Geometry-Aware Coverage Path Planning for Depowdering on Complex 3D Surfaces

Van-Thach Do and Quang-Cuong Pham

***Abstract*—This letter presents a new approach to obtaining nearly complete coverage paths (CP) with low overlapping on 3D general surfaces using mesh models. The CP is obtained by segmenting the mesh model into a given number of clusters using constrained centroidal Voronoi tessellation (CCVT) and finding the shortest path from cluster centroids using the geodesic metric efficiently. We introduce a new cost function to harmoniously achieve uniform areas of the obtained clusters and a restriction on the variation of triangle normals during the construction of CCVTs. The obtained clusters can be used to construct high-quality viewpoints (VP) for visual coverage tasks. Here, we utilize the planned VPs as cleaning configurations to perform residual powder removal in additive manufacturing using manipulator robots. The self-occlusion of VPs and ensuring collision-free robot configurations are addressed by integrating a proposed optimization-based strategy to find a set of candidate rays for each VP into the motion planning phase. CP planning benchmarks and physical experiments are conducted to demonstrate the effectiveness of the proposed approach. We show that our approach can compute the CPs and VPs of various mesh models with a massive number of triangles within a reasonable time.**

***Index Terms*—Additive manufacturing, industrial robots, motion and path planning, robotics in hazardous fields.**

I. Introduction

COVERAGE path planning (CPP) is the problem of computing a path that traverses all points in a given domain. Several criteria for a CPP include complete coverage, no overlap, and satisfying task-based additional requirements [1]. Designing a CPP algorithm to satisfy all those criteria is challenging and may not be achieved in practice. Thus, there will be a tradeoff in selecting the priorities among those criteria to meet the quality of a specific task.

Numerous previous CPP approaches aim to generate polylines with a nearly constant swath width on the target surface to ensure complete and consistent coverage. The coverage paths can be utilized for various tasks, such as indoor cleaning, surveillance, agriculture services, etc. Additionally, CPP plays a pivotal role in perception-based surface treatments using robotic end-effectors, including inspections and 3D reconstructions, where the quality of the selected viewpoints (VP) significantly affects the performance. Most existing methods for selecting VPs either randomly generate a set of candidate VPs around the object or choose them based on the normal direction of each triangle face of the mesh [2]. While these methods may perform well for simple meshes, they can lead to inadequate coverage of complex geometries and high overlap and may increase computation time if the number of sample VPs is high. Furthermore, because these methods do not account for self-collision or collisions between the robot and the surrounding objects, they cannot ensure that visual sensors reach the object surface.

In contrast to prior approaches, this study presents a new CPP algorithm that can attain nearly complete coverage and low overlapping on the object surface while producing high-quality VPs for cleaning, model-based inspection, or 3D reconstruction tasks. We also address the self-occlusion (occlusion between VPs and other triangle faces in the mesh) and the collision between the robot body and the surrounding environment. Then, we leverage the obtained VPs for an industrial cleaning application using manipulator robots. Our approach consists of three main contributions as follows:

i) We present a new energy function for the constrained centroidal Voronoi tessellation (CCVT) method [3] to segment the mesh model $\mathbb{M}$ into m clusters with low standard deviations (SD) in both area and triangle normal. VPs are constructed based on the mass centroids and proxy normals [4] (average triangle normals) of the resulting clusters.

ii) We propose an efficient approach, namely geodesic decomposition calculation, that has low time complexity to calculate the exact geodesic distances (GD) between centroids. Then, the coverage path on the surface is computed based on the obtained distance costs between centroids.

iii) Compared to prior works, we address both issues that may be encountered when positioning the robot tool at the obtained VPs: (1) self-occlusion between rays at VPs and other triangles in $\mathbb{M}$; and (2) infeasible robot configurations at computed poses due to joint limits or collisions with surrounding objects (rays at VPs in these cases are referred to as invalid rays). An efficient correction algorithm is proposed to obtain valid robot configurations with the closest direction (if possible) to the invalid rays by utilizing a set of candidate rays formed by an

Manuscript received 5 March 2023; accepted 10 July 2023. Date of publication 19 July 2023; date of current version 25 July 2023. This letter was recommended for publication by Associate Editor Z. Zhang and Editor X. Liu upon evaluation of the reviewers' comments. This work was supported in part by Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) under Grant RIE2020, in part by Funding Initiative, and in part by the cash and in-kind contribution from the industry partner, HP Inc., through the HP-NTU Digital Manufacturing Corporate Lab. *(Corresponding author: Van-Thach Do.)*

The authors are with HP-NTU Digital Manufacturing Corporate Lab and School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798 (e-mail: thach.do@ntu.edu.sg; cuong@ntu.edu.sg).

Digital Object Identifier 10.1109/LRA.2023.3296943

2377-3766 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

optimization strategy. We apply the proposed method to automate the removal of residual powder from 3D-printed (3DP) parts with complex geometries, which, to the best of our knowledge, has not been done before.

A. Related Work

The CPP problem has been widely studied and applied to various robotic applications [5], [6], [7], [8], [9], where most of the target surfaces are flat or have low curvature geometries. In [10], a global surface parameterization-based CPP is proposed to ensure non-intersecting paths with great coverage for general surfaces with complex topology. By considering minimizing energy consumption caused by gravity, a CPP algorithm is proposed to plan the Fermat spiral paths for mobile robots on general terrain surfaces [11]. In those approaches, the CPP results in nearly complete coverage paths on the target surface.

CPP-based surface cleaning solutions using robot manipulators have been widely developed. In [12], [13], surfaces cleaning applications are proposed, where the coverage paths are generated by projecting from 2D to 3D and obtaining from intersecting with equally spaced parallel planes, respectively. However, the quality of the generated coverage paths may degrade on high-curvature surfaces due to distortion. In [14], the problem of minimizing the cost function in the joint space for surface cleaning tasks is addressed. Here, the object surface is modeled as a set of planar patches. However, the obtained robot configurations may yield high overlapping rates due to randomly picking points from the acquired point cloud during the construction of surface patches. Also, the issues of self-occlusion and infeasible configurations due to joint limits are not addressed. In [15], a new method is introduced to generate a uv grid on three types of 3D freeform surfaces formed by non self-intersecting freeform curves. The method aims to create even coverage paths while considering task constraints on the end-effector pose. However, this approach necessitates human operators to provide transformed axes, and the resulting task constraints may not be suitable for high curvature surfaces, where the self-occlusion phenomenon exists.

