A General Approach for Constrained Robotic Coverage Path Planning on 3D Freeform Surfaces

Sean McGovern and Jing Xiao, *Fellow, IEEE*

***Abstract*— There are many industrial robotic applications which require a robot manipulator's end-effector to fully cover a 3D surface region in a constrained motion. Constrained surface coverage in this context is focused on placing commonly used coverage patterns (such as raster, spiral, or dual-spiral) onto the surface for the manipulator to follow. The manipulator must continuously satisfy surface task constraints imposed on the end-effector while maintaining manipulator joint constraints. While there is substantial research for coverage on planar surfaces, methods for constrained coverage of 3D (spatial) surfaces are limited to certain (parametric or spline) surfaces and do not consider feasibility systematically given manipulator and task constraints. There is a lack of fundamental research to address the general problem: given a manipulator, a 3D freeform surface, and task constraints, whether there exists a feasible continuous motion plan to cover the surface, and if so, how to produce a uniform coverage path that best satisfies task constraints. In this paper, we introduce a general approach to address this fundamental but largely open coverage problem. We have applied our approach to example 3D freeform surface coverage tasks in simulation and real world environments with a 7-DOF robotic manipulator to demonstrate its effectiveness.**

***Note to Practitioners*—This paper was motivated by the constrained coverage path planning problem on 3D freeform surfaces for many industrial applications, such as painting, spray coating, abrasive blasting, polishing, shotcreting, etc. It provides a principled and general approach that includes an automatic robotic system to find feasible robotic end-effector paths for covering a 3D freeform surface with some interaction from a human worker who provides key parameters related to the specific task without being an expert in robotics. Therefore, the approach enables a human worker who only has the domain knowledge of a specific coverage task to operate the general and automatic robotic system effectively for completing the task.**

***Index Terms*— Manufacturing automation, constrained coverage path planning (CPP), manipulator motion planning, computational geometry.**

I. Introduction

THE use of robots for surface coverage applications is common in many industries including those of automobiles, furniture, aircraft, construction, agricultural, and appliances. Industrial manipulator coverage applications (such as spray coating/painting, abrasive blasting, polishing, shotcrete, laser ablation, etc.) require a manipulator's end-effector to traverse the entire surface once while satisfying task criteria in terms of application thickness, cycle time, and material waste [1], [2], [3], [4], [5], [6], [7], [8], [9]. The quality of production is usually determined by manipulator tool coverage uniformity and how it was applied on the surface (i.e. angle of approach, surface offset, etc.). Manual generation of a continuous and even coverage tool path on a 3D freeform surface is complex, time consuming, ad hoc, and difficult to optimize. It also requires the human operator to have substantial knowledge of robotic manipulation. With the need for rapid and efficient production and repairs in related industries, it has become essential to enable automated and optimal robotic coverage on 3D freeform surfaces [10], [11].

A. Related Literature

Related literature can be clustered in four categories: (1) robotic coverage path planning, (2) robotic constrained surface coverage, (3) surface uv mapping, (4) human-robot interfaces in industry, as detailed below.

1) Coverage Path Planning: Coverage path planning is about finding a robot path to traverse a region to achieve certain desired coverage. Most general coverage path planning (CPP) methods are intended for 2D planar surfaces, commonly online, and for mobile robots [12], [13], [14], [15], [16]. There is also work exploring unknown environments [17]. Many methods for CPP on 3D surfaces are focused on sensor coverage for aerial or submersible robots (i.e. view planning) [18], [19], [20]; however, these are generally discontinuous and nonuniform coverage paths. Some methods proposed for agriculture [21], [22] attempt to make uniform coverage paths on 3D landscapes; nonetheless, these applications do not need to consider manipulator constraints.

2) Constrained Surface Coverage: Here the problem is concerned with enabling the motion of a robot manipulator constrained on the 3D surface for coverage. Previous methods intended for industrial manipulator coverage applications are limited to certain types of 3D surfaces, such as parametric or spline surfaces, and can produce unevenly spaced coverage paths [2], [3], [4], [5], [23], [24], [25]. Uniform coverage is necessary for optimal satisfaction of task requirements (e.g., uniform thickness) [15]. Furthermore, these methods are often focused on some specific tasks and not general, and they do not address the feasibility of continuous coverage.

Manuscript received 10 December 2022; revised 2 May 2023 and 17 August 2023; accepted 5 September 2023. This article was recommended for publication by Associate Editor V. Villani and Editor P. Rocco upon evaluation of the reviewers' comments. This work was supported in part by the U.S. Army Research Laboratory under Contract W911NF1920108 and in part by the NSF NRT under Grant 1922761. *(Corresponding author: Sean McGovern.)*

The authors are with the Robotics Engineering Department, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: smmcgovern@wpi.edu; jxiao2@wpi.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2023.3313228.

