machines

MDPI

Article

Deep Fusion of Kinematic Features and Task-Aware Partition Planning for Mold Surface Robotic Polishing

Miao Yu *, Xu Liu, Baowen He and Zhen Pan

School of Mechanical and Vehicle Engineering, Changchun University, Changchun 130022, China
* Correspondence: yum72@ccu.edu.cn

Abstract

Robotic polishing in CAD-free industrial settings relies on point-cloud data, yet noise and non-uniform sampling often compromise kinematic feasibility and finishing quality. This paper proposes an adaptive motion planning approach with explicit kinematic constraints. A downsampling–clustering–mapping-back strategy is first employed for rapid workpiece extraction. Subsequently, an improved supervoxel representation and attributed adjacency graph (AAG) are developed, utilizing a multi-objective energy formulation to partition sub-regions that satisfy geometric consistency and kinematic reachability. To handle point-cloud noise, a lightweight neural network predicts scanning directions and step-distance coefficients, followed by thick-slice serpentine path generation. Finally, closed-loop verification ensures safety through inverse-kinematics and safety-margin checks. Experimental results demonstrate consistent sub-micron finishing quality, with Ra $\approx$ 0.6 μm on complex mold surfaces. Moreover, the proposed pipeline achieves a 7.5$\times$ preprocessing speedup, completing workpiece extraction in 1.14 s for a 237,640-point scan, and improves kinematic feasibility to 100% IK success while reducing the mean TCP normal deviation by ~76% compared with a PCA-based baseline.

Keywords: robotic polishing; point cloud; toolpath planning; kinematics; supervoxel; mold surface

1. Introduction

Robotic manipulators have become the primary platform for precision finishing of mold surfaces due to their high accuracy, repeatability, and flexibility [1,2]. Yet mold surfaces often feature sharp curvature changes, complex boundaries, and reachability limitations, imposing multiple constraints on trajectory planning: coverage completeness, orientation continuity, efficiency, and safety [3], and potentially energy consumption. In addition, energy consumption minimization has been considered as an objective in trajectory planning, aiming to select energy-efficient motions rather than optimizing geometric criteria alone [4]. Automatic trajectory generation that is both efficient and high-quality for such workpieces is still challenging [5]. In this context, 'efficient' refers to the minimization of computational time during the preprocessing and planning stages, while 'high-quality' denotes a trajectory that ensures full kinematic feasibility (collision-free, singularity-free) and consistent surface coverage.

Manual teaching-based offline programming lacks robustness in ensuring consistent quality. CAD-based offline programming has matured as an alternative, with research focused on geometric slicing, tool-axis orientation planning, and process modeling.

Check for updates

Academic Editor: Panagiotis Kyratsis

Received: 27 January 2026
Revised: 17 February 2026
Accepted: 19 February 2026
Published: 21 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Wang et al. [6] integrated manipulator polishing process planning with existing CAD systems for systematic offline trajectory and parameter generation. Jamshidi and Budak [7] embedded a CAD-based polishing force model into robotic toolpath planning for complex surface finishing. However, these methods become impractical when reliable CAD models are unavailable or when fixture deviations and wear-induced geometry changes occur.

To reduce dependence on idealized models and precise calibration, sensor-based automatic path planning has become a research focus in recent years. In particular, 3D vision systems can directly acquire point cloud data from physical objects and enable model-free geometric-driven planning [8]. Wang et al. [9] generated initial toolpath points from measured point cloud slicing and further smoothed path curves and tool orientation fluctuations using spline interpolation and optimization strategies, thereby improving robustness to discontinuities caused by noise. Xiao et al. [10] proposed a model-driven polishing toolpath planning method for mold surfaces represented by point clouds, unifying model-free geometric representation with trajectory generation within a unified computational framework. Ren et al. [11] developed a point-cloud-driven polishing path planning system using slice projection to extract features; however, errors tend to propagate when point clouds are sparse or locally incomplete, and mesh partitioning strategies directly affect efficiency and stability.

Despite recent progress, point-cloud-driven polishing of curved mold surfaces still faces three practical limitations:

- Efficiency Bottlenecks: Large-scale point clouds impose significant computational and storage burdens, often creating efficiency bottlenecks in processing.
- Lack of Task-Oriented Organization: Existing segmentation methods prioritize geometric consistency but lack task-oriented region organization, making it difficult to explicitly consider coverage efficiency, tool direction-change costs, and reachability risks.
- Insufficient Verification: Planning evaluation typically remains at the geometric level, lacking a unified closed-loop verification framework that integrates kinematic reachability and safety margins.

Consequently, the primary objective of this study is to establish a comprehensive, closed-loop robotic polishing framework that specifically addresses the limitations of existing methods regarding computational efficiency, kinematic reachability, and process consistency. By integrating a rapid preprocessing strategy (D-C-M) with task-aware graph partitioning, this work aims to bridge the gap between unorganized point clouds and executable, high-quality robot trajectories for complex mold surfaces.

2. Point-Cloud Preprocessing and Workpiece Foreground Extraction via the D-C-M Strategy

Measured point clouds acquired in industrial settings typically contain substantial background noise and outliers, with data scales often reaching millions of points [12]. Directly performing region partitioning and path planning on raw high-resolution point clouds would incur prohibitive computational cost and render downstream algorithms highly sensitive to noise and background interference. This section aims to rapidly extract high-precision workpiece foreground point clouds from mixed point clouds while preserving surface details as much as possible.

Due to the inherent systematic errors in optical measurement devices, the raw point cloud $P = \{p_i\}_{i=1}^{N}$ inevitably contains outlier noise points (Figure 1a) [13]. To ensure the numerical stability of subsequent topological computations, statistical outlier removal is first applied to P. Specifically, the mean distance $\overline{d}_i$ from each point p_i to its k-nearest

neighbors is computed (Equation (1)), based on which a Gaussian-assumption-based rejection criterion is established:

$$\overline{d_i} = \frac{1}{k} \sum_{j \in N_k(i)} ||p_i - p_j|| \tag{1}$$

where k is the number of nearest neighbors, $\mathcal{N}_k(i)$ denotes the index set of the k-NN of point p_i, and N is the total number of points.

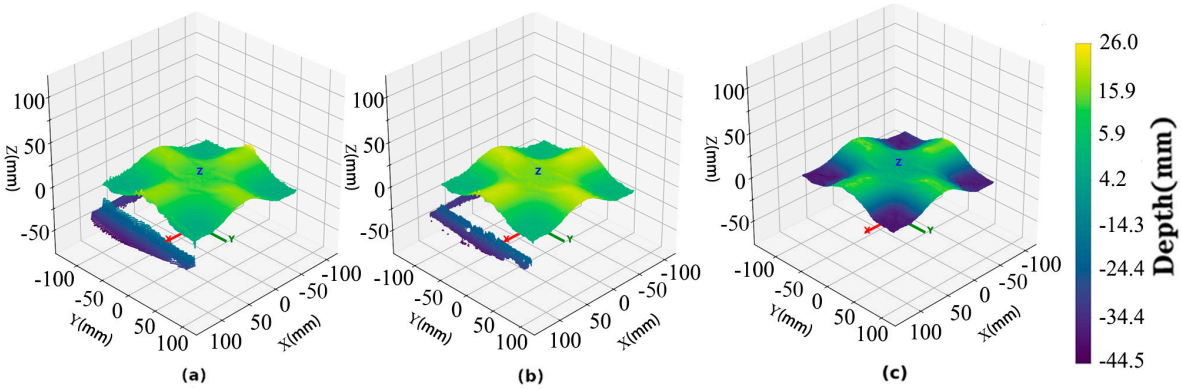


Figure 1. Extraction of target workpiece cluster from raw point cloud. (**a**) Original point cloud; (**b**) Preprocessed data; (**c**) Background removed.

If $\overline{d_i}$ satisfies Equation (2), the corresponding point is identified as an outlier and removed, thereby yielding a denoised point cloud (Figure 1b).

$$\overline{d_i} > \mu + \alpha\sigma \tag{2}$$

Here, μ and σ denote the mean and standard deviation of the global distribution of $\overline{d_i}$, respectively, and α is a sensitivity coefficient (details are provided in Table A1 of Appendix A).

Although the above statistical filtering (Equations (1) and (2)) effectively removes discrete measurement noise, the resulting point cloud remains fundamentally a mixed data field containing both the target workpiece and non-machining background. To address this, we propose a downsampling–clustering–mapping-back (D-C-M) strategy. This approach aims to tackle the dual challenge of background separation and feature fidelity, with spatial decoupling as its core principle. The algorithm first introduces a voxel-based downsampling mechanism to construct a sparse representative point set from the denoised dense data. The specific construction process is described below.

To reduce the data scale while preserving the macroscopic spatial structure of the point cloud, a voxel grid is first introduced to discretize the filtered point cloud. Given a voxel edge length v_0, the geometric centroid of all points within each non-empty voxel is computed, yielding a compact representative point set $D = \{\widetilde{p}_j\}_{j=1}^{M} (M \ll N)$, as defined in Equation (3). This procedure maps the complex raw data space onto a sparse representative point space, substantially shrinking the search domain for subsequent clustering.

$$\widetilde{P}_j = \frac{1}{|V_j|} \sum_{P \epsilon V_j} P \tag{3}$$

where V_j denotes the set of raw points falling into the j-th non-empty voxel, and $|\ V_j\ |$ is its cardinality.

Although voxelization reduces the data scale, background noise still needs to be removed. Therefore, density-based clustering is performed in the reduced space. Specifically,

DBSCAN is applied to the representative point set D [14]. Since the number of points involved in clustering decreases from N to M, the computational complexity is reduced from $O(N \log N)$ to $O(M \log M)$. As shown in Figure 2a, density-based clustering can effectively identify and separate the target workpiece (the red dominant cluster) from background disturbances (the gray and blue clusters), and each representative point $\widetilde{p}_j$ is assigned a cluster label l_D.

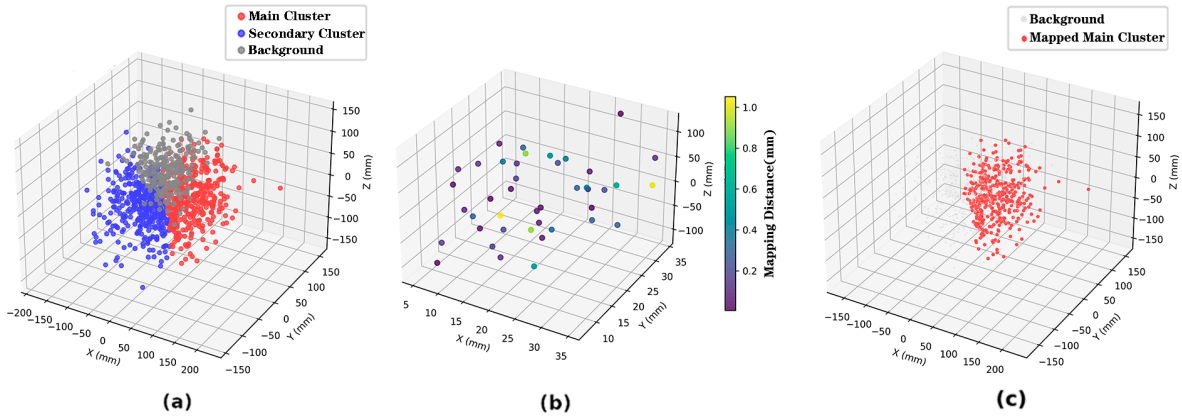


Figure 2. Schematic of the D-C-M strategy. (**a**) Downsampled clustering (D-C-M Stage 1&2); (**b**) local magnification and mapping error heat map; (**c**) high-resolution mapping (D-C-M Stage 3).

Although clustering results distinguish the workpiece from the background, the representative point set D loses fine-grained surface details. To recover the original resolution, a label mapping-back operation is performed. For any point $p \in P$ in the raw point cloud, its nearest neighbor $\widetilde{p}^*$ in the representative set D is queried using a KD-tree, and the cluster label is propagated according to spatial proximity [15]:

$$l_P(p) = \begin{cases} l_D(\widetilde{p}^*), & if \ ||p - \widetilde{p}^*|| \leq \delta_{map}, \\ -1, & otherwise. \end{cases} \tag{4}$$

where $l_P(\cdot)$ and $l_D(\cdot)$ denote the cluster labels in the raw and representative point sets, respectively; $\widetilde{p}^*$ is the nearest representative point of p; c is a scale coefficient; δ_{max} is the upper bound of the mapping threshold; and label -1 indicates unassigned.