In 3D inspection and reconstruction tasks, VP planning plays a crucial role in determining the optimal sets of sensor poses. The optimal set refers to the minimum number of VPs required to capture the target objects or scenes at a high coverage rate. In model-based methods, the VPs can be obtained in different ways. VPs are randomly generated [16], [17]. For high-curvature meshes, a large number of VP samples is required to ensure complete coverage. In [18], [19], VPs are sampled from mesh vertices. This approach suffers from VP redundancy. Another VP generation method is patch sampling [20], which involves segmenting the mesh model into sub-regions under specified constraints. The most relevant to our work is [21], where the mesh model is segmented using B-splines and guided by a set of feature functionals. Due to its independence from mesh resolution, this approach can lead to a reduction in the number of VPs compared to random sampling methods. However, the computed VPs in the *subdivided segments* step heavily rely on processing B-spline surfaces. This may result in a large number of redundant VPs at intersections of high-curvature B-spline surfaces. Also, the self-occlusion of VPs is not handled in that study. In [22], a targeted VP sampling strategy is proposed to find the optimal next VP. This is done by iteratively reformulating and solving the search problem, aiming to minimize the number of VPs and the travel cost while ensuring robot kinematics and avoiding collisions. However, the iterative search and sampling of every triangle face can result in a long computation time for high-resolution meshes. Our proposed method, which segments the mesh globally and derives VPs from the obtained clusters, benefits from the efficient computation of patch sampling while achieving nearly uniform areas of the obtained clusters for low curvature meshes. Additionally, we address the self-occlusion of VPs and compute an optimal set of collision-free robot configurations for performing the task.

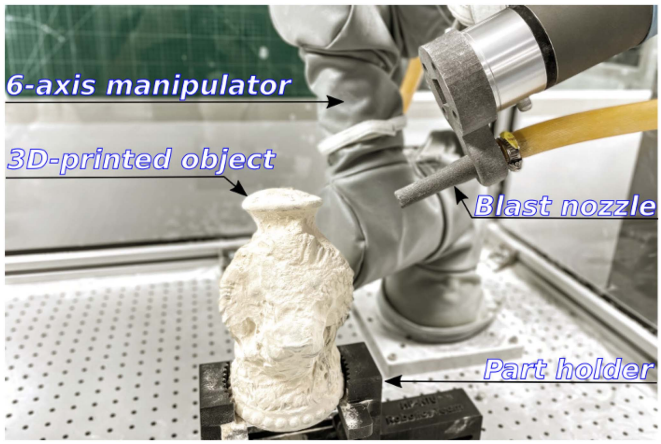

Fig. 1. Experimental setup of a depowdering system for 3D-printed objects.

Additive manufacturing is popular due to its ability to fabricate complex objects rapidly from CAD models. However, post-processing 3DP parts, such as support and powder removal and surface finishing, is time-consuming and expensive and poses potential hazards to workers. To assist and automate these processes, vision systems have been proposed to identify 6D poses of the 3DP parts under partly occluded by powder [23], [24], [25]. In [24], a full pipeline is proposed to perform residual powder removal. However, the designed mechanism aims to clean flat or low-curvature objects. In [25], a vision-based approach is proposed to perform depowdering for 3DP parts located in a powder bed. However, replacing actual powder materials, which are highly adhesive, with children's play sand simplifies this work compared to real-world scenarios. In their work, the roll and pitch motions of the robot tool are maintained during the operation, limiting the ability to handle complex objects. Their future work, i.e., investigating an advanced path planning algorithm to improve the depowdering automation, is addressed in our study.

B. Automated Depowdering Platform for 3DP Parts

Fig. 1 shows the experimental cleaning system for 3DP parts, which consists of a 6-axis robotic manipulator, a blast nozzle tool, a part holder, and a 3DP part. The blast nozzle is attached

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

to the robot end-effector, and a high-pressure compressed air stream containing tiny glass beads flows through the nozzle head to remove residual powder. Our goal is to propose an efficient CPP algorithm with collision-free robot trajectories to automate the removal of residual powder from 3DP parts after unpacking from 3D printing stations.

II. Method

A. Computation of Low Curvature Clusters Using CCVT Method

Let $\mathbb{M} = \{\mathbb{F}, \mathbb{N}\}$ be the input mesh, where $\mathbb{F}$ and $\mathbb{N}$ are the set of triangle faces and their corresponding normal vectors, respectively. Our objective is to divide $\mathbb{M}$ into m connected, low curvature clusters of faces $\{\mathbb{V}_i\}_{i=1}^m$ by computing discrete CCVT on $\mathbb{M}$ with triangle normal constraints.

Definition 1: Discrete Voronoi tessellation. Given a discrete set of points $\mathbb{W} = \{\mathbf{y}_i\}_{i=1}^{n_t} \subset \Re^N$, where $\mathbf{y}_i$ are centroids of n_t faces in $\mathbb{F}$. A set $\{\mathbb{V}_i\}_{i=1}^m$ is a tessellation of $\mathbb{W}$ if $\mathbb{V}_i \cap \mathbb{V}_j = \emptyset, \forall i \neq j$ and $\cup_{i=1}^k \mathbb{V}_i = \mathbb{W}$. Voronoi sets are defined as

$$\mathbb{V}_i = \{\mathbf{y} \in \mathbb{W} | \|\mathbf{y} - \mathbf{z}_i\| < \|\mathbf{y} - \mathbf{z}_j\|, j = 1, \ldots, m, j \neq i, \tag{1}$$

where $\{\mathbf{z}_i\}_{i=1}^m \subset \Re^N$ are referred to as generators. $\mathbb{V}_i$ is referred to as a Voronoi tessellation or Voronoi diagram.

Definition 2: Discrete centroidal Voronoi tessellation. A centroidal Voronoi tessellation (CVT) is a Voronoi tessellation where $\{\mathbf{z}_i\}$ is also the mass centroid of its Voronoi region. By choosing a uniform density function, we can obtain $\{\mathbf{z}_i\}$ for the discrete form of CVT [26], [27] as

$$\mathbf{z}_i = \frac{\sum_{\tau \in \mathbb{V}_i} \zeta_c(\tau)\zeta_a(\tau)}{\sum_{\tau \in \mathbb{V}_i} \zeta_a(\tau)}, \tag{2}$$

where $\zeta_c(\tau)$ and $\zeta_a(\tau)$ are the functions to obtain the centroid and the area of the face τ, respectively. Centroidal Voronoi diagrams minimize the following energy function

$$E_{l2} = \sum_{i=1}^m \sum_{\tau \in \mathbb{V}_i} \left(\zeta_a(\tau)\|\zeta_c(\tau) - \mathbf{z}_i\|^2\right) \tag{3}$$

Definition 3: Discrete constrained centroidal Voronoi diagram. $\{\mathbb{V}_i\}_{i=1}^m$ is the CCVT of $\mathbb{W}$ if and only if $\{\mathbf{z}_i\}_{i=1}^m$ is the constrained mass centroid of those regions. The constrained centroid $\mathbf{z}_i^c$ of $\mathbb{V}_i$ on a continuous compact set $\mathbf{S} \subset \Re^N$ is determined as the projection of $\mathbf{z}_i$ onto faces in $\mathbf{S}$ along the proxy normal of $\mathbb{V}_i$. For a a discrete set of face centroids of $\mathbb{M}$, this projection is approximately determined as

$$\min_{\mathbf{z}_i^c \in \mathbb{V}_i} \left(\|\mathbf{z}_i - \mathbf{z}_i^c\|^2\right) \tag{4}$$