Digital Object Identifier 10.1109/TASE.2023.3313228

There is substantial work on constrained manipulator motion planning [26], [27], [28], [29], [30], which focuses on finding a feasible path connecting two configurations while the end-effector remains constrained; however, these do not consider surface coverage and assume a feasible path already exists. A feasible constrained coverage path over a surface may not exist given certain manipulator and task constraints. In [39], we introduced a general method to inform on coverage path feasibility, which uses a uv space generated from parametric surface parameters to represent task constraints on 3D parametric surfaces. However, it is often difficult and not always possible to represent freeform surfaces with parametric equations. There is little work on uv space generation on 3D freeform surfaces for constrained surface coverage.

3) Surface uv Mapping: uv mapping for 3D surfaces is an area of interest mostly within the field of computer graphics. Texture mapping is a process in computer graphics which maps diffuse colors, normals, displacement, and other shading parameters onto a 3D polygon mesh from a 2D planar uv region. Generally, the main use of uv mapping for computer graphics is efficient storage of texture signals in a 2D image or to place an image in the 2D uv space for 3D mapping [31].

Creating a texture map involves manually separating the 3D mesh into segments that are unwrapped onto a 2D planar surface, effectively planarizing the 3D mesh. Seams are created when a 3D mesh is separated into segments that are unwrapped. The presence of seams and multiple mesh segments in the uv space introduces problems of discontinuities along the seams. Color transitions between the two mesh segments can be visible [32], [33]. In robotic constrained coverage of 3D freeform surfaces, uv mapping is useful for the generation of a coverage path pattern, which requires a seamless uv representation. However, there is no prior work on that.

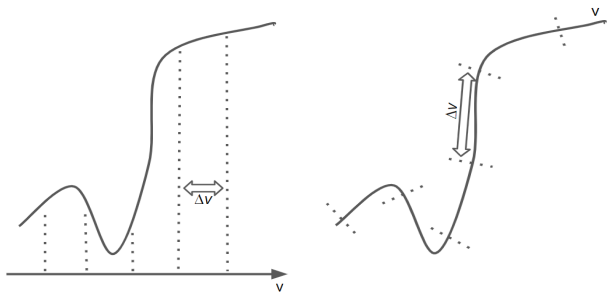
4) Human-Robot Interfaces in Industry: Many applications require some human interface with a robotic system. Trajectory and tooling methods for human-robot interaction in certain applications have been developed [34]. This interface or communication can occur physically through either touch or through non-physical communication such as gesturing or voice. Graphics interfaces have also been developed to help workers understand the needed input for the robot to complete complex tasks [35]. However, previous constrained surface coverage methods do not address how to include a human interface.

B. Contributions of This Paper

In this paper, we introduce a general approach to address the following fundamental problem: given a manipulator, a 3D freeform surface, and task constraints, how to enable a human worker, who is a task expert but not trained in robotics, to use automatic algorithms to determine the feasibility for the end-effector to traverse the entire surface continuously while maintaining the task and manipulator constraints, and if it is feasible, produce an evenly spaced, continuous coverage path that best satisfies task constraints. Our approach consists of the following contributions:

- a human-assisted method for automatic generation of uv grids on 3D freeform surfaces,
- a method for coverage feasibility checking of 3D freeform surfaces covered by uv grids,
- a method for automatic placement of a coverage pattern on the uv grid of a 3D freeform surface and mapping of the pattern from the uv grid to the Cartesian space,
- a method to find automatically a feasible coverage path of the manipulator to follow the coverage pattern.

The rest of the paper is as follows. In Section II, we provide an overview of our general approach. In Sections III-VI, we describe the generation of uv space and grids for 3D freeform surfaces, method for feasibility checking to satisfy task and manipulator constraints, placement of a coverage pattern on a freeform surface through its uv grid, and generation of feasible coverage paths for the robot manipulator to follow the coverage pattern, respectively. We present the testing results in simulation and in real world environments in Section VII and conclude the paper in Section VIII.



(a) Discretization along an axis results in uneven spacing on curve C. (b) Even discretization of curve C with v as a length parameter.

Fig. 1. Comparing two different ways of discretization along a curve C.

II. Overview of Methodology

Our general approach consists of four stages, which we briefly introduce in this section: (A) generating a uv space and grid on a 3D freeform surface, (B) constrained coverage feasibility checking, (C) putting a coverage pattern on a freeform surface, and (D) coverage path singularity detection and avoidance.

A. Generating uv Grid on 3D Freeform Surfaces

Previous work for coverage path planning on 3D surfaces [2], [3], [4], [5], [23], [24], [25] relied on Cartesian axes as references for coverage pattern spacing. This can result in uneven discretization of the surface, as illustrated in Fig. 1(a).

We consider three general types of 3D freeform surfaces in this paper, as defined in Section III. For those general types, we can define the uv space directly on the freeform surface and discretize it to generate a grid of evenly spaced cells, *uv-cells*. Our idea is to let the uv parameters be curve length parameters along the surface to achieve even spacing of the freeform surface along a curve, as illustrated in Fig. 1(b). Section III provides the details.