Here, the mapping-back threshold is defined as $\delta_{map} = \min(\delta_{max}, cv_0)$, which is introduced to prevent excessive label diffusion into unrelated regions. As shown in Figure 2c, this step successfully projects the semantic information from the low-dimensional representative space back to the original high-dimensional space. The mapped point cloud preserves a macroscopic density distribution consistent with that of the raw point cloud. A small number of points near the boundary remain unlabeled, resulting in a mapping-back success rate of 98.5%.

The core advantage of the D-C-M strategy lies in exchanging minimal geometric position uncertainty for a 7.5-fold improvement in processing speed (see Table 1). From a theoretical error perspective, the displacement deviation introduced by voxelization is bounded by the voxel diagonal length (Equation (5)). In practical data, point clouds concentrate on the workpiece surface, and the statistical error is significantly smaller than this conservative upper bound.

$$||p - \widetilde{p}|| \leq \sqrt{3}v_0, \tag{5}$$

Table 1 provides a quantitative comparison demonstrating the performance advantages of this method. Compared to clustering directly on full-resolution point clouds or using downsampling data alone, the D-C-M strategy processes large-scale point clouds in

just 1.14 s while maintaining a 98.5% mapping-back success rate. This demonstrates that the method not only meets the geometric accuracy requirements for manipulator polishing but also exhibits the potential for real-time performance compatible with industrial production cycles.

Table 1. Comparison of the D-C-M strategy with other background removal methods.

	N (Original)	M (Downsampled)	Mapping-Back Success Rate (%)	Mapping Distance 95th (mm)	Processing Time (s)	Speedup
D-C-M	237,640	11,882	98.5	1.78	1.14	7.5
DS-Only	237,640	11,882	N/A	N/A	0.78	11.0
FR-DBSCAN	237,640	N/A	100.0	N/A	8.59	1.0

3. Task-Aware Region Partitioning for Manipulator Polishing

Although the preprocessing strategy described above effectively removes background noise and yields a high-accuracy workpiece point cloud, the resulting data remain an unstructured and semantically disconnected set of discrete points. Direct global toolpath planning on such data can hardly meet the requirements of manipulator polishing in terms of trajectory continuity and kinematic reachability [16]. To bridge the gap between geometric disorder and machining order, this section presents a coarse-to-fine task-aware region partitioning framework. Specifically, the scattered point cloud is first supervoxelized into geometrically homogeneous units. An attributed adjacency graph is then constructed to model the topological relationships among these units. Finally, a multi-objective energy optimization that integrates kinematic constraints is formulated to generate continuous task sub-regions that are executable by the manipulator.

Before proceeding, to explicitly quantify kinematic feasibility in the subsequent partitioning stage, the robotic polishing system is modeled as a six-degree-of-freedom serial manipulator (Figure 3). Its workspace is constrained by joint limits. The reachability metric used in Equation (18) is computed based on the inverse-kinematics (IK) solver associated with this specific kinematic chain.

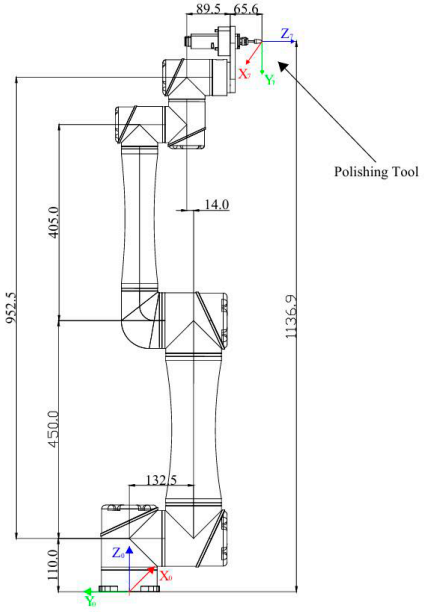


Figure 3. A six-degree-of-freedom serial manipulator for robotic polishing (unit: mm).

3.1. Supervoxel Construction for the Sampled Point Cloud

To transform the unstructured point cloud into an optimizable structured representation, this stage discretizes the processed point cloud P into a supervoxel set $S = \{s_u\}_{u=1}^{N_s}$, where each supervoxel serves as a basic unit with locally consistent geometry. The classical Voxel Cloud Connectivity Segmentation (VCCS) algorithm often suffers from uneven seed distribution and poor boundary adherence when handling non-uniformly sampled point clouds [17]. To address these limitations, this paper introduces adaptive improvements to both the initialization and region-growing mechanisms of VCCS.

During the initialization stage, to enhance the sensitivity of the algorithm to mold-surface features, we abandon the conventional VCCS seeding strategy that relies on a fixed resolution. Instead, a density-adaptive seed estimation mechanism is adopted. Specifically, the number of seeds n_{seeds} is dynamically determined from the axis-aligned bounding box (AABB) volume V of the point cloud and the desired spatial resolution r_s (Equation (6)). Subsequently, a feature space that integrates spatial coordinates and surface normals is constructed, and K-means initialization is performed within this space. This strategy encourages seeds to be preferentially distributed in regions with large curvature variations rather than being uniformly scattered over flat areas, thereby ensuring that the resulting segmentation units can better represent the underlying surface morphology from the outset.

$$n_{seeds} = \min\left(\left\lfloor \frac{V}{r_s^3} \right\rfloor, \frac{N}{S_{min}^{seg}}, N_{max}^{seed}\right) \tag{6}$$

Here, S_{min}^{seg} denotes the minimum number of points required for a single supervoxel, and N_{max}^{seed} is the predefined upper bound on the number of seeds.

During the region-growing stage, to prevent supervoxels from crossing sharp edges or right-angle boundaries of the workpiece, and thus blurring geometric features, a strict normal-consistency constraint is introduced [18]. A candidate neighboring point p_j is merged into the current supervoxel only when it simultaneously satisfies the spatial distance constraints (Equations (7) and (8)) and the normal-angle criterion with respect to the current seed point as defined in Equation (9). This constraint mitigates over-segmentation and ensures a high degree of geometric continuity within each generated supervoxel. After the growth process, a feature descriptor is extracted for each supervoxel s_u, including its centroid, mean normal vector, and shape index. These attributes are further normalized using PCA-weighted standardization and serve as the basic properties for subsequent topological modeling.

$$||p_j - p_i|| < d_{step} \tag{7}$$

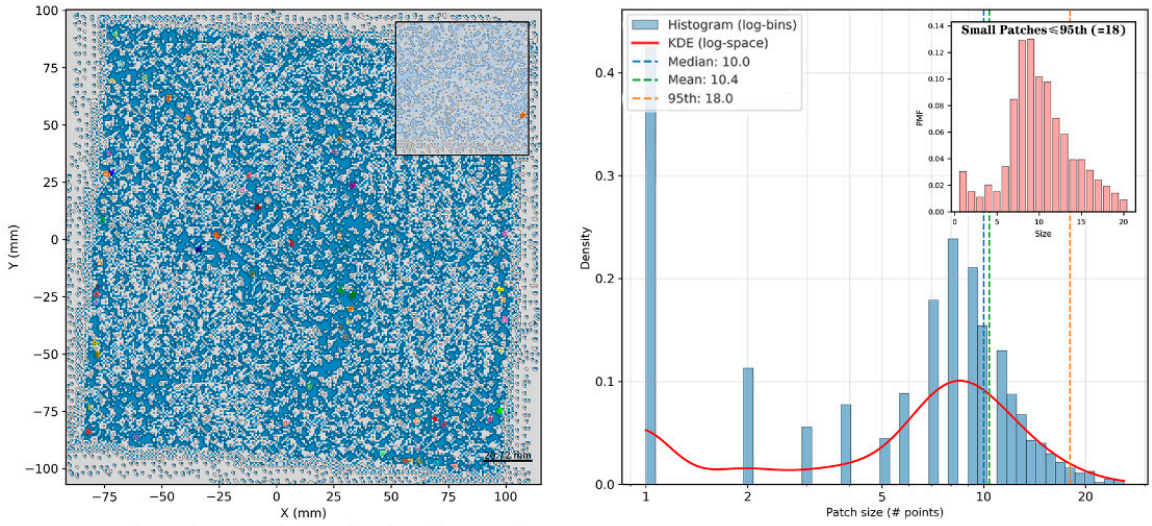
$$||p_j - p_{seed}|| < d_{global} \tag{8}$$

$$\angle(n_i, n_j) \leq \theta_{seg} \tag{9}$$

where p_i is the current seed point, p_j is a neighboring candidate point, p_{seed} is the initial seed of the current supervoxel, n_i and n_j are the corresponding estimated normals, and $\angle(n_i, n_j)$ denotes the included angle between two normals.

Here, θ_{seg} is weakly adapted to the supervoxel scale in the experiments to prevent region growing from expanding across curvature discontinuities and sharp edges.

The supervoxelization results are illustrated in Figure 4, where the point cloud is decomposed into small patches with relatively uniform geometric characteristics. The supervoxel size is jointly constrained by r_s, d_{step}, d_{global}, and S_{min}^{seg}, leading to an overall multi-modal distribution. In implementation, $d_{global} = c_g r_s$ and $d_{step} = c_s \bar{d}_{\text{knn}}$, where $\bar{d}_{\text{knn}}$ denotes the local mean KNN spacing. The coefficients c_g and c_s are fixed across all experiments (details are provided in Table A1 of Appendix A).



(a)Colored supervoxel visualization in the grid map

(b)Patch-size distribution (log-binned histogram with KDE), including summary statistics

Figure 4. Supervoxelization results and patch-size statistics. (**a**) Colored supervoxel visualization in the grid map, where different colors represent distinct individual supervoxels; (**b**) patch-size distribution (log-binned histogram with KDE), including summary statistics.

3.2. Attributed Adjacency Graph Construction and Edge Weight Modeling

After obtaining the supervoxel set, this section constructs an undirected weighted attributed adjacency graph (AAG) $G = (V, E, W)$, which encodes local topological relationships and multi-source similarities into a weighted graph structure to serve as the input for subsequent optimization. Here, the vertex set V corresponds one-to-one to the supervoxels. Each vertex $v_u \in V$ encapsulates the supervoxel centroid c_u, mean normal n_u, and a 16-dimensional feature vector f_u, forming the basic information unit for the following graph-based optimization.

During the graph connectivity construction, to accommodate the anisotropic scale variations in curved mold surfaces, we discard the conventional neighborhood criterion based on a fixed radius and instead design a scale-adaptive dynamic thresholding mechanism. The decision of whether to establish an edge between two supervoxels is determined by a spatial distance threshold $d_{thr}(i, j)$ and a normal-angle threshold $\theta_{thr}(i, j)$, as defined in Equations (10) and (11).

$$d_{thr}(i,j) = \text{clip}\left(d_0\left(1 + \eta \log \frac{m_i + m_j}{2m_{\text{ref}}}\right), d_{min}, d_{max}\right) \tag{10}$$

$$\theta_{thr}(i,j) = \text{clip}\left(\theta_0\left(1 + \beta\left(1 - \frac{d_{ij}}{d_{thr}(i,j)}\right)\right), \theta_{min}, \theta_{max}\right) \tag{11}$$

where d_0 and θ_0 are the base distance/normal thresholds, η and β are scale-adaptation coefficients, (d_{min}, d_{max}) and $(\theta_{min}, \theta_{max})$ are lower/upper bounds, and c_i, c_j denote the centroids of supervoxels s_i, s_j. Here, $d_{ij} = \| c_i - c_j \|$ denotes the centroid distance between two supervoxels. m_i and m_j represent the numbers of points contained in the corresponding supervoxels and are used as a proxy for their scales, while m_{ref} is a reference scale. The clipping operator is defined as $\text{clip}(x, a, b) = \min(\max(x, a), b)$ to ensure that the thresholds remain within a reasonable range.

This dynamic mechanism endows the algorithm with scale adaptivity. In relatively flat regions with large curvature radii, longer-range connections are allowed to accelerate merging. In contrast, in areas with abrupt curvature changes or fine geometric details, stricter spatial and normal constraints are enforced. Specifically, no edge is established

when the average curvature difference exceeds the threshold, i.e., $\mid \kappa_i - \kappa_j \mid > \kappa_{thr}$. As a result, erroneous connections across sharp edges are effectively blocked during graph construction, thereby preserving the sharp geometric features of the workpiece.

