructed meshes measured by the Hausdorff distance from the reconstruction to the ground truth. Bold entries mark cases where a deformed variant improves on the initial Poisson reconstruction.

4.2.1 Sharp-Feature-Constrained Hausdorff Distance

The feature-constrained mesh deformation, is designed to recover specifically the sharp edges and corners that the Poisson reconstruction tends to smooth out. To evaluate this contribution directly, we additionally compute a sharp-feature-restricted variant of

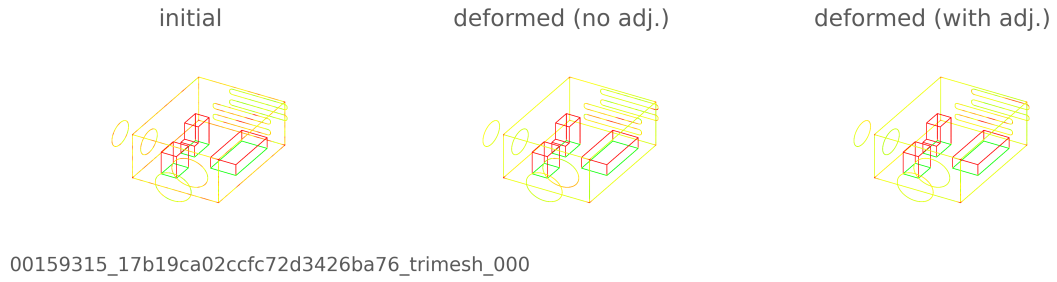
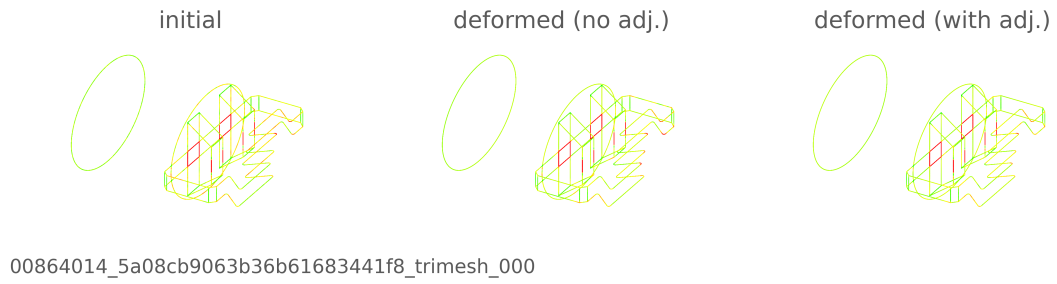
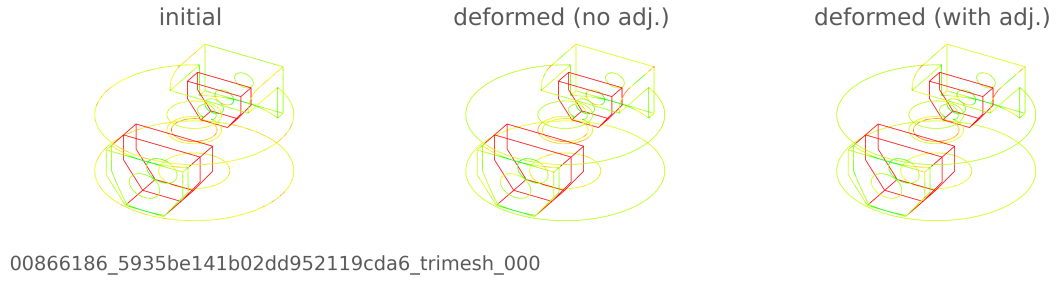
the Hausdorff metric. For each ground-truth mesh, we mark every edge whose adjacent face normals deviate by at least 40° as a *sharp* edge. We then sample 100,000 points uniformly along the union of these sharp edges, weighted by edge length so that longer features are represented proportionally. The same sample set is reused across the three variants so the comparison is paired. Distances are computed and normalized in the same way as for the global metric.