B. Continuous Coverage Feasibilty Checking

Coverage feasibility refers to the manipulator's end-effector path satisfying continuously both task and manipulator constraints, which depends on manipulator kinematics and surface parameters, while covering an entire surface. Violation of manipulator constraints causes either no solution for inverse kinematics or singularity configurations that prevent the end-effector to move smoothly along the surface.

In general, the position and orientation constraints on the manipulator or its end-effector are either equality or inequality constraints: **Equality constraints** on application surface coverage include end-effector offset and uniform thickness, which in turn constrain the end-effector position and velocity; **Inequality constraints** include limits on tool angle to surface, which constrain the end-effector orientation, manipulator joint limits, which constrain the joint positions, and velocity change limits, which constrain both the end-effector and manipulator trajectories.

Our previous coverage feasibility algorithm in [39] applies to parametric surfaces and checks that constraints are satisfied continuously while the end-effector visits every surface uv-cell once. To avoid expensive inverse kinematic calculations, our approach searches for the existence of a feasible end-effector path between uv-cells in the manipulator's joint space.

In this paper, for feasibility checking regarding freeform surfaces, we extend the algorithm by deriving constraints on 3D polygonal meshes and improve the algorithm to make end-effector joint space search more efficient based on information from the manipulator Jacobian, as described in Section IV.

C. Placing Coverage Pattern on the uv Grid

If the feasibility check determines that a feasible continuous coverage path exists for a 3D freeform surface given the task and manipulator constraints, a coverage pattern is placed on the uv grid and mapped to the Cartesian space to facilitate coverage motion determination. Coverage patterns such as raster scans or Archimedean spirals are generally used for constrained surface coverage applications due to their constant separation between adjacent turns, thus resulting in uniform coverage [1], [7].

Section V describes the selection of coverage patterns based on raster scans and how to put such a pattern on the uv grid of the surface.

D. Coverage Path Singularity Detection and Avoidance

Given that a feasible coverage path exists (from feasibility checking) and a coverage pattern, our system next generates the end-effector's coverage path that satisfies the position and orientation constraints. The initial end-effector position can be determined by applying an offset d from the surface (as required by the position task constraint), and the initial end-effector orientation can be determined as the optimal tool angle to surface (usually the surface normal, as required by the orientation constraint). An initial end-effector path can be created in terms of the sequence of end-effector positions and orientations corresponding to the centers of uv-cells along the coverage pattern. However, such an initial path may not be feasible, and our system finds a feasible path by detecting singularities along the initial path and modifying the end-effector configurations mainly through altering the end-effector orientations to avoid singularities.

Section VI presents our coverage path singularity detection and avoidance methods.

Figure 2 shows the overview of these stages in our general approach.

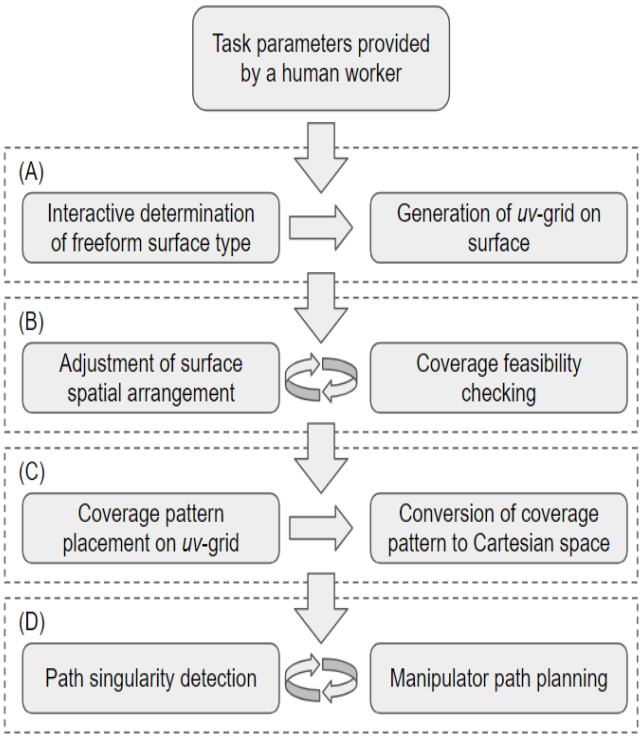


Fig. 2. Illustrates the sequential progression of the four stages in our general approach: (A) generation of a uv-grid on a 3D freeform surface, (B) constrained coverage feasibility checking, (C) coverage pattern placement on the uv-grid, and (D) coverage path singularity detection and avoidance.

III. Generating uv Space and Grid for 3D Freeform Surfaces

In this section, we first introduce three freeform surface types. We then discuss the human worker interface to identify the freeform surface type and key parameters for automatic generation of the uv grids. Next, we derive the uv space generation for each surface type and finally explain discretization of the uv space into a grid.