During the edge-weight computation stage, to precisely characterize the merging tendency between adjacent nodes, the final edge weight w_{final} is designed as a composite measure of local geometric continuity and global feature consistency. As shown in Equation (12), this weight is constructed by linearly coupling the geometric similarity term w_{geom} and the high-dimensional feature similarity S_{feat}:

$$w_{final}(i,j) = \gamma \cdot w_{geom}(i,j) + (1-\gamma) \cdot S_{feat}(i,j) \tag{12}$$

Here, the balancing factor γ (details are provided in Table A1 of Appendix A). is introduced to trade off local geometric continuity and global feature consistency, where w_{geom} is the geometric similarity term (distance/normal/curvature-based) and S_{feat} is the feature similarity computed in Equations (13) and (14).

To eliminate the influence of different physical scales, the geometric similarity w_{geom} integrates the spatial Euclidean distance, normal-angle deviation, and curvature difference. Each component is mapped into the [0, 1] range via a Gaussian kernel, thereby ensuring a normalized and comparable metric.

The feature similarity S_{feat} is designed to capture higher-order shape variations. Considering that different dimensions of the 16-dimensional feature descriptor contribute unequally to shape characterization, a PCA explained-variance-weighted Euclidean distance is employed (Equation (13)), which is further transformed into a similarity score using an exponential function (Equation (14)):

$$distance_{weighted(i,j)} = \sqrt{\sum_k w_k (f_{i,k} - f_{j,k})^2} \tag{13}$$

$$S_{feat}(i,j) = \exp\left(-\lambda \cdot distance_{weighted(i,j)}\right) \tag{14}$$

To determine the optimal balancing factor γ in Equation (12), we conducted a systematic parameter perturbation study. As illustrated in Figure 5, we analyzed the trade-off between segmentation quality (Fusion Quality Score) and kinematic reachability (Graph Connectivity Ratio) by sweeping the weighting parameter γ from 0 to 1. The results show a clear Pareto frontier: increasing the kinematic weight improves graph connectivity but slightly degrades the geometric adherence of supervoxels. At low γ (<0.2), the connectivity is insufficient (<70%) to guarantee a complete toolpath. As γ increases, connectivity rises sharply and saturates around γ = 0.65. Beyond this point, further increasing the weight yields diminishing returns in connectivity while excessively penalizing geometric features. Therefore, we empirically set γ = 0.5 (near the "elbow" point) to balance high segmentation quality with robust kinematic connectivity.

Here, $f_{i,k}$ denotes the k-th component of the feature vector of the i-th supervoxel. The weight w_k is adaptively determined from the PCA explained-variance ratio and satisfies $\sum_k w_k = 1$, thereby automatically emphasizing features with higher discriminative power. The hyperparameter λ controls the steepness of similarity decay.

As shown in Figure 6, this composite weighting strategy results in strong connections only within regions where nodes are not only spatially adjacent but also highly consistent in surface texture and morphological attributes. In contrast, for nodes that are spatially close yet exhibit substantially different attributes, the edge weights are suppressed. This provides clear gradient cues and boundary priors for the subsequent energy-minimization-based graph-cut optimization.

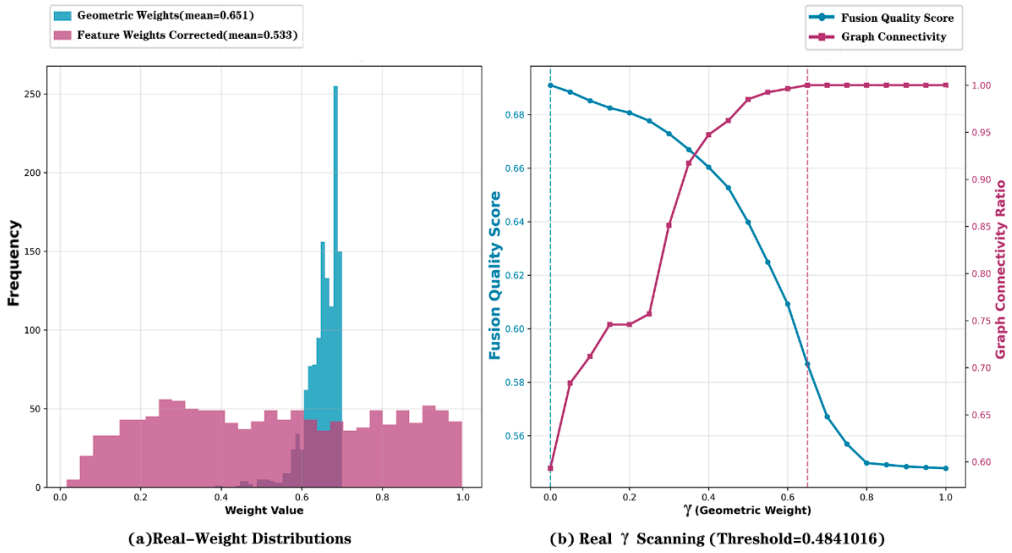


Figure 5. Sensitivity analysis of the weighting parameter γ.

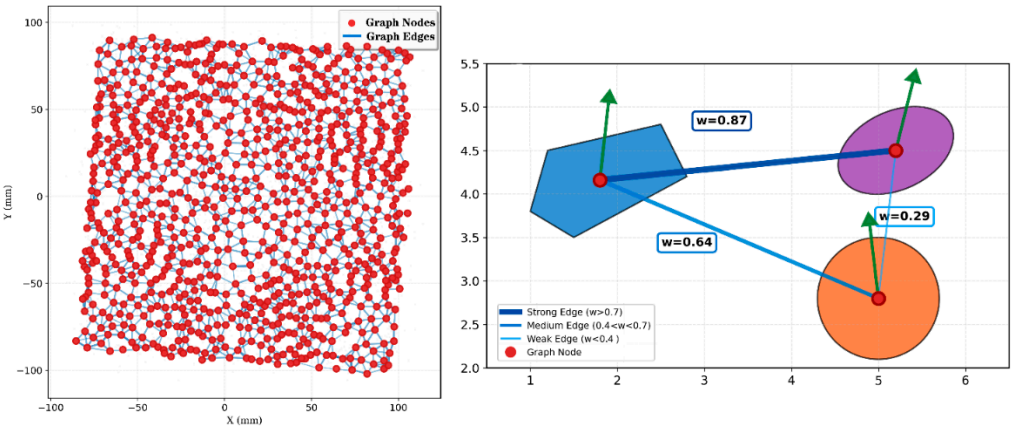


Figure 6. Attributed adjacency graph (AAG) construction and edge-weight computation. The green arrows in subfigure (**b**) represent the mean normal vectors of the corresponding nodes.

3.3. Energy-Minimization Task Partitioning for Executable Manipulator Polishing

Based on the constructed attributed adjacency graph, the region partitioning problem is mathematically formulated as a graph partitioning problem. To overcome the limitation of conventional geometric segmentation methods that lack awareness of the manipulator's machining capability, we cast the partitioning as a multi-term energy minimization problem. In this formulation, the energy terms represent geometric continuity, compactness, and manipulator reachability, respectively. The goal is to find an optimal partition $R = \{R_k\}$ that minimizes the total system energy E_{total}. As shown in Equation (15), the energy function is defined as a weighted sum, which enables a tight coupling between geometric features and task constraints:

$$E_{total} = \sum_{R \in \mathcal{R}} [w_h E_{hom}(R) + w_c E_{compact}(R) + w_r E_{reach}(R) + w_p E_{path}(R)] \quad (15)$$

Here, w_h, w_c, w_r, and $w_p \geq 0$ are the weighting coefficients for the corresponding energy terms.

To ensure the geometric integrity of the partitioned regions, the energy function first introduces a homogeneity term E_{hom} and a compactness term $E_{compact}$. Specifically, $E_{hom}(R)$ (Equation (16)) penalizes the dispersion of feature vectors within each region relative to

the regional mean μ_R, thereby encouraging the separation of regions with significantly different surface properties:

$$E_{hom}(R) = \sum_{v_u \in R} \| \mathrm{f}_u - \mu_R \|_2^2, \ \mu_R = \frac{1}{|R|} \sum_{v_u \in R} \mathrm{f}_u \tag{16}$$

The compactness energy $E_{compact}(R)$ is constructed based on the coefficient of variation in the centroid-distance distribution within each region, aiming to suppress the formation of elongated or fragmented regions. Let $r_u = \| c_u - \bar{c}_R \|$ denote the distance from node u to the region centroid $\bar{c}_R$. This term is defined as:

$$E_{compact}(R) = \frac{\sigma(r)}{\mu(r) + \varepsilon} \tag{17}$$

Here, $\mu(r)$ and $\sigma(r)$ denote the mean and standard deviation of the distance set, respectively.

Building upon the geometric constraints, to endow the partitioning results with machining executability, the model further introduces two key task-aware terms. The reachability energy $E_{reach}(R)$ explicitly incorporates the manipulator's kinematic constraints into the optimization objective:

$$E_{reach}(R) = 1 - S_{reach}(R) \tag{18}$$

Here, $S_{reach}(R) \in [0, 1]$ is obtained by uniformly sampling n_r points within the region and computing the success rate of inverse-kinematics (IK) solutions. This term drives the optimization to avoid region configurations located near the workspace boundary or in the vicinity of kinematic singularities [17].

In addition, the path-complexity energy $E_{path}(R)$ aims to optimize the texture-processing direction on the surface. The algorithm projects the regional point set onto the local tangent plane, computes the eigenvalues $\lambda_1 \geq \lambda_2$ of the resulting 2D covariance matrix, and defines:

$$E_{path}(R) = \frac{\lambda_2}{(\lambda_1 + \varepsilon)} \tag{19}$$

Minimizing this energy functional tends to generate serpentine strip-shaped regions with dominant elongation directions, a morphology that aligns well with the subsequent serpentine scanning strategy, thereby minimizing tool reorientation transitions and reducing the dynamic impact of accelerations and decelerations on the robotic arm.

Due to the non-convexity of the objective function ΔE_{total} and the high computational cost of E_{reach}, direct global optimization constitutes an NP-hard problem [19]. To address this, we design an efficient two-stage greedy merging strategy. The algorithm first utilizes the low-cost consistency increment ΔE_{hom} (Equation (20)) to rapidly screen potential merge candidates in $O(1)$ time:

$$\Delta E_{hom}(A, B) = \frac{n_A n_B}{n_A + n_B} ||\mu_A - \mu_B||^2 \tag{20}$$

where n_A and n_B are the numbers of nodes in regions A and B. Subsequently, only the prescreened candidate pairs undergo recalculation of the complete energy variation $\Delta E_{total}(A, B)$ including the kinematic term, yielding precise energy gains.

A merge is accepted if $\Delta E_{total}(A, B) < \tau_{energy}$. Here, we set $\tau_{energy} = -1.0 \times 10^{-4}$ to ensure strict energy reduction and avoid merges driven by numerical noise. Iterating this process until no region pair can reduce total energy further yields a stable task-aware partitioning. Figure 7 shows that valid merges monotonically decrease total energy throughout iteration, converging to a stable solution.

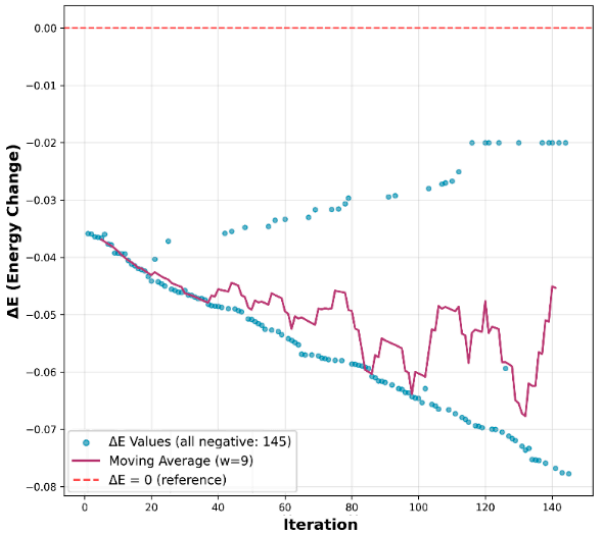


Figure 7. Energy convergence and optimization.