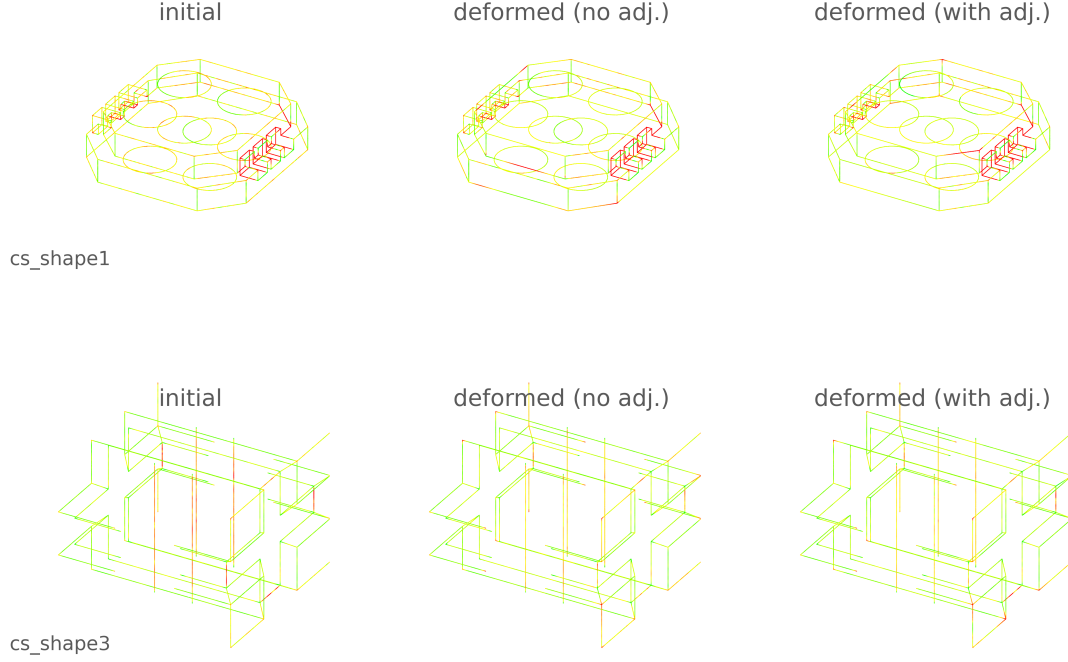
Model	Mean distance [% diag]			Max distance [% diag]			n_{edges}
	Initial	Deformed	No adj.	Initial	Deformed	No adj.	
abc_00159315	1.884	1.834	1.838	13.368	13.368	13.364	6354
abc_00821032	0.957	0.971	0.965	6.935	6.935	6.961	2544
abc_00864014	0.543	0.507	0.516	3.995	4.027	4.079	3079
abc_00866186	1.138	1.114	1.120	9.324	9.323	9.338	4384
ccat	0.255	0.260	0.260	2.988	2.988	2.938	1904
frog	0.305	0.309	0.306	1.214	1.269	1.276	907
cshape0	0.518	0.523	0.521	1.359	1.297	1.827	163
cshape1	0.733	0.717	0.716	6.699	6.699	6.728	432
cshape2	0.524	0.535	0.476	1.618	1.618	1.638	22
cshape3	0.391	0.386	0.395	1.100	1.100	1.067	93
<i>Mean</i>	0.725	0.716	0.711	4.860	4.862	4.922	

Table 4.3: Sharp-feature-restricted Hausdorff distance. Bold entries mark cases where a deformed variant improves on the initial Poisson reconstruction.

4.2.2 Visualization of Accuracy Error

Here we present visualization of the Sharp-Feature-Constrained Hausdorff distance measurement results, in each row there is the sharp edge accuracy error of the initial Poisson reconstructed mesh, then deformed mesh without and deformed mesh with intensity-based normals adjustment. Green color means minimal error, red stands for significant difference.





4.2.3 Interpretation of Results

The mean distance from the reconstruction to the ground truth increases slightly after deformation on every model, by 3.8-27.5% relative to the initial Poisson reconstruction. In absolute terms this corresponds to changes of 0.002-0.018% of the bounding-box diagonal, all below 0.02%. The maximum distance stays unchanged on average, but on three models the deformation reduces the worst-case error noticeably: most clearly on `cs_shape3` (-15.0%), but also on `cs_shape2` (-0.7%) and `ccat` (-2.7%).

The sharp-feature-restricted metric (Table 4.3) reveals a pattern that the global metric does not. On 6 out of 8 models whose ground truth is dominated by sharp CAD-style features, the deformation reduces the mean sharp-edge distance below the initial Poisson baseline. The mean across the test set drops from 0.725% (initial) to 0.716% (deformed) and 0.711% (no adjustment) of the bounding-box diagonal.