Various energy functions have been proposed for different applications. A $\mathcal{L}^{2,1}$ metric is proposed for capturing anisotropy and segmentation [28]. By combining metrics from spherical and hyperbolic spaces, a unified framework in universal covering space [29] is proposed to get uniform partitions and high-quality remeshing results. However, these approaches do not align with our scope, leading us to introduce a new energy function that combines distance and normal costs. In the CVT formulation for 3D surfaces, the GD is a natural choice [30], but its calculation can be computationally expensive, leading to the use of the Euclidean metric as an approximation. This approximation, however, may result in significant errors on high-curvature surfaces [26]. Compared to the ℓ^2 norm, the ℓ^1 norm is preferable in high dimensional spaces and has been efficiently utilized to approximate the GD in various fields [31], [32]. Motivated by this, we select the ℓ^1 metric for evaluating the distances on 3D surfaces in the distance cost. The normal cost is used to restrict the variation of triangle normals. The proposed energy function is formulated as follows:

$$E = \sum_{i=1}^m \sum_{\tau \in \mathbb{V}_i} \xi\left(\mathbf{z}_i, \tau\right) \tag{5}$$

where the cost function ξ is defined as

$$\xi\left(\mathbf{z}_i, \tau\right) = \left(\alpha_1^{-1}\alpha_2\zeta_a(\tau)\|\zeta_c(\tau) - \mathbf{z}_i\|_1 + \overline{\alpha}_2\zeta_a(\tau)\Upsilon_n\right), \tag{6}$$

where α_1 is a positive constant used to normalize the first term. It is determined based on the length of the diagonal bounding box of the mesh. α_2 is selected within the range of $\in [0, 1]$, $\overline{\alpha}_2 = 1 - \alpha_2$. Υ_n is the normal cost and is determined as

$$\Upsilon_n = \beta\left(\zeta_n(\tau), \zeta_n\left(\mathbf{z}_i\right)\right)\left(1 - \zeta_n(\tau) \cdot \zeta_n(\mathbf{z}_i)\right)/2 \tag{7}$$

where $\zeta_n(\tau)$ and $\zeta_n(\mathbf{z}_i)$ are the normal vector of face τ and the proxy normal of $\mathbb{V}_i$, respectively. The function β is selected as $\beta = 1$ if the dot product $\zeta_n(\tau) \cdot \zeta_n(\mathbf{z}_i) > \alpha_3$, and $\beta = \alpha_4$ otherwise. $\alpha_3 \in (-1, 1)$ and $\alpha_4 > 1$ are tuning constants. α_3 is the threshold at which a higher weight (α_4) is applied to the normal cost, aiming to remove and reorganize triangles from the assigned cluster if they exhibit large normal differences compared to the corresponding cluster's proxy normal. The CCVT is computed based on the trade-off between distance and normal costs, as described in (5). For meshes with high curvature surfaces, we mitigate the influence of the normal cost to avoid multiple components per cluster, thereby simplifying the post-processing step [26]. This can be achieved by increasing α_2 and decreasing α_3 and α_4. Conversely, for meshes with low curvature surfaces, we decrease α_2 and increase α_3 and α_4 to obtain low curvature clusters.

The CCVT is computed by the Lloyd algorithm as follows. (i) Randomly select m points $\mathbf{z}_i \in \mathbb{W}$. (ii) Update Voronoi regions: a face in $\mathbb{F}$, with the associated centroid $\{\mathbf{y}_v\}_{v=1}^{n_t}$, is assigned to $\mathbb{V}_q, q = 1, \ldots, m$ if

$$\mathbf{z}_q = \underset{\mathbf{z}_i \in \{\mathbf{z}_i\}_{i=1}^m}{\operatorname{argmin}} \ \xi\left(\mathbf{z}_i, \mathbf{y}_v\right) \tag{8}$$

(iii) Update the mass centroids of the computed CCVTs using (2). (iv) Update $\mathbf{z}_i$ to the constrained mass centroid of the CCVT $\mathbb{V}_i$ using (4). (v) Check if the convergence criteria are met; otherwise, repeat step (ii).

B. Determination of Coverage Path Using the Geodesic Metric

The cleaning task is performed by aligning the nozzle axis at m points located at $\mathbf{z}_i$, normal direction $-\zeta_n(\mathbf{z}_i)$, and the distance from the nozzle to $\mathbf{z}_i$ is r_s. The CPP of this task is to find the shortest path traversing all $\mathbf{z}_i$. The distances between $\mathbf{z}_i$ on the mesh $\mathbb{M}$ are computed using the geodesic metric. However, computing GDs on the entire $\mathbb{M}$ is computationally intensive and impractical, especially on high resolution meshes. To tackle this issue, we propose a new approach to decompose

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

the computing on the whole mesh into the computing on m subgraphs constructed by edges of each $\mathbb{V}_i$ and its neighbor-connected clusters as Algorithm 1, Lines 1-7. Here, `GetAdj`$(\cdot)$ and $\epsilon(\cdot)$ are used to get neighbor-connected clusters $(\cdot)$ and edges of the cluster $(\cdot)$, respectively. The exact GD and its path are computed using `ExactGeodesic` [33]. $\mathbb{C}_d$ and $\mathbb{P}_d$ are dictionary mappings containing cost and tour between generators obtained from computing the shortest path using Dijkstra's algorithm for multiple sources and destinations based on the graph $\mathcal{G}_g$. It is noted that Dijkstra's algorithm is only used to compute distances and paths of non-existing edges in $\mathbb{E}_g$. Lastly, the coverage path is determined by finding the shortest path to visit each generator $\mathbf{z}_i$ exactly once. This problem is formulated as the Traveling Salesman Problem (TSP), which is NP-hard. It is probably impossible to find optimal solutions in polynomial time [34]. Based on the obtained GD cost in $\mathbb{E}_g$, the 3-Opt algorithm (Algorithm 1, Line 15) is applied to find the near-optimal solution of the TSP in 3D space. As a result, we obtain a near-optimal tour of $\{\overline{\mathbf{z}}_i\}_{i=1}^m$ over $\mathbb{M}$ and the corresponding geodesic path, which represents the coverage path, extracted from $\mathcal{P}_{geo}$. The notations $\mathbb{X} \xleftarrow{+} x$ and $\mathbb{X} \leftarrow x$ represent appending the element x to the list $\mathbb{X}$ and assigning x to the list $\mathbb{X}$, respectively. $\mathbb{C}_d[i,j]$ returns the cost length from $\mathbf{z}_i$ to $\mathbf{z}_j$. $\mathbb{P}_d[i,j]$ is used to get geodesic path between $\mathbf{z}_i$ and $\mathbf{z}_j$.