To further eliminate sawtooth artifacts and isolated fragments at region boundaries and provide continuous region contours for subsequent path planning, the optimized region boundaries undergo lightweight normal-weighted smoothing to suppress jaggedness and avoid crossing sharp edges [20]. Figure 8 shows the final region partitioning.

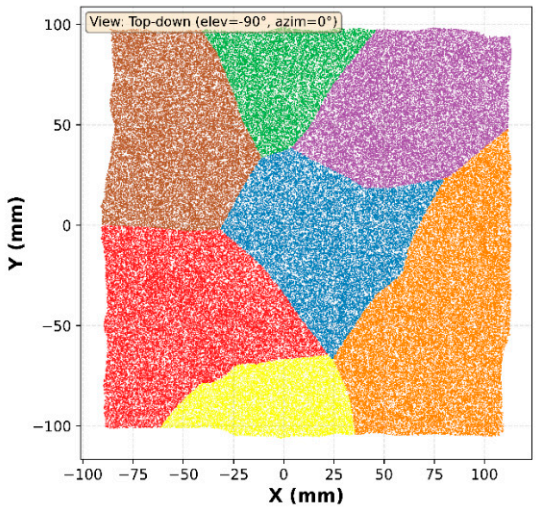


Figure 8. Region partitioning results. Different colors are used to visually distinguish distinct partitioned sub-regions.

4. Coverage Toolpath Generation for Manipulator-Executable Polishing

Based on the task-aware partitioning obtained in the previous section, the mold surface has been structured into a set of sub-regions suitable for machining. However, to generate high-quality polishing trajectories within these unstructured sub-regions, it is still necessary to overcome the local geometric uncertainty and the physical constraints of manipulator motion. Traditional purely geometric planning methods often struggle to simultaneously ensure coverage rate while taking into account trajectory smoothness and executability. For this purpose, this section proposes a hybrid path planning method assisted by data-driven decision-making. We will first utilize a deep learning model to replace traditional geometric operators to obtain robust process principal directions, then combine them with a thick-slicing technique to generate topologically consistent coverage

paths, and finally complete the trajectory temporal mapping from task space to joint space through kinematic optimization.

4.1. Learning-Based Process Parameter Prediction via Deep Geometric Feature Fusion

Although task-aware partitioning in the previous section decomposes the complex global planning problem into lower-dimensional local subproblems, generating concrete paths still requires addressing the challenge of local geometric uncertainty. Near curvature discontinuities or boundaries, PCA-based principal directions derived from second-order statistical moments tend to produce unstable results or even sign flips [21]. Additionally, fixed step distances struggle to simultaneously balance machining efficiency and consistent material removal [22]. To address this, we propose a data-driven process parameter decision method that jointly predicts the region scanning direction t^* and step-distance coefficient k^* using a lightweight dual-head network, thereby replacing conventional analytical operators.

To achieve geometric perception beyond conventional analytical operators, we construct an informative feature space capturing local topological context. A 128-dimensional task-aware descriptor fuses multi-scale geometric cues as network input (Table 2). The backbone uses a lightweight MLP (~7.4×10^4 parameters), designed with shared features and parallel dual-head branches for simultaneous direction and step-distance coefficient prediction (Figure 9).

Table 2. Composition of the 128-D task-aware descriptor used for direction and step-distance prediction.

Feature Group	Dim.	Representative Statistics (Examples)
Geometric statistics	20	Coordinate moments, AABB size, aspect ratios
PCA-based shape cues	15	Principal-component magnitudes and ratios
Curvature descriptors	20	Variance, extrema, curvature ratios
Normal distribution	30	Mean, normal consistency, dispersion
Boundary cues	15	Boundary-point ratio, edge length,
Density statistics	10	Local point density and multi-scale spacing
Higher-order interactions	18	Mixed statistics across multiple cues

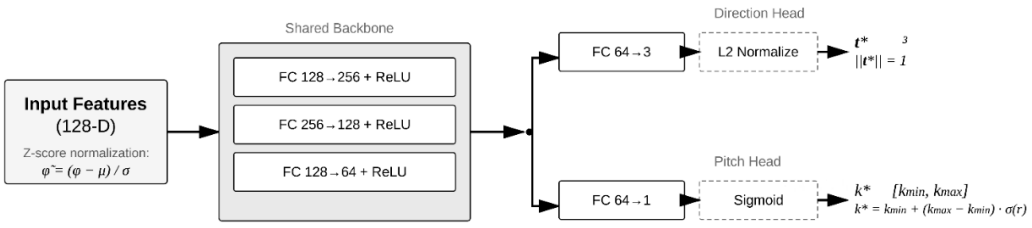


Figure 9. Joint scanning direction–step-distance coefficient prediction and process mapping framework based on lightweight MLP.

To eliminate training instability caused by features with different scales, all input feature vectors undergo Z-score normalization, mapping them to a standard normal distribution space. This accelerates gradient backpropagation and model convergence.

To enhance the stability of normals and curvatures under noisy and non-uniform sampling conditions, we introduce a learnable geometric attribute estimator to predict point cloud normals and curvatures, aggregating them into region-level statistical features as network input. The estimator is first pretrained on public point cloud attribute datasets and then fine-tuned on acquired workpiece point clouds to adapt to the actual data distribution [23,24].

Regarding supervision signal construction, the ground-truth scanning direction t_{gt} is obtained through offline pseudo-label generation: we constructed a dataset containing approximately 20,000 local point cloud patches, of which 80% are used for training and 20% for validation. The ground-truth step-distance coefficient k_{gt} is determined starting from a baseline step distance derived from the tool's effective contact width and preset overlap ratio, and the step distance range is then cropped and calibrated according to regional curvature statistics, forming (t_{gt}, k_{gt}) label pairs available for training [25].

Based on the region-level features and offline-generated supervision signals described above, we train the lightweight MLP to jointly predict the region scanning direction t^* and step-distance coefficient k^*.

The direction prediction head outputs a vector $d \in \mathbb{R}^3$, which is strictly constrained to a unit scanning direction t^* through L_2 normalization.

$$t^* = \frac{d}{\| d \|_2 + \varepsilon}, d \in \mathbb{R}^3 \tag{21}$$

where ε is a small positive constant preventing division by zero.

The step-distance coefficient head produces an unbounded scalar r, which undergoes Sigmoid squashing (Equation (22)) followed by affine scaling to the range $[k_{min}, k_{max}]$, obtaining the bounded step-distance coefficient k^*.

$$k^* = k_{min} + (k_{max} - k_{min})\sigma(r),\ \sigma(x) = \frac{1}{1 + e^{-x}} \tag{22}$$

where k_{min} and k_{max} are specified by process parameters.

With (t^*, k^*) obtained, we further compute the region baseline strip step distance s_{base}, which serves as the step advance and strip generation parameter for the thick-slice serpentine coverage in Section 4.2, defined as:

$$s_{base} = k^* \cdot 2 \cdot R_{eff} \cdot (1 - overlap) \cdot \cos(\theta_{tilt}) \tag{23}$$

where $2R_{eff}$ is the effective contact band width, *overlap* is the inter-strip overlap ratio, and $\cos(\theta_{tilt})$ projects contact width to the direction orthogonal to scanning direction in the tangent plane, yielding effective line spacing. The predicted t^* determines the strip progression vector u_g and guides the thick-slice serpentine path generation as a directional constraint [26].

For joint optimization of direction and step predictions, training uses a weighted loss function $\mathcal{L}$ balancing the two objectives:

$$\mathcal{L} = \lambda_{dir}\mathcal{L}_{dir} + \lambda_k\mathcal{L}_k \tag{24}$$

where the direction loss adopts a cosine similarity form $\mathcal{L}_{dir} = 1 - \mid t^* \cdot t_{gt} \mid$, the step-distance loss uses mean squared error $\mathcal{L}_k = \| k^* - k_{gt} \|_2^2$, and λ_{dir} and λ_k are balance coefficients. Training employs early stopping and weight decay to suppress overfitting.

Figure 10 presents comparative results between this method and the conventional PCA approach. It can be seen that in flat regions (Figure 10a,b), the predicted scanning direction t^* is essentially consistent with PCA directions; however, near curvature discontinuities or boundaries (Figure 10c,d), conventional PCA directions exhibit obvious deflection and sign flips, while the proposed t^* maintains consistency with local surface geometry, further indicating that the deep learning-based approach demonstrates superior robustness in complex regions [27].

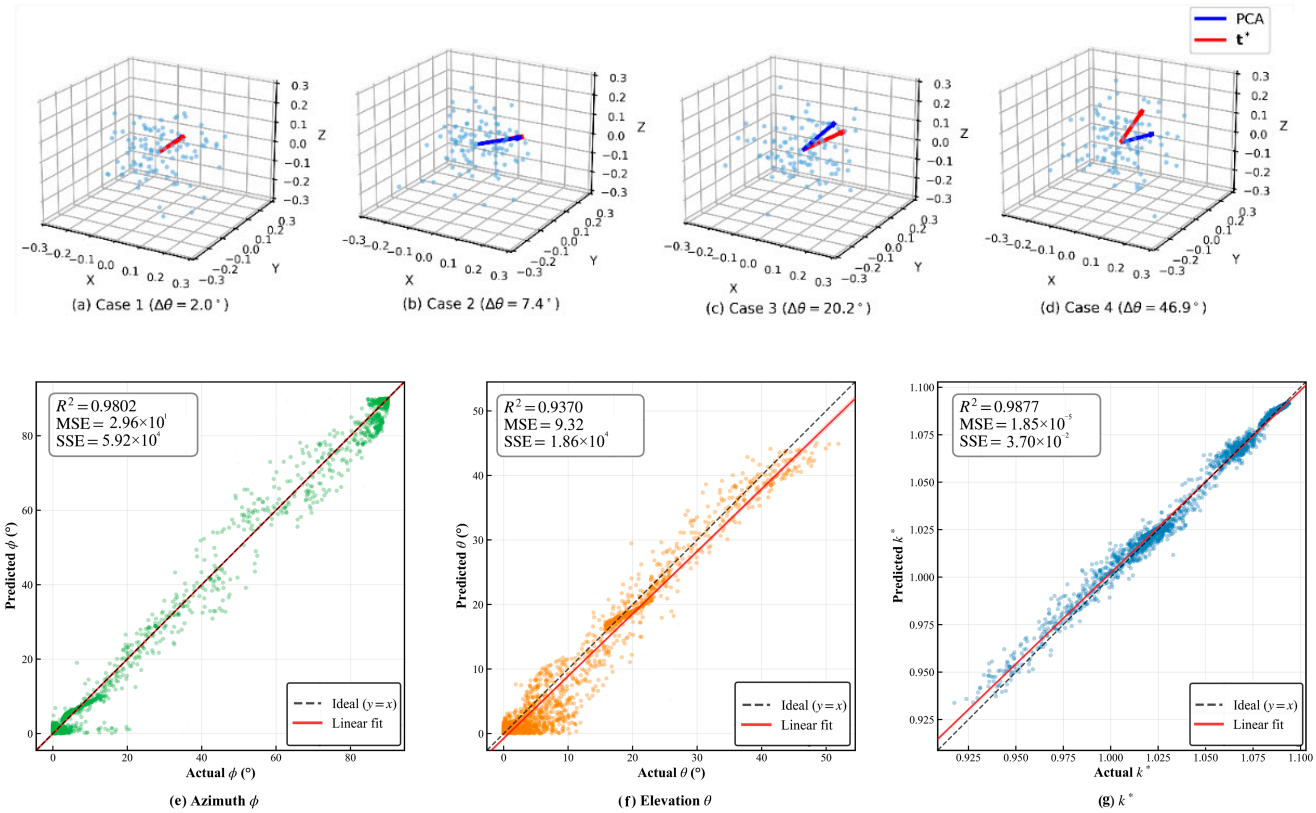


Figure 10. Qualitative and quantitative evaluation of the prediction model. The colored dots in subfigures (**e**–**g**) represent the individual test samples.