Differences between the deformed and no-adjustment variants remain small under both metrics, with no consistent winner across the test set.

It should be also noted, that because the models from the ABC-Dataset contain internal geometry that is not possible to reconstruct from outer scans, these show up as completely inaccurate (red colored) in the Accuracy Error Visualization 4.2.2.

4.3 Discussion

Our goal for this thesis is to investigate if scan angles derived from LiDAR backscatter intensity can be used to improve the reconstruction of sharp features in surface meshes. The obtained results demonstrate a measurable improvement at the level of estimated point normals (Section 4.1). At the level of the final reconstructed surface (Section 4.2), the deformation improves visibly the geometry of sharp features like edges and circular openings that the Poisson reconstruction smooths out. This is reflected in the sharp-feature-restricted Hausdorff metric, where the deformation reduces the mean error below the initial Poisson baseline on 6 of the 8 models dominated by sharp features. In contrast to that the global Hausdorff distance shows only a small change after deformation, because it is dominated by samples on smooth regions of the surface. In this section we discuss why the improvement that the normal adjustment method brings is not equal across the whole test sample, why the two Hausdorff variants show different results, and why the improvement in normals does not translate into a consistent mesh-level improvement when comparing the adjusted and unadjusted variants of the deformation.

4.3.1 Improvement with Normal Adjustment is Geometry-Dependent

The target incidence cosine for a point’s normal is recovered from the ratio of mean neighborhood cosine to mean neighborhood intensity (Equation 3.7), which is more accurate if the points in the neighborhood lie on surface patches with similar orientation, which is the case especially with the smooth models. On CAD models with a high density of sharp features, the same neighborhood frequently spans multiple surface patches across edges and corners, so the mean neighborhood cosine $\overline{\cos\theta}$ mixes point normals from differently orientated surfaces and the recovered target angle no longer corresponds to the true incidence cosine at the queried point. The adjustment still improves over the PCA baseline because most points in the neighborhood typically lie on the correct side, but less of its potential is realized. This explains the systematic pattern observed in Section 4.1: every model benefits from the adjustment, but organic models gain noticeably more than those dominated by sharp features.

The subsequent voxel-grid downsampling follows a similar logic. On models whose smooth patches are large relative to the voxel size, voxel averaging further attenuates residual estimation noise and reduces RMSE. On models with features finer than the voxel, the same averaging blurs across geometric discontinuities and partially offsets the adjustment, which is consistent with the observation that downsampling either fails to help or slightly worsens RMSE on the sharper CAD models.

4.3.2 Measuring Reconstruction Accuracy through The Hausdorff Metric

The global Hausdorff distance from the reconstruction to the ground truth increases on the mean after deformation across all ten models relative to the initial Poisson reconstruction. Taken in isolation, this looks like the deformation worsens reconstruction accuracy, but

we believe this is the case because the mean distance is computed over 100000 face samples drawn uniformly across the mesh surface, and the overwhelming majority of those samples lie on regions between the sharp features. The deformation moves vertices in two distinct ways: it pulls vertices near detected features toward the fitted sharp feature chain shape and it propagates that displacement through neighboring smooth vertices via the underlying deformation field (an unavoidable side effect). The global metric is dominated by the second effect because smooth-region samples dominate the average and the first effect, which is the one the deformation is designed for, is averaged out.

The sharp-feature-restricted variant of the metric isolates the first effect by sampling only along ground-truth edges with an inter-face angle of at least 40° . On this metric the result is qualitatively different. Of the eight models whose ground truth is dominated by sharp CAD-style features (`cs_shape0`-`cs_shape3` and the four ABC parts), at least one deformed variant reduces the mean sharp-edge distance below the initial baseline on six. The two exceptions, `cs_shape0` and `abc_00821032`, register small increases of around 1%. On the two organic models (`frog`, `ccat`), both deformed variants slightly worsen the metric, by 0.5-1.5% of the initial. The improvement on the six sharp-feature wins is consistent with the design of the deformation and the lack of improvement on the organic models is consistent with the absence of real sharp features to snap to, leaving only the side effect of vertex displacement. The maximum global Hausdorff distance shows the same pattern in a less pronounced form: it stays unchanged on average, but on three models the deformation reduces the worst-case error noticeably - most clearly on `cs_shape3` (-15.0%). Together the two metrics support a consistent interpretation: the deformation trades a small increase in mean accuracy on smooth regions for a reduction in error along sharp features, where it is designed to act.