Complexity: Assuming that the processed triangular mesh $\mathbb{M}$ is closed. The relation between the number of faces and edges n_e [35] is $n_e = 3/2n_t$. The complexity of an exact geodesic algorithm is $\mathcal{O}(n_e^2 log(n_e))$ [33]. To compute the coverage path, GDs between cluster centroids need to be computed (Algorithm 1, Lines 2-7). Assuming that each cluster has ϖ neighbors on average, our method achieves the complexity of $\mathcal{O}(m(\varpi/2)((\varpi+1)2n_e/(3\,m))^2 log((\varpi+1)2n_e/(3\,m))$ and requires $\mathcal{O}((\varpi+1)2n_e/(3\,m))$ space. Without using our proposed method, those distances are determined by computing on the entire edges of $\mathbb{M}$ with the complexity is $\mathcal{O}((m(m-1)/2)n_e^2 log(n_e))$ and requires $\mathcal{O}(n_e^2)$ space, which requires much larger space and is more computationally expensive compared to the proposed decomposition calculation, especially for high resolution meshes. The complexity for the rest computations are $\mathcal{O}(m^3)$(Line 8, Dijkstra) + $\mathcal{O}(m(m-1)/2)$ (Lines 9–14) + $\mathcal{O}(m^3)$(3Opt). Thus, the proposed CPP can be obtained within a reasonable time.

C. Correction of Infeasible Robot Configurations and Computation of Final Robot Trajectory

Let $\mathbf{r}_i = \{\overline{\mathbf{z}}_i, \zeta_n(\overline{\mathbf{z}}_i)\}, \zeta_n(\overline{\mathbf{z}}_i) = [n_{cx}, n_{cy}, n_{cz}]^T$, and $\mathbb{R}_{opt} = \{\mathbf{r}_i\}_1^m$. For the cleaning task, it is crucial that the ray $\mathbf{r}_i$ has no obstructions, enabling unobstructed airflow from the nozzle to $\overline{\mathbf{z}}_i$ (or unobstructed camera view for model-based inspection or 3D reconstruction tasks in regions of interest). However, this property cannot be guaranteed solely based on the obtained proxy normals $\zeta_n(\overline{\mathbf{z}}_i)$. Also, there exists a robot configuration, i.e., an inverse kinematics (IK) solution, ensuring the tool is aligned to $\mathbf{r}_i$ without colliding with the surrounding environment. We propose an optimization problem (see appendix A) to construct a set of candidate rays representing the possible directions of

Algorithm 1: Computation of a Coverage Path.

```
GetCoveragePath({V_i}_{i=1}^m, {z_i}_{i=1}^m)
1   P_geo = ∅; E_g = ∅;
2   for i = 1 to m do
3       V_a, z_a = GetAdj(V_i); E_p = ε(V_i ∪ V_a);
4       for z in z_a do
5           if (z_i, z_a) not in E_g then
6               c, p = ExactGeodesic(E_p, z_i, z_a);
7               E_g ←+ ((z_i, z_a), c); P_geo ←+ p;
            end
        end
    end
8   N_g = {z_i}_{i=1}^m; G_g = (N_g, E_g);
9   C_d, P_d = Dijkstra(G_g);
10  for i = 1 to m do
11      for j = i + 1 to m do
12          if (z_i, z_j) not in E_g then
13              E_g ←+ ((z_i, z_j), C_d[i, j]);
14              P_geo ←+ P_d[i, j];
            end
        end
    end
15  return 3Opt_GeoPath(E_g, P_geo);
```

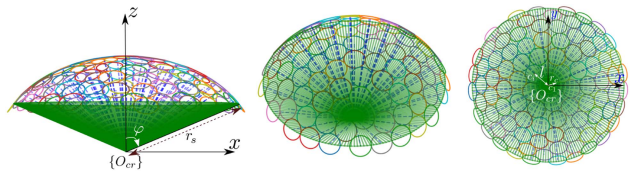

Fig. 2. Set of candidate rays in different views.

the nozzle at $\overline{\mathbf{z}}_i$ as shown in Fig. 2. The obtained result is a set $\mathbb{C}_{cr} = \{c_1, c_2, \ldots, c_{N_c}\}$ containing N_c center points of circles of radius r_c (nozzle radius). These circles are (almost) uniformly distributed in a specified region, defined by an elevation angle φ, of a sphere surface of radius r_s. Also, the returned centers of $\mathbb{C}_{cr}$ are then sorted in ascending order of distance to c_1. The set of near-optimal, collision-free IKs is computed using Algorithm 2. Those IKs are used to compute coverage trajectories to perform the cleaning task. Since the rotation about the ray is irrelevant for cleaning tasks, we can discretize the rotation about each ray into a set of angles Θ as

$$\Theta = \{2\pi(i-1)/I\}, i = 1, .., I, \tag{9}$$

where I is the discretization step size, thus, for each $\mathbf{r}_i$, there are I possible configurations. We use OpenRAVE [36] for computing IKs, checking collisions between the corresponding robot configurations and environment (`getIKs` and `isValidCfg`, Algorithm 2), and performing motion planning to obtain final robot trajectory. A valid robot configuration, i.e., `isValidCfg`$(\mathbf{r}, \theta_r)$ returns `True` if ray $\mathbf{r}$ has an elevation angle less than θ_r and the obtained configuration is collision-free with the surroundings. Trimesh [37] library is used to check self-collision $\mathbf{r}_i \cap \mathbb{F}$, Algorithm 2. A ray $\mathbf{r}_x$ is considered invalid

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Determination of Near-Optimal, Valid, and Collision-Free Robot Configurations.