To quantitatively evaluate the prediction accuracy of the proposed MLP, we further plot the regression between predicted and ground-truth values for the azimuth φ, elevation θ, and step-distance coefficient k^* on the test set (N = 2000), as shown in Figure 10e–g. The plots report R^2 as well as error metrics (MSE and SSE), indicating strong agreement between predictions and ground truth for all three outputs.

Through the above processing, the sign ambiguity of the scanning direction t^* within regions is eliminated, establishing a globally consistent direction field representation. However, this consistency addresses the sign stability of the geometric direction field; in the subsequent serpentine coverage, to minimize idle motion, the tool traversal directions of adjacent strips naturally exhibit alternating changes, which represents the tool reversal phenomenon arising from trajectory organization strategy and will be discussed in the next section regarding its orientation continuity handling.

4.2. Topology-Aware Thick-Slice Serpentine Path Planning

Upon obtaining locally optimal scanning direction t^* and step-distance coefficient k^* from the deep learning model, the core objective of path planning is to convert these discrete process parameters into continuous, smooth, and complete-coverage mechanical arm trajectories. Rather than employing traditional projection methods, we introduce a topology-aware thick-slice approach: stratifying the region three-dimensionally along the strip advancement direction u_g and mapping the complex surface manifold to a local two-dimensional parametric domain, thus enabling both regularized strip generation and topology-cognizant serpentine connectivity [28,29].

Since serpentine coverage uses reciprocal tool motions to enhance machining efficiency, the feed directions of adjacent strips necessarily alternate. To maintain the continuity of the end-effector orientation and machining contact throughout the coverage, orientation

interpolation and heading selection must be carefully constrained at strip transitions and direction reversals. We construct a K-nearest neighbor (KNN) undirected graph over the regional point cloud and propagate normal vector consistency via breadth-first search (BFS). Specifically, if the angular difference between normal vectors at two adjacent points exceeds 90°, the normal of the neighboring point is flipped to enforce local consistency: $n_i \cdot n_j > 0, \forall (i,j) \in \mathcal{N}$, where $\mathcal{N}$ denotes the adjacency set in the KNN graph. Following local consistency enforcement, the region-averaged normal is computed as $\bar{n} = \frac{1}{N}\sum_{i=1}^{N} n_i$. The desired outward normal direction is represented by the unit vector $e_z = (0,0,1)^\top$ aligned with the work frame z-axis. Should $\bar{n} \cdot e_z < 0$, all region normals are globally flipped to enforce global consistency: the normals are unified to reside within the half-space defined by e_z.

Figure 11a illustrates the local k-nearest neighbor graph prior to BFS-based normal consistency, wherein some reversed normals are present (indicated by red arrows). Figure 11b demonstrates that following BFS propagation and global orientation alignment, all normals are uniformly directed toward the +z direction of the work coordinate system, with only a representative subset of normals rendered for visualization purposes.

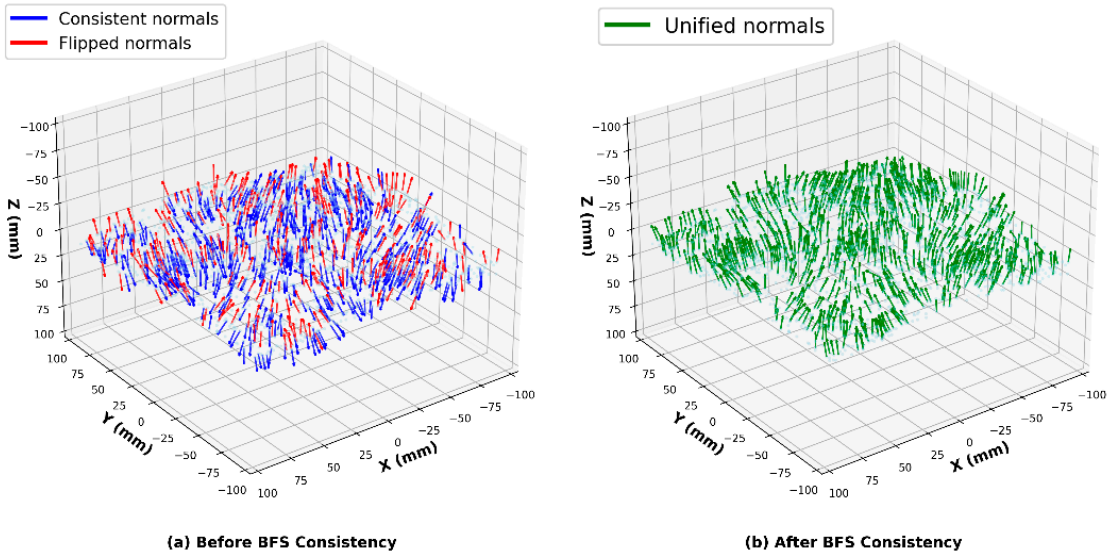


Figure 11. Normal consistency before and after BFS processing.

Despite achieving global normal consistency, residual noise points at the region boundaries may still induce path perturbations during the subsequent filling operation. To address this, an SDF-based inward offset is applied to the Region of Interest (ROI) prior to slice generation. Taking the reference step size s_{base}, as defined in Equation (23), the inward offset distance is formulated as

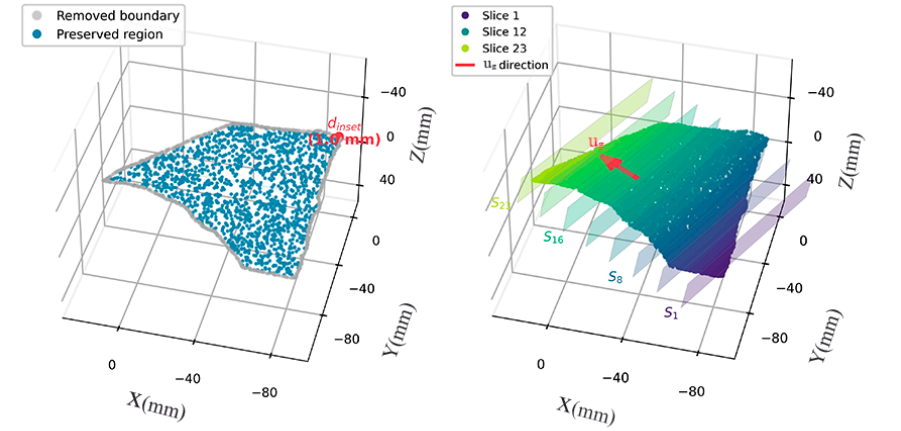
$$d_{inset} = \alpha s_{base},\ \alpha = 0.5 \tag{25}$$

The ROI point cloud is orthogonally projected onto a region-local 2D plane and discretized through rasterization; this 2D coordinate plane is constructed based on the region's mean normal vector and scanning direction. The signed distance field (SDF) is computed on this 2D grid via Euclidean distance transform; for each 3D point p, orthogonal projection yields the 2D grid coordinates $(u_g(p), v_g(p))$, which are subsequently used to query the corresponding grid SDF value and perform threshold-based filtering. The 2D coordinates (u_g, v_g) are constructed from the region's mean normal $\bar{n}$ and scanning direction t^*, following the construction scheme detailed in Equations (28) and (29), with

n_{avg} replaced by $\bar{n}$. The SDF computation and query operations are performed on this region-level 2D rasterized grid.

To ensure connectivity preservation, the outermost boundary contour of the region is first extracted from the 2D grid and subjected to a morphological closing operation, which fills small cracks and voids; subsequently, the signed distance field is computed only within the interior of this outer contour boundary. Consequently, the threshold criterion $\mathrm{SDF}(p) > d_{inset}$ effectively removes only a narrow strip along the region periphery, thereby preserving the topological connectivity of the inset ROI in most cases; should fragmentation occur, the largest connected component is retained while isolated components are discarded [30].

For each point p in the original point cloud, the signed distance to the region boundary $\mathrm{SDF}(p)$ is computed. Only points satisfying the condition $\mathrm{SDF}(p) > d_{inset}$ are preserved, thereby effectively eliminating unstable boundary artifacts and yielding the inset ROI point cloud, as illustrated in Figure 12a.



(a)Point clouds before and after ROI contraction (b)Schematic of thick slicing based on stepping direction

Figure 12. ROI inset and directional thick-slicing.

For the processed ROI point cloud, the algorithm performs thick-slice discretization along the strip stepping direction u_g. The projection direction u_g is constructed by the cross product of the region's average normal and the scanning direction t^* (Equation (26)).

$$d_i = p_i \cdot u_g \tag{26}$$

where $u_g = \frac{\bar{n} \times t^*}{\|\bar{n} \times t^*\|}$ is a unit vector.

The slice layer thickness is set to $h = s_{base}$. Compared to single-layer slicing, this strategy partitions the point cloud into several overlapping controlled subsets S_i (Equation (27)), effectively ensuring the integrity of local geometric features [31].

$$S_i = \left\{ p_j \mid d_{min} + ih \le p_j \cdot u_g < d_{min} + (i+1)h \right\} \tag{27}$$

where $d_{min} = min_i d_i$ and $d_{max} = max_i d_i$.

To enable dimensionality reduction in the three-dimensional path planning problem, a locally following coordinate frame (u, v, t^*) is established for each slice subset (Equations (28) and (29)), wherein the v-axis is aligned with the principal tangential direction, thereby transforming the complex spatial point cloud into a structured local 2D parametric domain.

$$u = \frac{u_g - (u_g \cdot n_{avg}) n_{avg}}{\| u_g - (u_g \cdot n_{avg}) n_{avg} \|}, v = u \times n_{avg} \tag{28}$$

When $\| u_g - (u_g \cdot n_{avg}) n_{avg} \|$ is too small, it degenerates to constructing u and v using the global coordinate system to avoid numerical instability.

$$u_i = p_i \cdot u, v_i = p_i \cdot v \tag{29}$$

Within the local parametric plane, a set of approximately parallel scanning lines $u = u_0 + m\, s_{base}$ are established at uniform intervals in the u direction with spacing s_{base}. Points along each scanning line are sorted according to the v-coordinate and subsequently subjected to sparse resampling along the v direction using s_{base} as the target inter-point distance, thereby generating equidistant parallel processing strips.

The algorithm constructs parallel scanning line sets Band_m (Equation (30)) with uniform spacing s_{base} and performs coordinate-based sorting and resampling operations on the point cloud along the v-axis. To address the idle-motion inefficiency inherent in conventional path planning, a topology-aware serpentine connection strategy is employed: adjacent scanning line endpoints are alternately connected based on parity, and transition segments are strategically inserted across slice boundaries following a minimum-distance criterion. The serpentine path topology ensures monotonic progression through the parametric domain, effectively preventing path self-intersection and abrupt normal discontinuities. Figure 13 illustrates the projection of the serpentine polyline path (computed for a single region) back onto the 3D surface, where gray lines represent scanning line boundaries, blue dots denote equidistant sample points, and red polylines depict the resultant serpentine machining trajectory.

$$\text{Band}_m = \left\{ p \in S_i \mid\mid u(p) - u_m \mid < \frac{s_{base}}{2} \right\},\ u_m = u_0 + m s_{base} \tag{30}$$

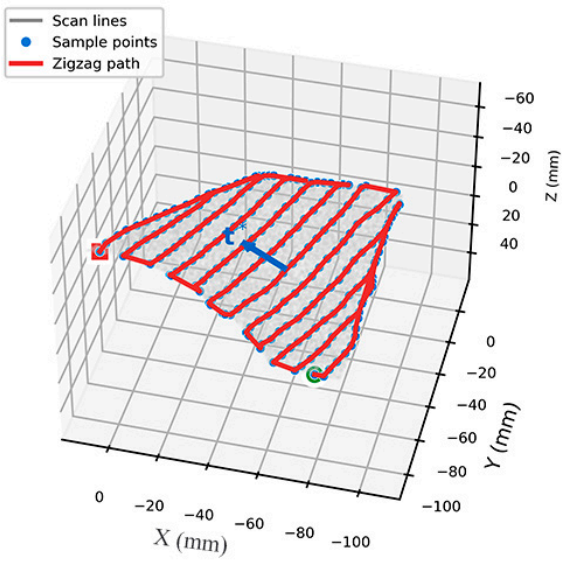


Figure 13. Schematic diagram of the serpentine path back-projected onto the 3D curved surface.