4.3.3 Adjustment vs. No-Adjustment Deformation

The differences between the deformed variants with and without normal adjustment are consistently small and inconsistent in sign as reported by the Hausdorff metrics. On the mean distance, the two variants differ by less than 0.015% of the bounding-box diagonal on every model; on the maximum, neither variant wins on more than half the test set. This stands in sharp contrast to the adjustment improving the estimated normals on every model by a measurable amount in Table 4.1. On the sharp-feature mean, the spread between the two variants is similarly narrow, with the adjusted variant winning on 4 of the 8 models dominated by sharp features.

This probably means that even when we improve the input data for the feature detection, the 10-20% improvement in the normals estimation accuracy only helps the subset of cases where the unadjusted normals failed to cross the detection threshold by a small margin. Feature detection is a discrete operation, because a point either qualifies as a sharp feature or it does not, and a feature chain is either grouped correctly or it is not. A continuous improvement in normal accuracy therefore only changes the outcome for points that lie close to a decision threshold, and the effect is not strictly beneficial. An

adjusted normal can push a point across the threshold so that it is now correctly detected, but it can equally pull a previously detected point back below the threshold, removing it from a feature it should have belonged to. Across a whole model these two effects partly cancel, which is consistent with the small and sign-inconsistent differences we observe.

4.4 Limitations

The test set comprises ten models taken from three different sources, which is sufficient to identify the geometry-dependent patterns reported above but probably too small to support any quantitative claims.

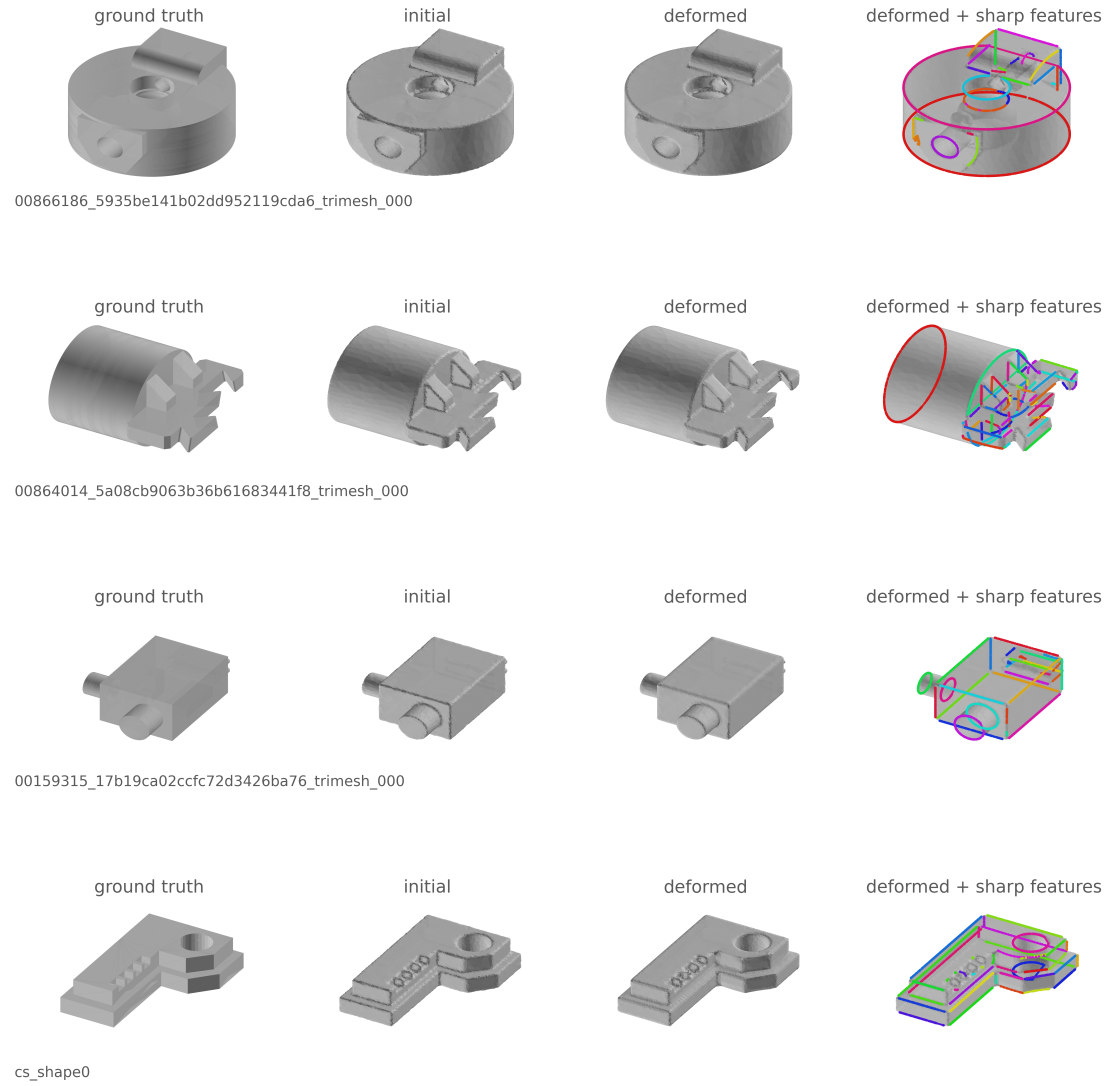
The intensity-derived adjustment assumes the local surface albedo ρ is approximately constant across each k_{adj} neighborhood. All test models are single-material and the assumption holds throughout. On models or scenes that mix surface materials, a change in intensity at a material boundary is indistinguishable from a change in surface orientation under the Lambertian reflectance model [TTC⁺21], and the calculated target angle would no longer correspond to the true incidence cosine. The adjustment would still rotate the normal, but we did not evaluate the pipeline under such conditions, so we can't report on the accuracy in such cases.