GetOptRobotCfgs$(\mathbb{R}_{opt}, \mathbb{C}_{cr})$
1 $\mathcal{C}_{free} = \emptyset$;
2 **for** $i = 1$ **to** m **do**
3 **if** $\mathbf{r}_i \cap \mathbb{F} == \emptyset$ **and** **isValidCfg**$(\mathbf{r}_i, \theta_r)$ **then**
4 $\mathcal{C}_{free} \xleftarrow{+}$ **getIKs**$(\mathbf{r}_i)$;
else
$\mathbf{r}_t =$ **GetFreeRay**$(\mathbf{r}_i, \mathbb{C}_{cr})$;
5 **if** $\mathbf{r}_t \neq$ None **then**
6 $\mathcal{C}_{free} \xleftarrow{+}$ **getIKs**$(\mathbf{r}_t)$;
end
end
7 **return** **ComputeOptCfg**$(\mathcal{C}_{free})$;
GetFreeRay$(\mathbf{r}_x, \mathbb{C}_{cr})$
8 $\overline{\mathbb{C}}_{cr} =$ **AlignCR**$(\mathbf{r}_x, \mathbb{C}_{cr})$
9 **for** $k = 1$ **to** N_c **do**
10 $o_k = \overline{\mathbb{C}}_{cr}^{[k]}$; $\mathbf{r}_c = \{\overline{\mathbf{z}}_x, \overrightarrow{\overline{\mathbf{z}}_x o_k}\}$;
11 **if** $\mathbf{r}_c \cap \mathbb{F} == \emptyset$ **and** **isValidCfg**$(\mathbf{r}_c, \theta_r)$ **then**
12 **return** $\mathbf{r}_c$;
end
end
13 **return** None;
AlignCR$(\mathbf{r}_x, \mathbb{C}_{cr})$
14 $\phi_y = \arccos(n_{cz})$; $\phi_z = \arctan2(n_{cy}, n_{cx})$;
15 $\mathbb{C}_{tmp} \leftarrow \emptyset$; $\mathcal{T}_a = \mathcal{T}(\overline{\mathbf{z}}_x)\mathcal{R}_z(\phi_z)\mathcal{R}_y(\phi_y)$;
16 **for** $k = 1$ **to** N_c **do**
17 $\mathbb{C}_{tmp} \xleftarrow{+} \mathcal{T}_a \mathbb{C}_{cr}^{[k]}$;
end
18 **return** $\mathbb{C}_{tmp}$;

if it either collides with the mesh or has no IK solution aligned with that ray. An invalid ray is corrected (if possible) by using `getIKs` and aligning the candidate rays in `AlignCR` to new poses such that O_{cr} is located at $\overline{\mathbf{z}}_i$ and c_1 lies in the direction of $\mathbf{r}_x$. A set of valid robot configurations $\mathcal{C}_{free}$ is obtained by applying Algorithm 2, Lines 1-7. $\overrightarrow{\overline{\mathbf{z}}_x o_k}$ is the normalized vector from $\overline{\mathbf{z}}_x$ to o_k. $\mathcal{R}_a(b) \in SO(3)$ is the rotation matrix representing the rotation about the a-axis by angle b. $\mathcal{T}(b) \in SE(3)$ is the translation matrix translating a point from the origin to position b. Based on the collision-free set of IKs $\mathcal{C}_{free}$, an undirected graph $\mathbb{G}$ is constructed to find a set of optimal robot configurations in configuration space, where the tour cost has a minimal length in the configuration-space metric. An optimal tour of IK solutions over coverage lines $\overline{\mathcal{C}}_{free}$ can be obtained using the function `ComputeOptCfg`$(\mathcal{C}_{free})$, Algorithm 2 (by solving $\mathbb{G}$ using Dijkstra's algorithm in the same manner as [38]).

III. Experiments

We use a Ubuntu workstation with an Intel Xeon W-2255 CPU, 3.70 GHz x 20, 24-GB RAM to benchmark the proposed algorithm on four objects with the order ascending of complexity and size (from left to right in Fig. 5), including

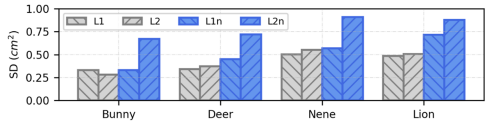

Fig. 3. Standard deviations of the cluster areas of the four objects.

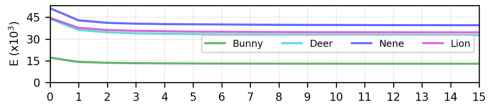

Fig. 4. Convergence of the energy function of the four objects.

Bunny (69,451 faces), Deer (139,306 faces), Nene (193,254 faces), and Lion (206,328 faces). The parameters are selected as $\alpha_1 = 1/6d_{dgn}, \alpha_2 = 0.93, \alpha_4 = 7$, where d_{dgn} is the diagonal length of the object bounding box. We select $\alpha_3 = 1/1.9$ for Bunny, Deer, and Nene. For Lion, we select $\alpha_3 = 1/3$ due to the high roughness of its geometry. The number of clusters m is determined based on the nozzle radius $r_c = 5\sqrt{2} \times 10^{-3}$ m, i.e., $m = \zeta_a(\mathbb{F})/\sigma_e$, where $\zeta_a(\mathbb{F})$ is the total area of all triangle faces and $\sigma_e = \pi r_c^2$ is the expected cleaning area at a robot configuration.

A. Coverage Path Planning Results

To assess the effectiveness of the proposed energy function in generating uniform cluster areas, we perform mesh segmentation experiments using different norm types in the distance function, represented by (3) with ℓ^1 norm (l_1), (3) with ℓ^2 norm (l_2), (5) with ℓ^1 norm (l_{1n}, proposed method), and (5) with ℓ^2 norm (l_{2n}). We evaluate its uniformity by computing the SD of the obtained cluster areas for the four objects, as shown in Fig. 3. Results show that l_2 yields the lowest SD in the Bunny object (lowest curvature). For higher curvature meshes, however, ℓ^1-based norms perform better at generating more uniform cluster areas. Particularly, the use of l_{1n} with the normal cost results in significantly more uniform cluster areas compared to l_{2n}. Fig. 4 shows the convergence of the energy function l_{1n} of all objects after 16 iterations. The result of computing CCVTs is described as the second row of Fig. 5. The blue lines are geodesic paths connecting all generators of neighbor-connected clusters. The triangle normals on each cluster vary within a specific range while maintaining a nearly uniform distribution of clusters over the mesh.

The CPs on the meshes are shown in the third row of Fig. 5, where a unique, shortest path connects all generators to form CPs on mesh models. The computation time of our CPP mainly depends on the computation of exact GDs (Algorithm 1, Lines 2–7). Using the proposed decomposition method, the GDs between segmented clusters for the four objects are computed in 10 s (271 clusters), 23 s (209 clusters), 37 s (233 clusters), and 33 s (332 clusters), respectively. This is a significant improvement compared to the computing times of 39,240 s, 56,160 s, 91,908 s, and 225,828 s without applying the decomposition

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON RESULTS

	Coverage (%)				Overlap (%)				RSD (%)				$\mathbb{F}_{\text{unreach}}$ (%)			
	bl	$bl_{\times 3}$	l_2	Ours	bl	$bl_{\times 3}$	l_2	Ours	bl	$bl_{\times 3}$	l_2	Ours	bl	$bl_{\times 3}$	l_2	Ours
Bunny	64.6	95.1	97.7	97.6	44.1	86.3	10.7	11.5	0.8	0.43	1.1	1.3	5.7	1.6	5.4	0.5
Deer	80.6	97.6	98.9	97.5	63.6	92.7	26.9	28.8	2.1	1.2	2.2	2.4	21.3	5.9	21.3	5.9
Nene	67.3	95.3	94.4	92.3	45.1	87.9	10.7	8.3	1.4	0.86	2.0	2.4	16.7	5.8	15.0	2.5
Lion	69.0	93.6	95.1	92.4	57.8	86.1	11.3	11.4	2.1	1.2	1.6	1.9	19.7	5.1	17.7	13.7