The initial path is geometrically connected; however, the sharp corners at U-shaped turns easily cause acceleration discontinuities during manipulator motion. To address this dynamics challenge, we introduce piecewise B-spline fitting for parametric fitting of the machining segments and employ cubic Bezier curves for corner rounding transitions. To prevent excessive smoothing that causes loss of workpiece edge features, the algorithm introduces a surface normal angle threshold $\Delta\theta_{max}$: when the pose deviation caused by fitting is too large, the algorithm automatically reverts to the original path, thereby

achieving a dynamic balance between geometric fidelity and motion smoothness in high-curvature regions.

$$\Delta\theta_{max} = \max_i \arccos(n_i \cdot n_{i+1}) \tag{31}$$

where n_i is the surface outward normal at the i-th sample point of the path. During computation, $n_i \cdot n_{i+1}$ is clamped to $[-1, 1]$ to avoid numerical errors.

Through normal consistency and thick-slice planning, as well as adaptive trajectory refinement processing, we have successfully constructed a complete path from discrete point clouds to continuous machining trajectories. This path does not yet include the joint physical constraints of the manipulator and the dynamic properties of the time dimension. To transform the static geometric curve into motion commands that can be directly executed by the manipulator controller, the next section will further introduce inverse kinematics solving and dynamics constraints, elucidating the mapping and optimization methods from task-space poses to joint-space time trajectories [32].

4.3. Kinematically Feasible Pose Mapping and Temporal Trajectory Parameterization via Inverse Kinematics

The path point set generated in the previous section satisfies geometric coverage requirements; however, to convert it into executable instructions for the manipulator, we must solve the nonlinear mapping problem from Cartesian space to joint space while simultaneously satisfying pose process requirements, joint continuity, and dynamics limits.

First, to ensure consistency of the polishing contact state, we need to transform geometric points with only position information into TCP poses containing six-degree-of-freedom information. Unlike traditional machining, manipulator polishing requires the tool axis to strictly follow the local surface normal or maintain a specific inclination angle [31]. For this purpose, we construct the following local surface coordinate system $R_s(k)$ based on the path tangent t_k and surface normal n_k:

$$R_s(k) = \begin{bmatrix} \hat{t}_k & b_k & n_k \end{bmatrix} \tag{32}$$

where $\hat{t}_k$ is the normalized feeding direction, and the binormal vector b_k is determined by the right-hand rule. To eliminate non-orthogonality caused by measurement noise, Gram–Schmidt orthogonalization correction is introduced during the construction process. On this basis, considering that practical processes often require setting a forward tilt angle or lateral tilt angle to optimize material removal rate, the final tool coordinate system pose $R_{tcp}(k)$ is defined as the result of rotating the local frame about the binormal by tilt angle θ_{tilt}:

$$R_{tcp}(k) = R_s(k) R_{tilt}(\theta_{tilt}) \tag{33}$$

Meanwhile, combined with the geometric offset h_c, corresponding to the contact wheel radius and preload force, the spatial position of the TCP is determined as

$$p_{tcp}(k) = p_k + h_c \, z_{tcp}(k), z_{tcp}(k) = R_{tcp}(k) \, e_z \tag{34}$$

After obtaining a continuous sequence of Cartesian poses, the algorithm faces the core challenge of maintaining smooth joint configuration among multiple IK solutions. The kinematic redundancy of the manipulator may lead to drastic jumps or even singularities in joint space at adjacent trajectory points. To address this problem, we introduce the minimum displacement criterion when solving the inverse kinematics:

$$q_k = IK\big(p_{tcp}(k), R_{tcp}(k)\big), q_{min} \le q_k \le q_{max} \tag{35}$$

where q_{min}, q_{max} are the physical joint stroke limits for each joint of the manipulator. During the solving process, the solution closest to the previous joint configuration q_{k-1} in the Euclidean distance sense is selected to avoid frequent jumps among multiple inverse solution branches. For individual trajectory points that are reachable in the workspace but infeasible in IK due to pose constraints, they are only used for diagnostic and region-level statistics; the final output trajectory for evaluation is the repaired feasible sequence, and the IK success rate is subsequently calculated based on this.

Finally, to convert the discrete joint angle sequence into a dynamic trajectory executable by the controller, the path must undergo time parameterization. This paper adopts a trapezoidal velocity profile-based planning algorithm, assigning a time scale t to the joint sequence, ensuring that at any instant the velocity and acceleration of each joint are strictly limited within physical bounds:

$$| \dot{q}_i(t) | \leq \dot{q}_{i,max} \ , \ | \ddot{q}_i(t) | \leq \ddot{q}_{i,max} \tag{36}$$

where $\dot{q}_{i,max}, \ddot{q}_{i,max}$ are the joint velocity and acceleration limits set by the manufacturer. Through time parameterization, the geometrically continuous joint path is converted into a time trajectory $q(t)$ satisfying the controller's dynamic constraints, which can be directly used for manipulator controller interpolation and execution.

To ensure operational safety in unstructured environments, the system incorporates a continuous collision detection module based on signed distance fields. During trajectory generation, the algorithm computes in real time the minimum signed distance $\phi(\cdot)$ from the sample point set $\mathrm{x}_{k,m}$ on the tool entity to environmental obstacles. Only when the entire trajectory satisfies the following safety margin constraint is it judged as valid [33]:

$$\phi(\mathrm{x}_{k,m}) \geq d_{\text{safe}} \tag{37}$$

This geometry–physics dual verification mechanism ensures that the final generated trajectory not only meets process requirements but also possesses high robustness and safety in engineering implementation.

Through the systematic processing from trajectory refinement to kinematic mapping described above, we have successfully transformed unstructured measurement point clouds into executable time trajectories $q(t)$ that satisfy manipulator dynamics constraints. At this point, the trajectory planning architecture for point-cloud-driven polishing of mold surfaces has been constructed at the theoretical level.

5. Robotic Polishing Simulation and Experiments

This section systematically verifies the proposed method from both theoretical and engineering practice perspectives. First, through a high-precision simulation environment, the kinematic feasibility, safety, and process geometric features of planned trajectories are quantitatively assessed. On this basis, a physical polishing platform is established, through which the surface roughness metrics after actual machining verify the effectiveness and consistency of simulation analysis conclusions in physical machining.

The workpiece used in this study is a complex curved injection-mold core made of S136 (AISI 420) stainless steel. The material was heat-treated to a hardness of 48–52 HRC. S136 is widely used for injection-mold tooling due to its good polishability and corrosion resistance. In the heat-treated condition, its microstructure is typically dominated by tempered martensite, which is generally associated with stable mechanical properties and reduced variability in abrasive material removal.

5.1. Simulation-Based Trajectory Executability and Process-Level Quality Evaluation

In the scenario of point-cloud-driven polishing on mold surfaces, the point-cloud-driven polishing path planning method proposed in this paper is systematically verified. The simulation platform employs the Gazebo–MoveIt integrated framework under ROS [34]. The experimental subject is a six-degree-of-freedom industrial manipulator, with its key parameters specified, as shown in Table 3. The visual input adopts point clouds acquired by Mech-Eye LOG-S, and after preprocessing, the point cloud serves as both the planning input and collision detection scenario [35]. The planning result generated by the thick-slice serpentine algorithm is shown in Figure 14.

Table 3. Basic attributes of six-DOF industrial manipulator.

Attributes	Values
DOF	6
Rated payload (kg)	5
Arm extension (mm)	952.5
Repeatability (mm)	±0.03
Max TCP speed (m/s)	3
Robot weight (kg)	16.5

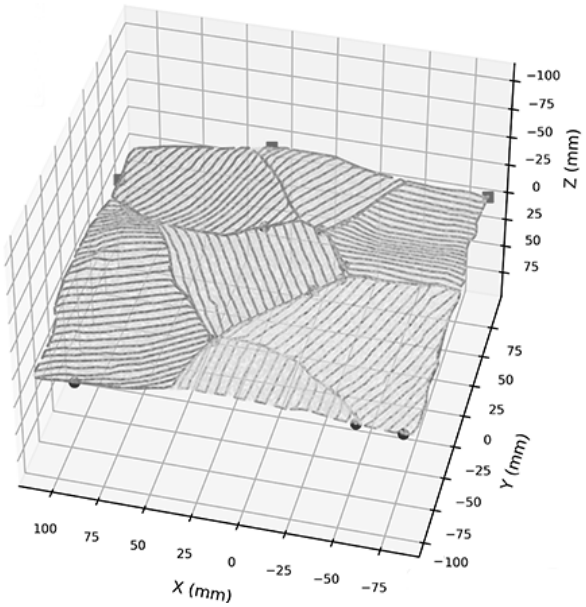


Figure 14. Visualization of serpentine polishing trajectory generated by thick-slice serpentine planning in simulation.

To quantitatively evaluate trajectory executability, this paper adopts the signed distance field (SDF) as the primary performance metric. Specifically, the SDF metric quantifies the minimum Euclidean distance between the tool entity geometry and the nearest environmental obstacle, continuously computed in real-time throughout the trajectory execution.

Figure 15 illustrates the SDF margin variation curves for seven sub-regions generated by the planning algorithm. Statistical results (shown in Table 4) indicate that, in a trajectory containing 9283 path points in total length, the global minimum SDF margin is 2.09 mm. Only when the entire trajectory satisfies the safety margin constraint (Equation (37)) is it judged as valid. In our simulations, we set $d_{safe} = 2.0$ mm. This serves as a conservative threshold, significantly larger than the robot's repetitive positioning error of ±0.03 mm (Table 3), to accommodate potential calibration uncertainties. This demonstrates that even in regions with complex mold surface features, the planning algorithm effectively avoids

collision risks. Simultaneously, the inverse kinematics (IK) success rate reached 100%, and idle-motion paths account for only 1.98% of total path length. The above data strongly suggest that the region partitioning strategy and attributed adjacency graph proposed in this paper effectively generate continuous trajectories, avoiding singularity locking and frequent jumps commonly seen in conventional methods [36].

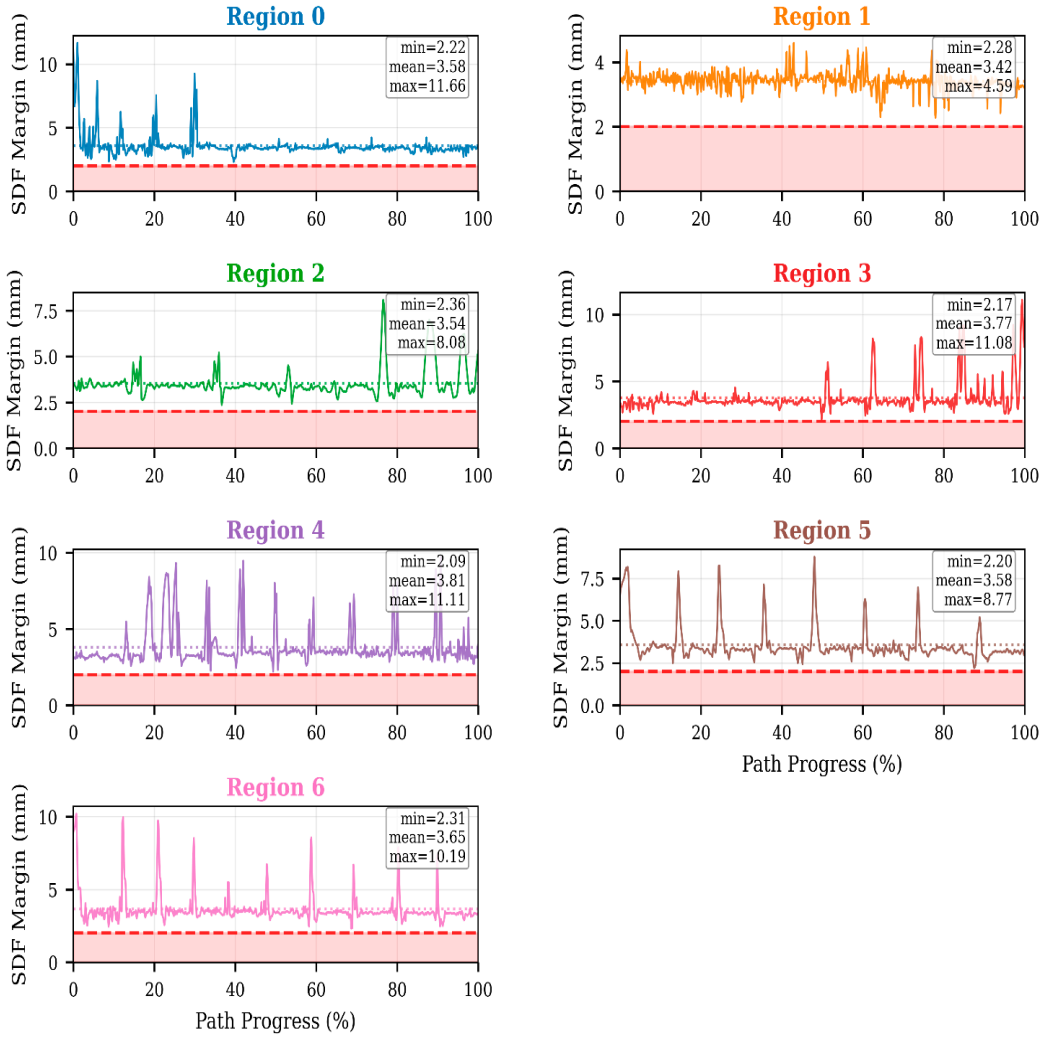


Figure 15. SDF safety margin evolution curves for multiple sub-regions. The red dashed line and shaded area represent the safety margin threshold (d_{safe} = 2.0 mm), and the dotted line indicates the mean SDF margin.