A. Freeform Surface Types

We consider a 3D polygon mesh representation of a physical freeform surface, S. The physical surface of an object can be scanned using modern 3D scanning technology and approximated as a 3D polygon mesh, which is the most common way to represent a general surface when there is no other more precise model. Additionally, S must fit within the workspace of a robotic arm to allow the end-effector to cover the surface. A large surface may be segmented to small S's using hierarchical methods as done in [36] and [37].

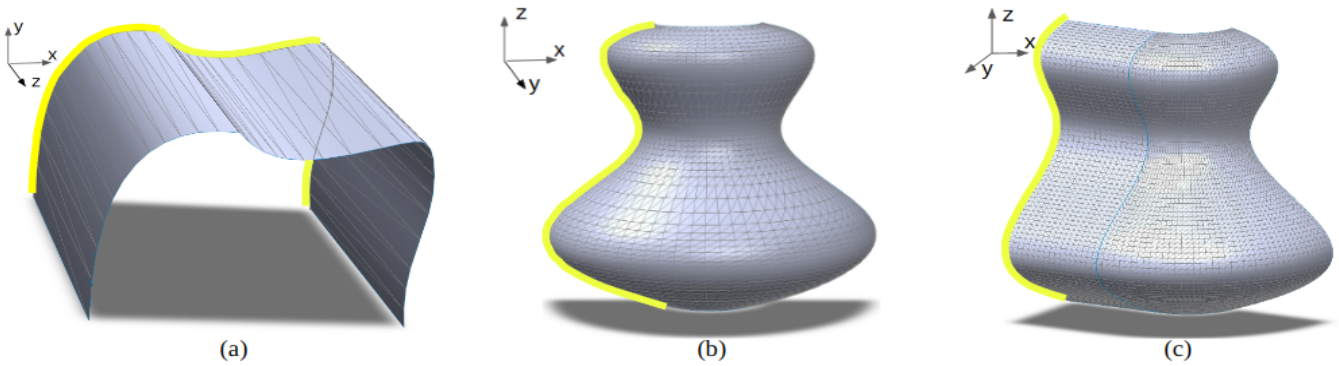


Fig. 3. Illustrations of 3D freeform surface types (embedded polygon meshes). Curve C_0 for each definition is highlighted in yellow. (a) Freeform surface of translation along the z axis. (b) Freeform surface of rotation about the z axis. (c) Freeform surface of translation (on the xy plane) and rotation (about axis parallel to the z axis).

We define and consider the following three types of general 3D freeform surfaces that can be generated from some transformation of a planar, non self-intersecting freeform curve C, along or about some axis, called $\mathcal{A}$ (see Fig. 3),

- **Surface of translation (SoT)**: translation of C, along $\mathcal{A}$.
- **Surface of rotation (SoR)**: rotation of C, about $\mathcal{A}$.
- **Surface of translation and rotation (SoTR)**: combinations of SoTs and SoRs from a single C, such that all rotation axes of C are parallel to $\mathcal{A}$ and all translation axes of C are on a plane normal to $\mathcal{A}$.

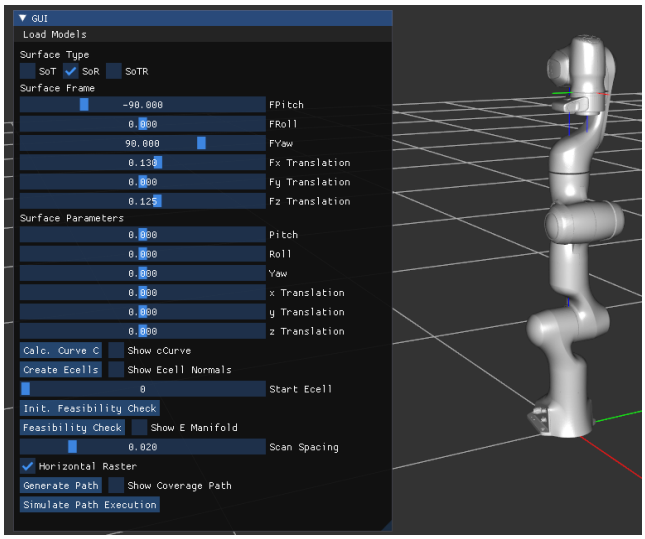
The curve C can be known or approximated from the input of a human worker.

Note that the above surface types share an important property that curve C is theoretically the same along each axis of translation or about each axis of rotation[1]. For a SoTR, the surface can be a continuous combination of translations and rotations of curve C, which may occur simultaneously, as long as the rotation axis remains in the same direction, and the translation axis is always orthogonal to the rotation axis. The above surfaces can capture many surfaces in the real world.