The simulated LiDAR scan does not introduce sensor noise and we did not vary scan parameters (viewpoint count, range, resolution) across runs. Bigger variety of real world LiDAR data would test the pipeline under noise and density conditions that our evaluation does not cover. Intensity in particular is known to depend on range, incidence angle, and atmospheric conditions in ways that the simulator does not fully reproduce.

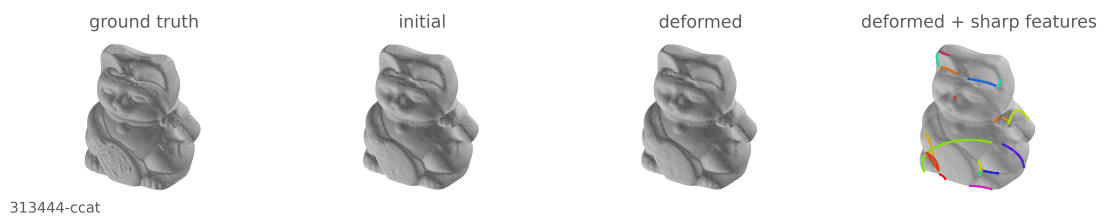
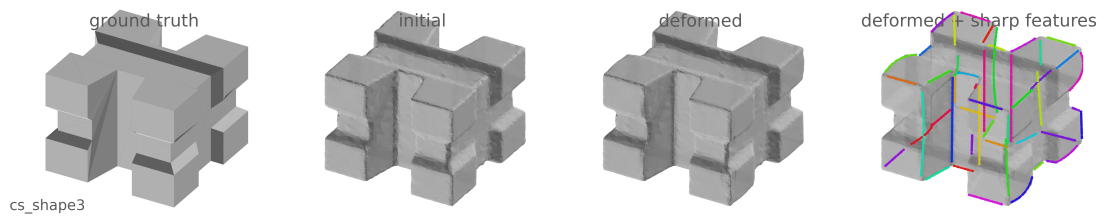
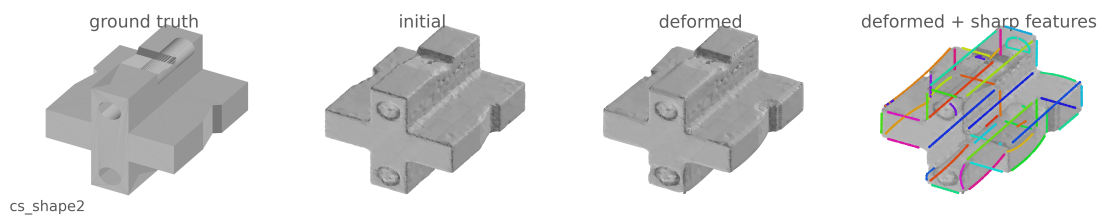
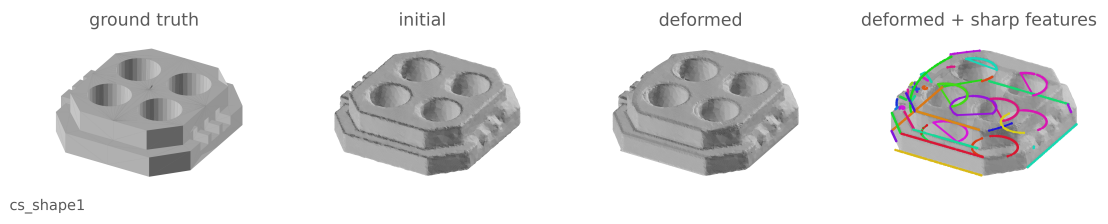
Finally, even the sharp-feature-restricted Hausdorff variant we introduce in Section 4.2 is a closest-point metric: it requires only that a sample on a ground-truth sharp edge be close to *some* point on the reconstruction. A sample can therefore register a small distance even when the reconstruction has no sharp feature at the corresponding location, as long as a smooth face happens to pass nearby. A stricter feature-to-feature metric, which constrains the closest point on the reconstruction to itself lie on a detected sharp edge, would be a more direct measurement of whether the deformation has correctly placed sharp features. We did not implement such a metric.