Table 4. Trajectory performance and coverage metrics.

Metric	Value
Path Point Evaluation Baseline	9283
IK Success Rate	100% (9283/9283)
Global Minimum Safety Margin	2.09 MM
Idle-Motion Path Ratio	1.98%
Global Coverage Rate	99.42%
Global Over-Polishing Rate	0.50%
Average Coverage Magnitude	9.996
Sub-Region Coverage Range	98.77–100.00%
Maximum Regional Over-Polishing	3.45%

Second, after completing the trajectory executability assessment, the process quality of the executed trajectory must be evaluated. Polishing quality depends largely on the consistency between the tool axis and the workpiece surface normal. To verify the effectiveness of this paper's method, a comparison was made with the conventional method of PCA principal direction combined with fixed-step reciprocal motion. Figure 16 shows the normal deviation field heatmaps of trajectories generated by the two methods in the workpiece coordinate system.

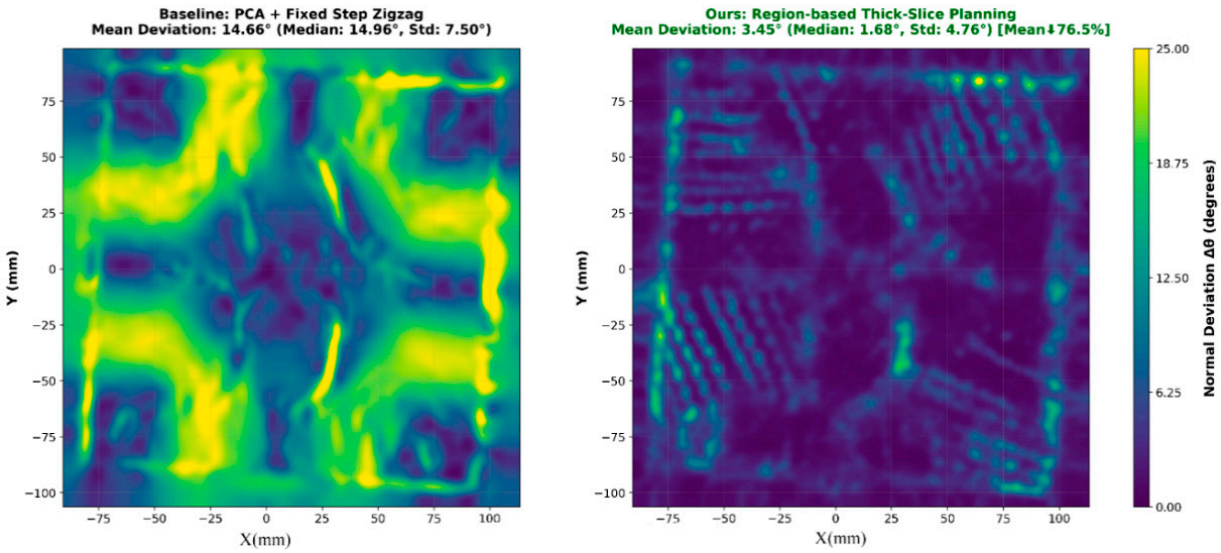


Figure 16. TCP orientation error heatmaps: proposed vs. conventional PCA-based method.

As can be seen from the left side of Figure 16, the baseline method exhibits pose mismatch in regions with large curvature variations (bright yellow areas). Its mean normal deviation reaches 14.66°, with a standard deviation of 7.50°. Such a large deviation can easily lead to over-polishing or under-polishing in practical machining. In contrast, the normal deviation shown on the right side of Figure 16 is significantly reduced and more uniformly distributed. The statistical results (Table 5) indicate that the mean normal deviation of our method decreases to 3.45°, and the median is only 1.68°. This improvement is mainly attributed to the high-dimensional perception capability of the deep feature fusion network proposed in Section 4.1 for complex topology, which enables the manipulator end-effector pose to closely conform to geometric variations in the curved surface. The high geometric consistency achieved in simulation provides a theoretical foundation for high-quality surface finishing in subsequent physical experiments.

Table 5. Comparative performance metrics: baseline method versus proposed method.

Metric	Baseline Method	Proposed Method
IK Success Rate	92%	100%
Min. Safety Margin	1.33 mm	2.09 mm
Mean Deviation	14.66	3.45
Median Deviation	14.96	1.68
95th Percentile	25.04	13.01
Std. Deviation	7.50	4.76
Coverage Rate	100.00%	99.42%
Avg. Coverage	7.75	9.996

Notes: Metrics are computed over $N = 9283$ discrete trajectory samples along the planned toolpaths in simulation.

Table 5 provides an explicit comparison against the conventional PCA-based planning baseline. Compared to the baseline, the proposed framework improves IK executability and increases the minimum SDF safety margin, indicating better kinematic feasibility and safer motion in the tested scenario (Table 5). Moreover, the proposed method substantially reduces TCP normal deviation (mean $14.66^\circ \rightarrow 3.45^\circ$, i.e., an $\approx$76% reduction), which is particularly important on high-curvature regions where pose mismatch may cause non-uniform material removal (Table 5). These results show that the contribution of our approach goes beyond achieving a low roughness value by providing a kinematically feasible and pose-consistent trajectory generation pipeline under CAD-free point-cloud inputs.

5.2. Mold Surface Polishing Experimental Verification

Building on the theoretical results from simulation-based verification, this section further validates the engineering effectiveness of the planned trajectories through experiments. The experiments were conducted on a polishing platform, as shown in Figure 17, where the manipulator and vision sensor configuration is consistent with the simulation environment. Surface roughness (Ra) was measured using a portable stylus profilometer (Mitutoyo Surftest SJ-210, Mitutoyo Corporation, Kawasaki, Japan). The instrument provides a display resolution of 0.002 μm, and measurements were conducted following the manufacturer's specified accuracy and repeatability.

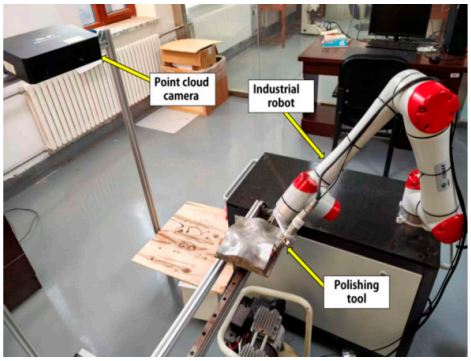


Figure 17. Experimental scene schematic diagram.

We selected one typical test point (P1–P7, see Figure 18) in each of the seven different regions of the workpiece, and measured surface roughness before and after polishing at each point three times. As shown in Table 6, the experimental data demonstrate that after polishing with the trajectory planned by our method, the average Ra value of the workpiece surface decreased from the initial 7.767 ± 0.493 μm to 0.607 ± 0.108 μm. The roughness at all seven test points was reduced to the sub-micrometer level, and the surface displayed uniform metallic luster.

Table 6. Ra changes before and after polishing at experimental points P1–P7.

Point	Before (Mean ± Std) (μm)	After (Mean ± Std) (μm)	ΔRa (μm)
P1	8.091 ± 0.529	0.669 ± 0.061	7.422
P2	8.078 ± 0.365	0.611 ± 0.086	7.467
P3	7.414 ± 0.528	0.626 ± 0.116	6.788
P4	7.575 ± 0.689	0.595 ± 0.125	6.980
P5	7.647 ± 0.653	0.529 ± 0.103	7.118
P6	7.783 ± 0.361	0.647 ± 0.135	7.137
P7	7.674 ± 0.481	0.561 ± 0.115	7.113

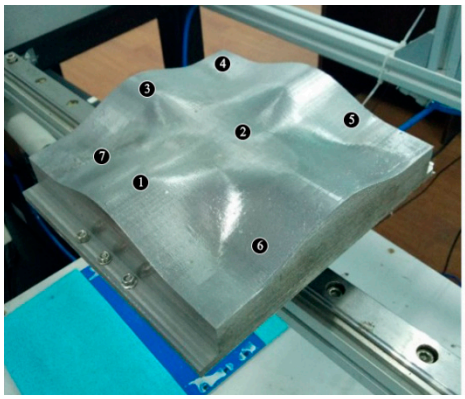


Figure 18. Mold surface workpiece.

The improvement trend in Ra values in the physical experiments is highly consistent with the normal deviation distribution in the simulation of Section 5.1. Regions where the normal deviation is smallest and pose alignment is best in simulation generally achieved better surface finish in the physical experiments. Notably, the standard deviation of Ra values decreased significantly, validating the effectiveness of our method in solving the problem of poor processing consistency in complex regions.

We note that a significant roughness reduction can also be observed in conventional polishing processes. Therefore, in addition to reporting the absolute Ra improvement, we emphasize the consistency of the achieved finish across different surface zones. As shown in Table 6, the post-polishing Ra values across the seven representative regions (P1–P7) fall within a narrow range of 0.529–0.669 μm, with an overall average of 0.607 ± 0.108 μm (Table 6). This zone-to-zone consistency supports the practical value of the proposed method for standardized finishing on complex mold surfaces. While manual polishing relies heavily on operator skill and is susceptible to fatigue-induced inconsistencies, especially on large-scale molds with limited reachability, the proposed robotic framework ensures deterministic quality (low standard deviation) and precise kinematic control (low normal deviation) suitable for industrial mass production. Furthermore, the proposed D-C-M strategy enables rapid foreground-extraction preprocessing (1.14 s for 237,640 points; Table 1), substantially reducing the preprocessing overhead prior to downstream partitioning and toolpath generation.

The experimental results demonstrate that, even in unstructured environments lacking CAD models and subject to perception noise, our method still achieves substantial improvements in finishing quality and consistency. This supports the fundamental advantages of the task-aware regional segmentation and deep geometric feature fusion strategy in addressing the challenging problem of point-cloud-driven polishing of complex surfaces.

6. Conclusions and Future Work

This study addresses the three key limitations identified in the Introduction for point-cloud-driven robotic polishing of complex mold surfaces: (i) the efficiency bottleneck in processing large-scale raw point clouds, (ii) task-agnostic region organization that ignores kinematic feasibility during planning, and (iii) the lack of closed-loop feasibility verification beyond purely geometric trajectories. To this end, we propose a systematic framework that bridges raw measurement data and executable robot trajectories by integrating point-cloud preprocessing, task-aware region partitioning with kinematic constraints, and data-driven trajectory generation.

Comprehensive validation via simulation and physical experiments demonstrates that the proposed framework improves efficiency, executability, and process quality. For effi-

ciency, the D-C-M strategy achieves a 7.5× preprocessing speedup with a 98.5% mapping-back success rate, reducing the preprocessing overhead prior to downstream planning. For task organization and kinematic feasibility, incorporating reachability constraints enables 100% inverse-kinematics solvability and maintains a larger minimum safety margin compared with a PCA-based baseline. In terms of trajectory quality, the mean TCP normal deviation is reduced from 14.66° to 3.45°, mitigating orientation mismatch on complex curvature regions. Physical polishing results further confirm stable sub-micron finishing quality with low zone-to-zone variation across P1–P7.