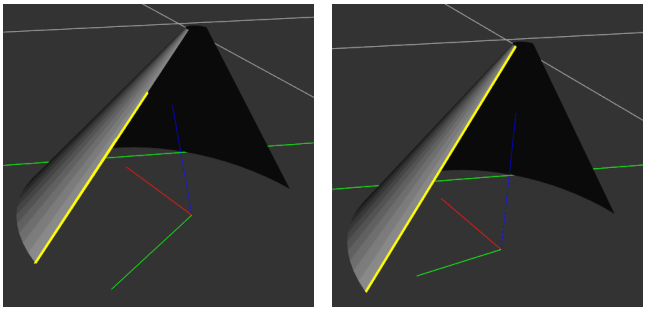
B. Interactive Determination of Key Parameters

Our simulation interface provides a human worker, who is a task-domain expert but not a robotics expert, with a visual and numerical representation of the spatial and orientation arrangement between the surface S and the manipulator. We denote S_f as the coordinate frame of S. The human worker may adjust the position and orientation of S_f with respect to the manipulator base (world frame), thus changing the arrangement between S and the manipulator. Additionally, S_f is positioned by the human worker with respect to surface mesh vertices to facilitate automatic generation of the uv grid. Fig. 4(a) shows the interface (numerical sliders) and visual display for manipulating S_f.

To enable automatic uv grid generation, the human worker first determines the surface S as one of the three types defined. Then, they align the (blue) z-axis of the surface frame S_f with axis $\mathcal{A}$, and our underlying automatic system helps the human worker to verify if S_f is placed accurately by showing the resulting curve C (highlighted in yellow). Fig. 4(b) shows an inaccurate placement of S_f, where the curve C is too short, and Fig. 4(c) shows the accurate placement of S_f.



(a) Simulation interface for interactive determination of key parameters.



(b) Inaccurate placement of surface frame, S_f, resulting in poor curve C (yellow) calculation.

(c) Accurate placement of surface frame, S_f, resulting in good curve C (yellow) calculation.

Fig. 4. Illustration of simulation interface for human worker to interactively determine key parameters to facilitate automatic uv space generation.

C. Automatic Generation of uv Space

We now describe automatic generation of uv space for the three types of freeform surfaces, surface of translation

[1]Practically, for a freeform surface mesh, C is approximated as a concatenation of (short) straight-line segments as the result of the triangle mesh and can be slightly different at different locations on the surface mesh.

(SoT), surface of rotation (SoR), and surface of translation and rotation (SoTR), in turn. For each type of surface, the surface coordinate frame S_f is set such that the z axis is along the axis $\mathcal{A}$, as shown in Fig. 3. The v coordinates are defined along the curve C, recalling that the curve C remains the same theoretically during translation and rotation to form the surface. The starting v coordinate is at one end point of C. The u coordinates are defined along curves perpendicular to C. The following describes how to obtain the $[u, v]$ coordinates for all vertices of the triangle mesh for each type of surface.

uv Space for SoT:

Let C_0 be the curve C on the xy plane at the beginning of translation (obtained by the human worker) with z_0.

Our algorithm sweeps (i.e., translates) a plane perpendicular to the z axis from z_0 to increase z value, starting from the xy plane. The sweeping pauses at every triangle vertex encountered and records the z value as z_i, $i = 1, \ldots, m$. Our algorithm then obtain the corresponding curve C_i as the intersection of the xy plane with surface S at z_i.

Let $(x_{i,j}, y_{i,j}, z_{i,j})$ be the coordinates of the jth triangle vertex on the curve C_i, $j = 0, \ldots, n_i$. These points share the same u coordinate:

$$u_{i,j} = z_i. \tag{1}$$

For $j = 1, \ldots, n_i$, let $\Delta x_{i,j} = x_{i,j} - x_{i,j-1}$ and $\Delta y_{i,j} = y_{i,j} - y_{i,j-1}$. We can compute $v_{i,j}$ by initializing $v_{i,0} = 0$ and:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta x_{i,j}^2 + \Delta y_{i,j}^2}, \quad j = 1, \ldots, n_i. \tag{2}$$

Thus,

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta x_{i,k}^2 + \Delta y_{i,k}^2}. \tag{3}$$

uv Space for SoR:

We denote the angle of rotation as ϕ and radius (i.e. vertex distance from axis of rotation) as $r(z)$, which is an unknown function of z due to the freeform curve C. Let C_0 indicate the curve C on the plane of $\phi = \phi_0$.

Now, by increasing ϕ and rotating the corresponding plane (which goes through the z axis), our method pauses the rotating plane at every triangle vertex encountered and records the ϕ value as ϕ_i, $i = 1, \ldots, n$. Our algorithm then obtains the corresponding curve C_i as the intersection of the plane with surface S at ϕ_i.