Final Output

Here we present example of how the final output looks like for these objects, in each row there is the initial Poisson reconstructed mesh, final deformed mesh and deformed mesh with visible feature chains.



4. RESULTS





Conclusion

This thesis investigated whether scan angles derived from LiDAR backscatter intensity can be used to improve the reconstruction of sharp features in surface meshes. We presented a pipeline that estimates point normals with PCA, refines them using an intensity-derived incidence angle under a Lambertian reflectance assumption, reconstructs a surface mesh, and deforms that mesh toward detected sharp feature chains to recover edges and corners that Poisson reconstruction tends to smooth out. We evaluated the approach on ten models drawn from the Thingi10K, ABC datasets and from custom-authored CAD shapes, comparing against a ground truth obtained from a simulated LiDAR scan.

The intensity-based adjustment improved the accuracy of the estimated normals on every model in the test set. The magnitude of the improvement was geometry-dependent: smooth organic surfaces, where the adjustment’s neighborhood assumptions hold best, benefited most, while models with a high density of sharp features benefited less. At the level of the final reconstructed surface, the deformation visibly recovers sharp features lost in the Poisson reconstruction. The sharp-feature-restricted Hausdorff metric confirms this quantitatively: on six of the eight sharp-feature models, the deformed variants reduce the mean sharp-edge distance below the initial Poisson baseline. The global Hausdorff distance, in contrast, is dominated by smooth-region samples and does not distinguish the deformation’s contribution.

We also found that the improvement in normal accuracy did not translate into a consistent, measurable improvement in the final mesh surface when comparing reconstructions with and without the adjustment. We attribute this to the discrete nature of feature detection: a continuous improvement in normal quality only changes the outcome for points near a decision threshold, and it can move points across that threshold in either direction. Overall, the results show that intensity-derived scan angles are useful and inexpensive processing-wise for improving normal estimation, and that the benefit is most pronounced precisely where standard PCA estimation is weakest, but that propagating this benefit

cleanly through to the final surface deformation remains limited by the downstream feature-detection stage, at least evaluated by the used metric.

5.1 Future work

Lambertian Reflectance Model & Multi-material scenes. The intensity-based adjustment assumes a local constant surface albedo, which holds for the single-material test models used here but would break down at material boundaries. Extending the method to detect or estimate albedo changes, for example by segmenting the point cloud by some material property (e.g. reflectance) before the adjustment, would allow it to be applied to scenes that mix materials and finishes. It would be also beneficial to use the combined Lambertian-Beckmann model [TTC⁺21] that works better on both diffuse and specular materials.

Evaluation on real LiDAR data. All results for the evaluation were obtained from a simulated LiDAR scan that does not fully reproduce sensor noise or the dependence of intensity on range and atmospheric conditions. Evaluating the pipeline on variety of real scan data would test whether the improvements observed here hold under realistic noise and intensity behavior, and is a necessary step toward practical use.

Unified parameter configuration. Pipeline parameters were tuned per model in this work. Finding a single configuration, or an automatic parameter-selection scheme, that performs well across a range of geometries would make the pipeline more practical and would allow a fairer assessment of its robustness.