Notwithstanding the favorable experimental outcomes achieved in mold surface polishing, the proposed method presents certain limitations. Despite employing boundary-shrinking and buffer zone strategies to manage complex edges, an over-polishing rate of approximately 3.45% is still observed in overlapping regions. Additionally, the current deep learning prediction model exhibits a dependency on specific mold geometric features.

Consequently, future research will be concentrated on two primary avenues: First, the integration of a force-feedback-driven adaptive dwell time control algorithm is planned to further improve material removal consistency at boundary regions. Second, we aim to construct a comprehensive industrial point cloud dataset encompassing diverse topological structures and material properties. By leveraging transfer learning, we seek to enhance the model's generalization adaptability to unseen workpieces with irregular geometries, thereby facilitating the deployment of this technology in a broader spectrum of smart manufacturing applications. Furthermore, inspired by the complex interaction between surface texture and fluid dynamics, future work will also explore the integration of Moldflow analysis data into the trajectory planning stage. Specifically, aligning the robotic polishing direction with the plastic material flow direction could significantly reduce flow resistance and enhance the demolding performance of the injection molds. Establishing a data interface between the planning algorithm and Moldflow software will be a key focus of our next research phase.

Author Contributions: Conceptualization, M.Y. and X.L.; writing—original draft, X.L.; formal analysis, B.H. and Z.P.; data curation, Z.P. and B.H.; writing—review and editing, X.L. and M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Jilin Province, grant number 20240101095JC.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: All authors have no conflicts of interest.

Appendix A

Table A1. Summary of key hyperparameters and their selection rationale.

Algorithm Stage	Parameter (Symbol)	Value	Selection Rationale
Preprocessing	Outlier Sensitivity (α)	2.0	Standard statistical threshold ($\approx 2\sigma$) assuming approximately Gaussian measurement noise; removes sparse outliers while retaining the majority of inliers.
	Mapping Threshold (δ_{map})	Adaptive	Computed as $min(\delta_{map},\ c \cdot v_0)$ to balance label diffusion with boundary precision during mapping-back.

Table A1. *Cont.*

Algorithm Stage	Parameter (Symbol)	Value	Selection Rationale
Partitioning	Seed Constraint (c_g, c_s)	1.5/2.0	Fixed empirically to control seed density relative to the local feature scale and maintain stable supervoxel growth across datasets.
	Graph Weight Balance (γ)	0.5	Selected via sensitivity analysis (Pareto frontier in Figure 5) to balance kinematic connectivity and geometric adherence.
	Similarity Decay (λ)	10.0	Controls the steepness of feature-similarity decay; set empirically to obtain clear separation between intra-region and inter-region affinities.
	Merge Threshold (τ_{energy})	-1.0×10^{-4}	Small negative threshold to enforce strictly monotonic energy reduction and suppress numerical noise-induced marginal merges.
Path Planning	Loss Weights (λ_{dir}, λ_k)	Tuned	Tuned to balance convergence rates of the direction branch (cosine loss) and the step-distance branch (MSE).
	ROI Inset Ratio (α_{inset})	0.5	Set to $0.5 \cdot s_{base}$ to ensure the tool center remains strictly inside the boundary during path generation.
Validation	Safety Margin (d_{safe})	2.0 mm	Conservative threshold set significantly higher than the robot repeatability ($\pm$0.03 mm) to ensure safe operation under perception uncertainty.

References

1. Zhu, D.; Feng, X.; Xu, X.; Yang, Z.; Li, W.; Yan, S.; Ding, H. Robotic Grinding of Complex Components: A Step towards Efficient and Intelligent Machining—Challenges, Solutions, and Applications. *Robot. Comput.-Integr. Manuf.* **2020**, *65*, 101908. [CrossRef]
2. Márquez, J.J.; Pérez, J.M.; Ríos, J.; Vizán, A. Process Modeling for Robotic Polishing. *J. Mater. Process. Technol.* **2005**, *159*, 69–82. [CrossRef]
3. Ke, X.; Yu, Y.; Li, K.; Wang, T.; Zhong, B.; Wang, Z.; Kong, L.; Guo, J.; Huang, L.; Idir, M.; et al. Review on Robot-Assisted Polishing: Status and Future Trends. *Robot. Comput.-Integr. Manuf.* **2023**, *80*, 102482. [CrossRef]
4. Peta, K.; Wiśniewski, M.; Kotarski, M.; Ciszak, O. Comparison of Single-Arm and Dual-Arm Collaborative Robots in Precision Assembly. *Appl. Sci.* **2025**, *15*, 2976. [CrossRef]
5. Gong, L.; Li, L.; Zhang, L.; Fan, C.; Li, N. Uniform Coverage Trajectory Planning for Polishing of Equation-Free Surfaces. *Machines* **2025**, *13*, 568. [CrossRef]
6. Wang, K.; Ding, L.; Dailami, F.; Matthews, J. A Methodology to Support Robotic Polishing of Molds Integrated with CAD/CAM. *J. Adv. Manuf. Syst.* **2024**, *23*, 349–390. [CrossRef]
7. Jamshidi, H.; Budak, E. An Analytical Grinding Force Model Based on Individual Grit Interaction. *J. Mater. Process. Technol.* **2020**, *283*, 116700. [CrossRef]
8. Shan, Z.; Cai, Z.; Chen, T.; Chen, Z.; Hu, Y. A Trajectory Planning Technique for Grinding and Polishing Aerospace Turbine Blades' Thermal Barrier Coatings Based on PCL Point Cloud Processing. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2025**, *239*, 1565–1579. [CrossRef]
9. Wang, Z.; Zou, L.; Li, J.; Zhang, J.; Wang, W. Point-Driven Toolpath Curve and Orientation Smoothing in Robotic Belt Grinding for Turbine Blade. *Robot. Comput.-Integr. Manuf.* **2025**, *96*, 103046. [CrossRef]
10. Xiao, M.; Ding, Y.; Yang, G. A Model-Based Trajectory Planning Method for Robotic Polishing of Complex Surfaces. *IEEE Trans. Automat. Sci. Eng.* **2022**, *19*, 2890–2903. [CrossRef]
11. Ren, X.; Chen, G.; Wang, Z.; Wang, Z.; Sun, L. Polishing Path Planning Based on Point Cloud. In Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence, Xiamen, China, 8–11 May 2020; pp. 222–228.

12. Zeng, Z.; Jiang, C.; Ding, S.; Li, Q.; Zhai, Z.; Chen, D. Trajectory Planning of Shape-Following Laser Cleaning Robot for the Aircraft Radar Radome Coating. *Appl. Sci.* **2024**, *14*, 1163. [CrossRef]
13. Rusu, R.B.; Cousins, S. 3D Is Here: Point Cloud Library (PCL). In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011*; IEEE: New York, NY, USA, 2011; pp. 1–4.
14. Zhou, N.; Jiang, P.; Jiang, S.; Shu, L.; Ni, X.; Zhong, L. An Identification and Localization Method for 3D Workpiece Welds Based on the DBSCAN Point Cloud Clustering Algorithm. *J. Manuf. Mater. Process.* **2024**, *8*, 287. [CrossRef]
15. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
16. Zeng, X.; Zhu, G.; Gao, Z.; Ji, R.; Ansari, J.; Lu, C. Surface Polishing by Industrial Robots: A Review. *Int. J. Adv. Manuf. Technol.* **2023**, *125*, 3981–4012. [CrossRef]
17. Papon, J.; Abramov, A.; Schoeler, M.; Worgotter, F. Voxel Cloud Connectivity Segmentation—Supervoxels for Point Clouds. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2027–2034.
18. Lai, B.; Yuan, Y.; Zhang, Y.; Hu, B.; Yu, Q. An Accurate and Efficient Supervoxel Re-Segmentation Approach for Large-Scale Point Clouds Using Plane Constraints. *Remote Sens.* **2023**, *15*, 3973. [CrossRef]
19. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vis.* **2004**, *59*, 167–181. [CrossRef]
20. Yuan, Y.; Wang, J.; Li, W.; Wang, K.; Rao, H.; Xu, J. Fast Supervoxel Segmentation of Connectivity Median Simulation Based on Manhattan Distance. *Int. J. Appl. Earth Obs. Geoinf.* **2024**, *133*, 104108. [CrossRef]
21. Guo, W.; Huang, X.; Qi, B.; Ren, X.; Chen, H.; Chen, X. Vision-Guided Path Planning and Joint Configuration Optimization for Robot Grinding of Spatial Surface Weld Beads via Point Cloud. *Adv. Eng. Inform.* **2024**, *61*, 102465. [CrossRef]
22. Han, Y.; Zhang, L.; Guo, M.; Fan, C.; Liang, F. Tool Paths Generation Strategy for Polishing of Freeform Surface with Physically Uniform Coverage. *Int. J. Adv. Manuf. Technol.* **2018**, *95*, 2125–2144. [CrossRef]
23. Li, Q.; Feng, H.; Shi, K.; Gao, Y.; Fang, Y.; Liu, Y.-S.; Han, Z. SHS-Net: Learning Signed Hyper Surfaces for Oriented Normal Estimation of Point Clouds. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 13591–13600.
24. Song, X.; An, J.; Ma, X. Study on Trajectory Planning for Polishing Free-Form Surfaces of XY-3-RPS Hybrid Robot. *Actuators* **2025**, *14*, 269. [CrossRef]
25. Guerrero, P.; Kleiman, Y.; Ovsjanikov, M.; Mitra, N.J. PCPN et Learning Local Shape Properties from Raw Point Clouds. *Comput. Graph. Forum* **2018**, *37*, 75–85. [CrossRef]
26. Rososhansky, M.; Xi, F.J. Coverage Based Tool-Path Planning for Automated Polishing Using Contact Mechanics Theory. *J. Manuf. Syst.* **2011**, *30*, 144–153. [CrossRef]
27. Wang, Z.; Zou, L.; Zhang, J.; Li, H.; Wang, W.; Huang, Y. Tool Axis Vector Optimization for Robotic Grinding Based on Measured Point Cloud of Complex Curved Blade. *Adv. Eng. Inform.* **2024**, *62*, 102716. [CrossRef]
28. Sun, Y.; Jia, J.; Xu, J.; Chen, M.; Niu, J. Path, Feedrate and Trajectory Planning for Free-Form Surface Machining: A State-of-the-Art Review. *Chin. J. Aeronaut.* **2022**, *35*, 12–29. [CrossRef]
29. Tafuro, A.; Ghalloub, C.; Zanchettin, A.M.; Rocco, P. Autonomous Robotic Polishing of Free-Form Poly-Surfaces: Planning from Scanning in Realistic Industrial Setting. *Procedia Comput. Sci.* **2024**, *232*, 2488–2497. [CrossRef]
30. Wen, Y.; Jaeger, D.J.; Pagilla, P.R. Uniform Coverage Tool Path Generation for Robotic Surface Finishing of Curved Surfaces. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4931–4938. [CrossRef]
31. Lu, L.; Zhang, L.; Fan, C.; Wang, H. High-Order Joint-Smooth Trajectory Planning Method Considering Tool-Orientation Constraints and Singularity Avoidance for Robot Surface Machining. *J. Manuf. Process.* **2022**, *80*, 789–804. [CrossRef]
32. Nieto Bastida, S.; Lin, C.-Y. Autonomous Trajectory Planning for Spray Painting on Complex Surfaces Based on a Point Cloud Model. *Sensors* **2023**, *23*, 9634. [CrossRef]
33. Yang, F.; Cai, Z.; Chen, Y.; Dong, S.; Deng, C.; Niu, S.; Zeng, W.; Wen, S. A Robotic Polishing Trajectory Planning Method Combining Reverse Engineering and Finite Element Mesh Technology for Aero-Engine Turbine Blade TBCs. *J. Therm. Spray Technol.* **2022**, *31*, 2050–2067. [CrossRef]
34. Chitta, S.; Sucan, I.; Cousins, S. MoveIt! [ROS Topics]. *IEEE Robot. Automat. Mag.* **2012**, *19*, 18–19. [CrossRef]
35. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
36. Li, J.; Zou, L.; Luo, G.; Wang, W.; Lv, C. Enhancement and Evaluation in Path Accuracy of Industrial Robot for Complex Surface Grinding. *Robot. Comput.-Integr. Manuf.* **2023**, *81*, 102521. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.