Let $(x_{i,j}, y_{i,j}, z_j)$ be the local coordinates of the jth triangle vertex on the curve C_i with C_i, for $j = 0, \ldots, n_i$. We compute its u coordinate as:

$$u_{i,j} = r(z_j) \cdot \phi_i. \tag{4}$$

For $j = 1, \ldots, n_i$, let $\Delta z_j = z_j - z_{j-1}$ and $\Delta r_j = r_j - r_{j-1}$. We can compute $v_{i,j}$ by initializing $v_{i,0} = 0$:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta z_j^2 + \Delta r_j^2}, \quad j = 1, \ldots, n_i. \tag{5}$$

Thus,

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta z_j^2 + \Delta r_j^2}. \tag{6}$$

uv Space for SoTR:

In this case, all rotation axes are parallel to the z-axis, and all translation axes are orthogonal to the z-axis. Let C_0 be on plane $\mathcal{P}$ that also includes the z axis.

The SoTR surface and the xy plane intersects at curve U, which intersects C_0 at point p_0. Our method sweeps the plane $\mathcal{P}$ along U while maintaining it perpendicular to U at each point[2], and the sweeping pauses at every triangle vertex encountered on S by $\mathcal{P}$ and records the Cartesian position of the intersection point with U. Let p_i indicate the ith intersection point between the swept $\mathcal{P}$ and U, and let the corresponding C curve be C_i, for $i = 1, \ldots, m$.

Let $(x_{i,j}, y_{i,j}, z_{i,j})$ be the coordinates of the jth triangle vertex on the curve C_i, for $j = 0, \ldots, n_i$. For $i = 1, \ldots, m$, let $\Delta_j x_{i,j} = x_{i,j} - x_{i-1,j}$ and $\Delta_j y_{i,j} = y_{i-1,j} - y_{i,j}$. By initializing $u_{0,j} = 0$, we can compute the u coordinate of the point as:

$$u_{i,j} = u_{i-1,j} + \sqrt{\Delta_j x_i^2 + \Delta_j y_i^2}, \quad i = 1, \ldots, m. \tag{7}$$

Thus,

$$u_{i,j} = \sum_{k=1}^{i} \sqrt{\Delta_j x_k^2 + \Delta_j y_k^2}. \tag{8}$$

Now, Let $\Delta_i x_{i,j} = x_{i,j} - x_{i,j-1}$, $\Delta_i y_{i,j} = y_{i,j} - y_{i,j-1}$, $\Delta_i z_{i,j} = z_{i,j} - z_{i,j-1}$, $j = 1, \ldots, n_i$. By initializing $v_{i,0} = 0$, we can compute the v coordinate of the point as:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta_i x_{i,j}^2 + \Delta_i y_{i,j}^2 + \Delta_i z_{i,j}^2}, \quad j = 1 \ldots, n_i. \tag{9}$$

Thus,

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta_i x_{i,k}^2 + \Delta_i y_{i,k}^2 + \Delta_i z_{i,k}^2}. \tag{10}$$

D. Automatic Discretization of uv Space

For every triangle $\mathcal{T}$ on the surface S, its vertex with position $\tau_i = [\tau_{ix}, \tau_{iy}, \tau_{iz}]^T$, $i = 1, 2, 3$, now has uv coordinates $[\tau_{iu}, \tau_{iv}]^T$. Note that the uv coordinates at each of those points are real numbers as computed above, even though those discrete points are indexed by integers.

Additionally, for any other point s inside the triangle, its coordinates of either Cartesian space or uv space can be found based on its distance to one of the triangle vertices. Let $\mathbf{p}_s$ indicates point s's position in the Cartesian space, point s's barycentric coordinates, w_i, can be found from the following equations:

$$\mathbf{p}_s = \Sigma_i w_i \tau_i, \quad 0 \le w_i \le 1, \quad \Sigma_i w_i = 1. \tag{11}$$

From the barycentric coordinates, the point s's uv-space coordinates $[u, v]$ can be determined. In general, the Barycentric coordinates w_i's can be used to find either Cartesian or uv-space coordinates of a point inside a triangle from the corresponding vertex coordinates of the triangle.

To discretize the uv space into a grid with uniform cells, let Δu and Δv be the desired dimensions of an uv-cell. We denote a uv-cell with the cell center coordinates $[u_j, v_k]$, where j and

[2]by translation or rotation or simultaneous translation and rotation depending if the surface portion is SoT or SoR or both.

k are integer indices for the discretization, even though the uv-space coordinates u_j and v_k are real numbers from the original continuous space.

IV. Feasibility Checking

In this section, we first introduce a general position and orientation constraint formulation and next present an improved joint-space feasibility checking algorithm, which in turn determines end-effector coverage path feasibility.