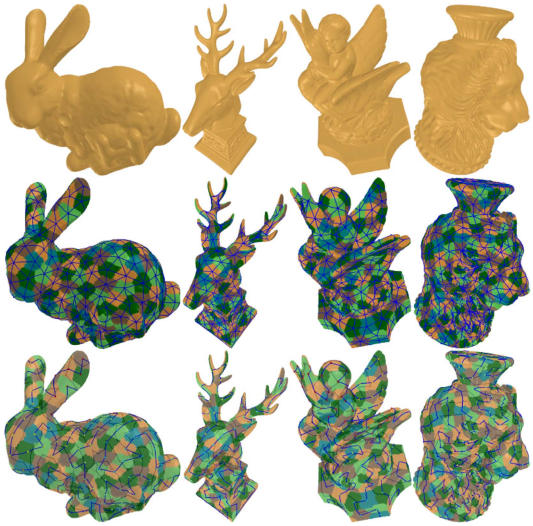

Fig. 5. Results of the planned coverage paths (Algorithm 1).

approach. The proposed method results in a slightly longer computation time for Nene than Lion due to Lion's more complex geometry, which requires a larger number of segmented clusters.

We further compare our proposed method (l_{1n}) with a conventional CCVT approach (l_2) and a baseline [17] by additional CPP criteria as Table I. bl and $bl_{\times 3}$ are the results produced by the baseline method using the same number of VPs and three times the number of VPs generated by our method, respectively. Here, we use GD from the triangles of interest to the generators of adjacent clusters and a threshold determined by r_c to estimate coverage and overlap rates. RSD is the relative standard deviation of the expected area per cluster computed based on clusters' area and σ_e. $\mathbb{F}_{\text{unreach}}$ is the set of unreachable triangles per object in terms of the normal cost. A triangle is added to $\mathbb{F}_{\text{unreach}}$ if the angle constructed by its normal and the corresponding generator's normal is larger than a certain value θ_0, i.e., $\theta = \arccos(\zeta_n(\tau) \cdot \zeta_n(\mathbf{z}_i)) > \theta_0$. The physical meaning of $\mathbb{F}_{\text{unreach}}$ is that if θ is greater than that value, then the performance of the task will be degraded. For the cleaning task, we select $\theta_0 = \pi/3$ rad. For each object, $\mathbb{F}_{\text{unreach}}$ (%) is determined by $\mathbb{F}_{\text{unreach}}/n_t$. Table I shows that the baseline method bl has significantly higher rates of overlapping and unreachable features ($\mathbb{F}_{\text{unreach}}$) and significantly lower coverage rates compared to our proposed method (l_{1n}). The coverage rates can be improved by $bl_{\times 3}$; however, the overlapping becomes markedly worse. Both l_2 and our proposed method (l_{1n}) achieve nearly complete coverage with much lower overlapping rates compared to the baseline.

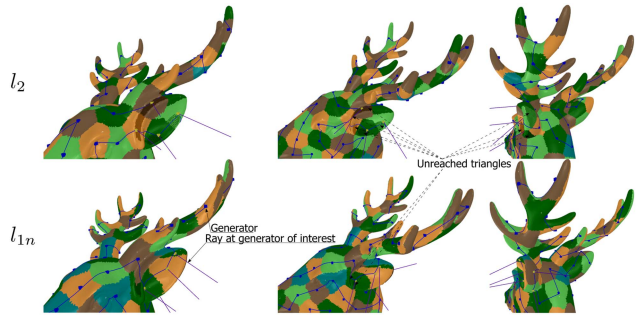

Fig. 6. CCVT results in different views around the left ear of the Deer.

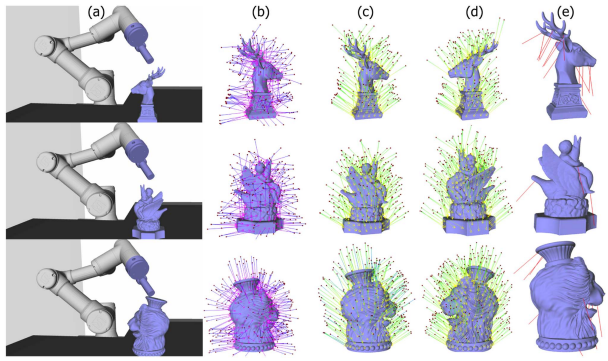

Fig. 7. Results of finding the set of valid configurations (Algorithm 2). (a) Object and robot in OpenRAVE. (b) Pre-calculated rays. (c) and (d) Rays of valid configurations after applying Algorithm 2. (e) Unrecoverable rays.

Fig. 6 shows the computed CCVTs in different views around the Deer's left ear. Generators on the ear are marked as yellow. For each cluster, edges of unreached triangles are visualized in darker colors compared to its corresponding cluster color. By using the proposed energy function, our method achieves a significantly lower portion of unreached elements. The presence of larger and more numerous dark portions (high $\mathbb{F}_{\text{unreach}}$ rate) in the l_2 method degrades the performance of the cleaning task as the airflow's impact on the target clusters is either diminished or absent. For model-based inspection and 3D reconstruction tasks, capturing entire clusters at computed VPs is not feasible, rendering VP planning using a set covering problem [17] ineffective. With nearly complete coverage, considerably low rates of RSD and $\mathbb{F}_{\text{unreach}}$, our proposed method allows for generating a high-quality set of VPs for surface-based coverage tasks.

B. Actual Cleaning Demonstration

Practical cleaning demonstrations are conducted using a collaborative robot (UR3e) for Deer, Nene, and Lion objects, which have high geometry complexity. The 3DP objects after