Feature-to-feature evaluation. The sharp-feature-restricted Hausdorff metric 4.3 introduced in this work measures the distance from ground-truth sharp samples to the closest point on the reconstructed surface. A stricter metric that requires the closest point on the reconstruction to itself lie on a detected sharp feature would more directly evaluate whether the deformation has placed sharp features in the right location, rather than merely whether the reconstructed surface passes near the ground-truth feature. This would be an improvement for evaluating feature-aware reconstruction methods.

Overview of Generative AI Tools Used

AI tools were used during the writing and development of this bachelor thesis. Claude Sonnet 4.5, 4.6 and Opus 4.6 was used for code development, mainly in the code regarding visualization, for researching relevant algorithms and helping with creating diagrams in the thesis, for example 3.1. For writing, Claude was sometimes used to improve the structure and clarity of sentences, without altering the content or meaning of the original.

List of Figures

3.1	Overview of the whole computational pipeline.	8
3.2	Detection of sharp features, classified either as edge feature (green) or corner (blue).	16
3.3	The tangents $\mathbf{t}_i$ and $\mathbf{t}_j$ at the two edge feature points must be nearly parallel, i.e. their absolute dot product must exceed τ_{tan}	18
3.4	Left: a valid connection along a single edge, where the connection direction $\mathbf{d}_{ij}$ aligns with both tangents. Right: two parallel edges facing each other, where $\mathbf{d}_{ij}$ is orthogonal to both tangents and the test $ \mathbf{d}_{ij} \cdot \mathbf{t}_i \geq \tau_{\text{dir}}$ and $ \mathbf{d}_{ij} \cdot \mathbf{t}_j \geq \tau_{\text{dir}}$ fails, keeping the edges separate.	18
3.5	Clustered feature edges. Each color represents a distinct feature chain. . .	20
3.6	Results after the deformation stage.	25
4.1	3D objects used for evaluation.	27
4.2	Detected sharp features on reconstructed mesh.	28
4.3	Normals orientation difference on reconstructed mesh.	28

List of Tables

3.1	Normal estimation and adjustment parameters.	9
3.2	Surface reconstruction parameters.	12
3.3	Feature detection and refinement parameters.	13
3.4	Feature clustering and chain extraction parameters.	17
3.5	Feature fitting and projection parameters.	21
3.6	Model selection priority.	22
3.7	Mesh deformation and smoothing parameters.	23
4.1	Normal estimation accuracy with and without intensity-based adjustment, using $k_{adj} = 20$	29
4.2	Surface accuracy of the reconstructed meshes measured by the Hausdorff distance from the reconstruction to the ground truth. Bold entries mark cases where a deformed variant improves on the initial Poisson reconstruction. .	30
4.3	Sharp-feature-restricted Hausdorff distance. Bold entries mark cases where a deformed variant improves on the initial Poisson reconstruction.	31

List of Algorithms

Bibliography

- [ABCO⁺03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
- [BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.
- [CDR07] Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1105, 2007.
- [ECR⁺26] Mohamed Emara, Marta Castellini, Cristiano Riminesi, Mohamed A. Jaziri, Andrea Pandurino, Alessandro Conti, Lidia Fiorini, Adele Meucci, Riccardo Colella, Francesco Taurino, Davide Zecca, Francesco Giuri, Matteo Greco, Alessandra Chirivì, Grazia Tucci, and Alberto Bucciero. IoT-driven digital twin based on VR for smart monitoring of cultural heritage sites: La Grotta degli Animali in florence case study. In Lucio Tommaso De Paolis, Pasquale Arpaia, and Marco Sacco, editors, *Extended Reality (XR Salento 2025)*, volume 15741 of *Lecture Notes in Computer Science*, pages 496–515. Springer, Cham, 2026.
- [Far02] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, San Francisco, CA, 5th edition, 2002.
- [FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [Fou13] David F. Fouhey. Multi-model estimation in the presence of outliers. https://www.researchgate.net/publication/266881743_Multi-Model-Estimation-in-the-Presence-of-Outliers, 2013. Undergraduate thesis, Carnegie Mellon University. Accessed: 2026-03-22.