A. Task Constraint Equations Regarding 3D Freeform Surfaces

In general, the end-effector's position and orientation task constraints are related, in that the distance constraint from the end-effector position to the surface S should be measured along the end-effector approach vector, which is determined by the end-effector's orientation task constraint. In [40], the end-effector's position and orientation constraints were introduced independently, which prohibited deviations of the approach vector from the optimal surface normal in order to satisfy the end-effector position constraint (to maintain constant offset) with respect to the surface. As the result, it was difficult to handle transitions around sharp edges and vertices (see Fig. 5(a)).

In order to achieve smoother transitions around edges and vertices of a 3D freeform surface (in polygonal mesh) during surface traversal (see Fig. 5(b)), this paper considers interrelated position and orientation task constraints as follows.

We denote the end-effector position in Cartesian space with respect to the surface coordinate frame as $\mathbf{p}$ and the end-effector approach vector as the unit vector $\mathbf{a}$ (along the z axis of the end-effector), such that:

$$\mathbf{a} = [r_{13}, r_{23}, r_{33}]^T, \tag{12}$$

where r_{**} is from the rotation matrix R of the manipulator end-effector. A task constraint is imposed on the end-effector such that its position must be offset from the surface at a distance d along the approach vector $\mathbf{a}$, satisfying

$$\mathbf{p} + d \cdot \mathbf{a} = \mathbf{p}', \tag{13}$$

where $\mathbf{p}'$ is the position of a point on the surface S. The approach vector $\mathbf{a}$ must also satisfy the orientation constraint:

$$-\mathbf{b} \cdot \mathbf{a} \leq \cos\alpha, \tag{14}$$

where $0 \leq \alpha \leq \epsilon$ is the allowable approach angle deviation from the surface normal, $\mathbf{b}$.

Let $\mathbf{p}'$ be on the uv-cell $[u_j, v_k]$, which is on a triangle $\mathcal{T}$ of the surface with vertex positions τ_1, τ_2, and τ_3. Using the task constraint of Eq. (13), we have

$$\mathbf{p} + d \cdot \mathbf{a} = \Sigma_i w_i \tau_i, \quad 0 \leq w_i \leq 1, \quad \Sigma_i w_i = 1, \tag{15}$$

which expresses the task constraint on the corresponding end-effector position $\mathbf{p}$ in terms of $\tau_\mathbf{i}$'s.

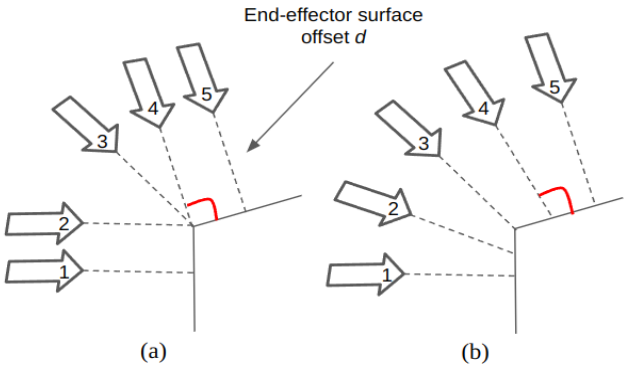


Fig. 5. Illustration to compare the effects of (a) position and orientation constraints defined independently, which causes large orientation shift around surface edges, and (b) position constraint defined along the approach vector $\mathbf{a}$, i.e., related to the orientation constraint, which allows for smooth orientation change around surface edges.

B. Joint-Space Task Constraints and Feasibility Checking

With end-effector position $\mathbf{p}$ expressed in terms of the triangle parameters of the uv-cell $[u_j, v_k]$ and the angle α as in equations (13)-(15), we can relate an n-dimensional robotic manipulator's link parameters, $\mathbf{l}$, and joint variables, $\mathbf{q}$, directly to the task constraint equations using forward kinematics to obtain *joint space task constraints*:

$$f(\mathbf{l}, \mathbf{q}) = c(u_j, v_k, \mathcal{T}, \alpha). \tag{16}$$

The manipulator joint configurations that satisfy the joint space task constraints Eq. (16) form what we call a *J-manifold*. We can discretize the J-manifold into an n-dimensional grid, where each cell, called a *J-cell*, corresponds to a joint configuration $\mathbf{q}$. The distance between two neighboring J-cell joint configurations is $\mathbf{dq}$ such that each joint variable's value q_i in $\mathbf{q}$ is increased or decreased by δq or remains the same to reach its neighboring J-cell. Thus, a given J-cell has $3^n - 1$ neighboring J-cells. Through forward kinematics a J-cell corresponds to a feasible end-effector configuration, an *E-cell* with position $\mathbf{p}$ and orientation R. We denote the space of all E-cells as the *E-manifold*.

An E-cell with position $\mathbf{p}$ maps to a point on the surface with position $\mathbf{p}'$, by Eq. (13), and $\mathbf{p}'$ can be converted to uv space (see Section III). Note that multiple E-cells with different $\mathbf{a}$ (approach) vectors may map to the same surface position $\mathbf{p}'$ or the same uv coordinates. Mappings between the J-manifold, E-manifold, and the uv grid are illustrated in Fig. 6.