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

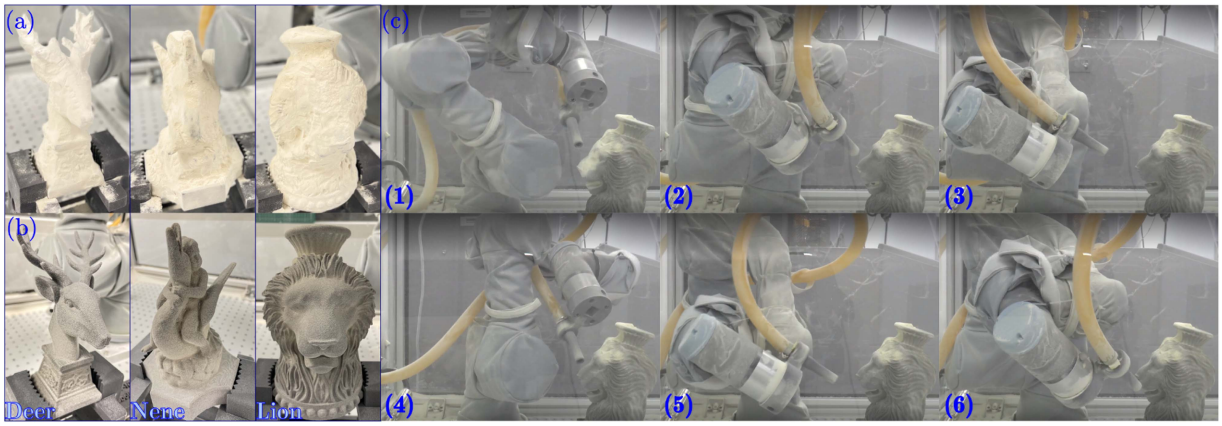

Fig. 8. Actual photos of 3DP samples. (a) After unpacking from a 3D printing station. (b) After performing the cleaning process. (c) Snapshots of the actual cleaning actions in an attempt to remove residual powder from Lion's mouth. The full video is available at https://youtu.be/eD5AjJKkEaI.

unpacking from a printing station (HP MJF5200) are shown in Fig. 8(a). The blasting pressure is set at 60 psi, $\theta_r = \pi/3$ rad, and $r_s = 0.05$ m. Fig. 7 shows the results of finding a set of valid robot configurations using Algorithm 2. The relative position of the object and the UR3e is (0.24 m, 0.2 m, 0.08 m), as shown in Fig. 7(a). By using Algorithm 2, a part of the rays in $\mathbb{R}_{opt}$ (Fig. 7(b)) is either adjusted by correcting their normal directions to attain feasible IK solutions $\mathcal{C}_{free}$, as shown in Fig. 7(c) and (d), or removed because there is no feasible IK solution satisfying the condition in Algorithm 2, line 5, as shown in Fig. 7(e). The set of unrecoverable rays can be reduced by increasing θ_r depending on the manipulator's arm length and the object placement. However, a larger θ_r also increases the planning and execution time, as it requires more time for correcting rays and transitioning between highly stretched configurations. Based on $\overline{\mathcal{C}}_{free}$, the final trajectory is computed using built-in functions in OpenRAVE and is sent to the robot using the Universal Robots ROS Driver. Fig. 8(c) shows the sequential snapshots of the cleaning actions for the Lion, where the robot fits its configurations properly to the computed collision-free configuration at generators. As a result, all residual powder is removed from Lion's mouth and also from the entire object surface, which demonstrates the effectiveness of our approach. The cleaning time is about 300 s. Although the computed coverage is not complete, practical implementation achieves nearly complete coverage as the actual affected area of the nozzle surpasses r_c.

In depowdering tasks, the residual powder on the object surface undergoes variations as it is blown away during the cleaning process. In addition, the affected cleaning area in specific configurations depends on various factors, such as surface properties (material, geometry complexity), the adhesion profile of unfused powder, and airflow power, making analytical determination of the affected cleaning size per VP impossible. In our approach, we select r_s, σ_e, θ_r, and blasting pressure from practical perspectives to ensure sufficient airflow for effective removal of residual powder at computed robot configurations, while consistently maintaining the distance of r_s at these configurations. However, this distance is relaxed during transitions between configurations to provide flexibility and rapidity for relocating the nozzle and the attached hose.

IV. Conclusion

In this study, we have introduced a new approach to obtaining a nearly complete coverage path on general surfaces and high-quality VPs using the information extracted from their mesh models. By introducing a new energy function, the mesh model is segmented into multiple clusters using the CCVT method such that coverage quality (uniform cluster areas) and the variation of triangle normals can be obtained harmoniously. Our results showed that using the ℓ^1 norm in the proposed energy function outperforms the ℓ^2 norm in achieving lower standard deviations of cluster areas on complex surfaces. A decomposition calculation is proposed to speed up the finding of the shortest path visiting all segmented clusters using the geodesic metric. As a result, we can attain high coverage, low overlapping, and low curvature of segmented clusters on the coverage path within a reasonable time. An effective optimization-based strategy is then proposed to address the self-occlusion of viewpoints and support the finding of valid robot configurations. Then, a collision-free robot trajectory is computed, allowing the CPP to be applied to cleaning and model-based inspection applications. We validated the effectiveness of our approach on an industrial cleaning platform used for additive manufacturing. Our future plans include: (1) Investigating advanced approaches [30], [39] to accelerate the construction of CCVTs. (2) Implementing faster approaches with high-quality approximated GDs [40] to accelerate CPP's computation. (3) Developing advanced model-based inspection and 3D reconstruction algorithms based on the obtained VPs.

Appendix A
Construct the Set of Candidate Rays Using Optimization

We propose the following optimization problem to construct the set of candidate rays:

$$\underset{c_i}{\text{minimize}} \quad \sum_{i=2}^{N_c} x_i^2 + y_i^2 \tag{10}$$

$$\text{subject to} \quad (x_1, y_1) = (0, 0), \tag{11}$$

$$z_i \geq R\cos(\varphi), \tag{12}$$

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.

$$\|c_i - c_j\|_2 \geq l,$$
$$i = 1, \ldots, N_c, j = i+1, \ldots, N_c, \qquad (13)$$

where $\mathbf{z}_i = \sqrt{R^2 - x_i^2 - y_i^2}$, $c_i = [x_i, y_i, z_i]^T$, l is the distance between two adjacent circles determined as

$$l = 2r_c \cos\left(\arctan\left(\frac{r_c}{r_s}\right)\right), \qquad (14)$$

The objective function (10) and the function on the left-hand side of (13) are both convex. Thus, the above formulation is not a convex optimization problem. This problem can be solved using disciplined convex-concave programming [41].