- [FSYL23] Yi-Fei Feng, Li-Yong Shen, Chun-Ming Yuan, and Xin Li. Deep shape representation with sharp feature preservation. *Computer-Aided Design*, 157:103468, 2023.
- [GKU11] M. Gschwandtner, R. Kwitt, and A. Uhl. BlenSor: Blender sensor simulation toolbox. In *Advances in Visual Computing: 7th International Symposium (ISVC 2011)*, Las Vegas, Nevada, USA, September 2011.
- [HDD⁺92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 71–78. ACM Press, 1992.
- [HWG⁺13] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics*, 32(1):1–12, 2013.
- [Jel92] Albert V. Jelalian. *Laser Radar Systems*. Artech House, 1992.
- [KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In Konrad Polthier and Alla Sheffer, editors, *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 61–70. Eurographics Association, 2006.
- [KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanerke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 57–66. ACM Press, 2001.
- [KH13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, 2013.
- [KMJ⁺19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9601–9611, 2019.
- [Lev04] David Levin. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, pages 37–49. Springer, 2004.
- [Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [LXXW23] Qi Liu, Shibiao Xu, Jun Xiao, and Ying Wang. Sharp feature-preserving 3D mesh reconstruction from point clouds based on primitive detection. *Remote Sensing*, 15(12):3155, 2023.

- [MOG11] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J. Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, 2011.
- [NF24] Franco Niccolucci and Achille Felicetti. Digital twin sensors in cultural heritage ontology applications. *Sensors*, 24(12):3978, 2024.
- [PDHF07] Norbert Pfeifer, Peter Dorninger, Alexander Haring, and Hongchao Fan. Investigating terrestrial laser scanning intensity data: Quality and functional relations. In *Proceedings of the VIII Conference on Optical 3D Measurement Techniques*, pages 328–337, Vienna, Austria, 2007.
- [PHB⁺08] Norbert Pfeifer, Bernhard Höfle, Christian Briele, Martin Rutzinger, and Alexander Haring. Analysis of the backscattered energy in terrestrial laser scanning data. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XXXVII, pages 1045–1052, Beijing, China, 2008.
- [PKG03] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–289, 2003.
- [Poi25a] Point Cloud Library Contributors. How 3d features work in PCL – normal estimation. https://pcl.readthedocs.io/projects/tutorials/en/latest/normal_estimation.html, 2025. Accessed: 2026-03-22.
- [Poi25b] Point Cloud Library Contributors. PCL NormalEstimation class reference. https://pointclouds.org/documentation/classpcl_1_1_normal_estimation.html, 2025. Accessed: 2026-03-22.
- [Poi25c] Point Cloud Library Contributors. PCL RandomSampleConsensus class reference. https://pointclouds.org/documentation/classpcl_1_1_random_sample_consensus.html, 2025. Accessed: 2026-03-22.
- [Poi25d] Point Cloud Library Contributors. PCL SampleConsensusModelCircle3D class reference. https://pointclouds.org/documentation/classpcl_1_1_sample_consensus_model_circle3_d.html, 2025. Accessed: 2026-03-22.
- [Poi25e] Point Cloud Library Contributors. PCL VoxelGrid class reference. https://pointclouds.org/documentation/classpcl_1_1_voxel_grid.html, 2025. Accessed: 2026-03-22.
- [QLP⁺22] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 23192–23204, 2022.

- [SYM10] Nader Salman, Mariette Yvinec, and Quentin Mérigot. Feature preserving mesh generation from 3D point clouds. *Computer Graphics Forum*, 29(5):1623–1632, 2010.
- [The25a] The CGAL Project. CGAL poisson surface reconstruction. https://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html, 2025. Accessed: 2026-03-22.
- [The25b] The CGAL Project. CGAL polygon mesh processing. https://doc.cgal.org/latest/Polygon_mesh_processing/index.html, 2025. Accessed: 2026-03-22.
- [TTC⁺21] Wenxin Tian, Lingli Tang, Yuwei Chen, Ziyang Li, Jiajia Zhu, Changhui Jiang, Peilun Hu, Wenjing He, Haohao Wu, Miaomiao Pan, Jing Lu, and Juha Hyypä. Analysis and radiometric calibration for backscatter intensity of hyperspectral LiDAR caused by incident angle effect. *Sensors*, 21(9):2960, 2021.
- [WJHM15] Martin Weinmann, Boris Jutzi, Stefan Hinz, and Clément Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304, 2015.
- [WXX⁺20] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. PIE-Net: Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems*, 33:20167–20178, 2020.
- [YLF⁺18] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *European Conference on Computer Vision (ECCV)*, 2018.
- [ZJ16] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.