ACKNOWLEDGMENT

The authors would like to thank Dr. Yang Liu for sharing insightful comments on the CCVT topic, as well as Daren Ho for his generous support in operating the cleaning platform.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.
[2] M. Peuzin-Jubert, A. Polette, D. Nozais, J.-L. Mari, and J.-P. Pernot, "Survey on the view planning problem for reverse engineering and automated control applications," *Comput.-Aided Des.*, vol. 141, 2021, Art. no. 103094.
[3] H. Nguyen, J. Burkardt, M. Gunzburger, L. Ju, and Y. Saka, "Constrained CVT meshes and a comparison of triangular mesh generators," *Comput. Geometry*, vol. 42, no. 1, pp. 1–19, 2009.
[4] M. Skrodzki, E. Zimmermann, and K. Polthier, "Variational shape approximation of point set surfaces," *Comput. Aided Geometric Des.*, vol. 80, 2020, Art. no. 101875.
[5] D. Gleeson et al., "Generating optimized trajectories for robotic spray painting," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 3, pp. 1380–1391, Jul. 2022.
[6] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, "Indoor coverage path planning: Survey, implementation, analysis," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1718–1725.
[7] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Appl. Sci.*, vol. 1, no. 8, pp. 1–24, 2019.
[8] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *J. Intell. Robot. Syst.*, vol. 74, no. 3/4, pp. 965–983, 2014.
[9] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021.
[10] Y.-Y. Lin, C.-C. Ni, N. Lei, X. David Gu, and J. Gao, "Robot coverage path planning for general surfaces using quadratic differentials," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5005–5011.
[11] C. Wu et al., "Energy-efficient coverage path planning for general terrain surfaces," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2584–2591, Jul. 2019.
[12] J. Moura, W. Mccoll, G. Taykaldiranian, T. Tomiyama, and M. S. Erden, "Automation of train cab front cleaning with a robot manipulator," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3058–3065, Oct. 2018.
[13] A. M. Kabir, K. N. Kaipa, J. Marvel, and S. K. Gupta, "Automated planning for robotic cleaning using multiple setups and oscillatory tool motions," *IEEE Trans. Automat. Sci. Eng.*, vol. 14, no. 3, pp. 1364–1377, Jul. 2017.
[14] J. Hess, G. D. Tipaldi, and W. Burgard, "Null space optimization for effective coverage of 3D surfaces using redundant manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1923–1928.
[15] S. McGovern and J. Xiao, "UV grid generation on 3D freeform surfaces for constrained robotic coverage path planning," in *Proc. IEEE 18th Int. Conf. Automat. Sci. Eng.*, 2022, pp. 1503–1509.
[16] C. Dornhege, A. Kleiner, and A. Kolling, "Coverage search in 3D," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2013, pp. 1–8.
[17] W. Jing, J. Polden, P. Y. Tao, C. F. Goh, W. Lin, and K. Shimada, "Model-based coverage motion planning for industrial 3D shape inspection applications," in *Proc. IEEE 13th Conf. Automat. Sci. Eng.*, 2017, pp. 1293–1300.
[18] M. Gronle and W. Osten, "View and sensor planning for multi-sensor surface inspection," *Surf. Topogr.: Metrology Properties*, vol. 4, no. 2, Apr. 2016, Art. no. 024009. [Online]. Available: https://dx.doi.org/10.1088/2051-672X/4/2/024009
[19] B. J. Englot, "Sampling-based coverage path planning for complex 3D structures," Ph.D. dissertation, Mech. Eng., MIT Press, Cambridge, MA, USA, 2012.
[20] F. Prieto, R. Lepage, P. Boulanger, and T. Redarce, "A CAD-based 3D data acquisition strategy for inspection," *Mach. Vis. Appl.*, vol. 15, pp. 76–91, 2003.
[21] D. Mosbach, P. Gospodnetić, M. Rauhut, B. Hamann, and H. Hagen, "Feature-driven viewpoint placement for model-based surface inspection," *Mach. Vis. Appl.*, vol. 32, pp. 1–21, 2021.
[22] E. Glorieux, P. Franciosa, and D. Ceglarek, "Coverage path planning with targetted viewpoint sampling for robotic free-form surface inspection," *Robot. Comput.- Integr. Manuf.*, vol. 61, 2020, Art. no. 101843.
[23] J. X.-Y. Lim and Q.-C. Pham, "Automated post-processing of 3D-printed parts: Artificial powdering for deep classification and localisation," *Virtual Phys. Prototyping*, vol. 16, no. 3, pp. 333–346, 2021.
[24] H. Nguyen, N. Adrian, J. L. X. Yan, J. M. Salfity, W. Allen, and Q.-C. Pham, "Development of a robotic system for automated decaking of 3D-Printed parts," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8202–8208.
[25] Z. Liu, J. Geng, X. Dai, T. Swierzewski, and K. Shimada, "Robotic depowdering for additive manufacturing via pose tracking," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10770–10777, Oct. 2022.
[26] S. Valette and J.-M. Chassery, "Approximated centroidal Voronoi diagrams for uniform polygonal mesh coarsening," in *Computer Graphics Forum*, vol. 23. Hoboken, NJ, USA:Wiley, 2004, pp. 381–389.
[27] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637–676, 1999.
[28] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *Proc. ACM SIGGRAPH Papers*, 2004, pp. 905–914.
[29] G. Rong, M. Jin, L. Shuai, and X. Guo, "Centroidal Voronoi tessellation in universal covering space of manifold surfaces," *Comput. Aided Geometric Des.*, vol. 28, no. 8, pp. 475–496, 2011.
[30] Y. Liu et al., "On centroidal Voronoi tessellation–energy smoothness and fast computation," *ACM Trans. Graph.*, vol. 28, no. 4, pp. 1–17, 2009.
[31] M. G. Mozerov and J. van de Weijer, "Improved recursive geodesic distance computation for edge preserving filter," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3696–3706, Aug. 2017.
[32] F. Briggs, R. Raich, and X. Z. Fern, "Audio classification of bird species: A statistical manifold approach," in *Proc. IEEE 9th Int. Conf. Data Mining*, 2009, pp. 51–60.
[33] J. S. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM J. Comput.*, vol. 16, no. 4, pp. 647–668, 1987.
[34] U. A. Brodowsky, S. Hougardy, and X. Zhong, "The approximation ratio of the k-Opt heuristic for the euclidean traveling salesman problem," 2021. [Online]. Available: https://arxiv.org/abs/2109.00069
[35] J. B. Allan, B. Wyvill, and I. H. Witten, "A methodology for direct manipulation of polygon meshes," in *New Advances in Computer Graphics*. Berlin, Germany: Springer, 1989, pp. 451–469.
[36] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.
[37] M. Dawson-Haggerty et al., "Trimesh," 2019. [Online]. Available: https://trimsh.org/
[38] F. Suárez-Ruiz, T. S. Lembono, and Q.-C. Pham, "RoboTSP–A fast solution to the robotic task sequencing problem," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1611–1616.
[39] G. Rong, Y. Liu, W. Wang, X. Yin, D. Gu, and X. Guo, "GPU-assisted computation of centroidal Voronoi tessellation," *IEEE Trans. visualization Comput. Graph.*, vol. 17, no. 3, pp. 345–356, Mar. 2011.
[40] Y. Y. Adikusuma, Z. Fang, and Y. He, "Fast construction of discrete geodesic graphs," *ACM Trans. Graph.*, vol. 39, no. 2, pp. 1–14, 2020.
[41] X. Shen, S. Diamond, Y. Gu, and S. Boyd, "Disciplined convex-concave programming," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 1009–1014.

Authorized licensed use limited to: ULAKBIM UASL - MIDDLE EAST TECHNICAL UNIVERSITY. Downloaded on November 18,2023 at 09:44:25 UTC from IEEE Xplore. Restrictions apply.