Our coverage feasibility checking algorithm in Algorithm 1 explores the uv grid, using a tree search, reaching every uv-cell once. Starting from an initial uv-cell, uv_1, which inversely maps to a J-cell, jc_1, the algorithm stores neighbor uv-cells in *Neighbors* and randomly generates a none-zero $\mathbf{dq}$ by randomly assigning a value from $\{-\delta q, 0, \delta q\}$ for each joint. It calls Algorithm 2 to search the J-manifold moving through neighboring J-cells, until a corresponding E-cell is reached, which maps inside a neighboring uv-cell (i.e. the uv-cell with its center closest to the corresponding uv point of that E-cell), and establish that there is a *neighboring feasibility continuity* of the manipulator to cover the two uv-cells.

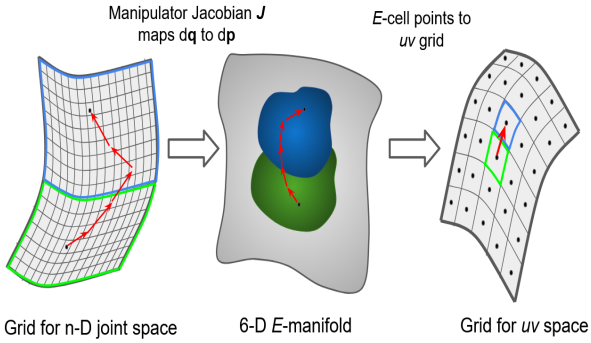


Fig. 6. Illustration showing the conversion from J-manifold (left), to the E-manifold (center), and then to the uv grid (right). The corresponding regions in each manifold are shown in the same blue/green colors.

Algorithm 1 Continuity Check on uv Grid

Global $Neighbors$;
Input uv_1 and corresponding jc_1;
Initialize Tree T at uv_1;
Initialize graph G containing all uv-cells with no edges;
T, G = TREESEARCH(T, G, jc_1);
if G is a connected component **then**
  return "∃ feasible path"
else
  return "no feasible path"
end
procedure TREESEARCH T, G, jc:
From jc get corresponding uv-cell called uv;
$Neighbors$ = unvisited neighbor uv-cells of uv;
repeat:
Randomly generate none-zero **dq**;
call Algorithm 2 with jc, **dq**, which returns *continuity* between uv and a neighbor uv_N and the corresponding J-cell jc_N;
$Neighbors$ =$Neighbors$ - $\{uv_N\}$;
if $continuity$ **then**
  Add edge between uv and uv_N if uv_N is not in G;
  if uv_N is not in T **then**
    Add uv_N as child to uv in T;
    T, G = TREESEARCH(T, G, jc_N)
  end
end
until $Neighbors = \emptyset$;
return T, G;

We make the Algorithm 2's joint space search more efficient by allowing the search to continue in a direction, **dq**, instead of a random direction for each J-cell transition, thus narrowing the search space. If following a direction **dq**, the search reaches a joint configuration **q** that satisfies joint space task constraints, then **dq** is used again until the constraints are no longer satisfied. Once these constraints are no longer satisfied, **dq** is adjusted using the Jacobian matrix **J(q)** to identify the order of joint variable q_i which least affects the resulting change **dx** in the end-effector configuration. Starting with the least significant joint variable, the adjustment procedure generates a new value that is not previously used for that joint in **dq**, trying to obtain a new **dq** that is close to the previous **dq** in the hope of satisfying joint space task constraints with as little deviation from the original search direction as possible.

Algorithm 2 Continuity Check on J-Manifold Between Neighboring uv-Cells

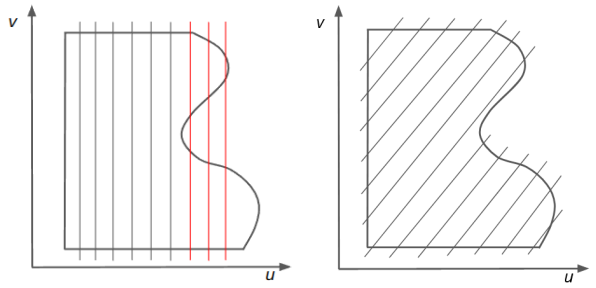
procedure CONT jc, **dq**:
repeat:
From **dq** find the neighbor jc' of jc;
if jc' satisfies joint-space task constraints and joint limits **then**
  E-cell ec is obtained from jc via forward kinematics;
  dx = **J**(**q**) · **dq**;
  From ec and **dx**, obtain next E-cell ec_N;
  By mapping ec and ec_N to the uv space, get the direction from uv to a neighbor uv_N;
  if $uv_N \in Neighbors$ **then**
    if uv_N is reached **then**
      $continuity = true$; $jc_N = jc'$;
      return $continuity$, uv_N, and jc_N;
    end
    call CONT(jc', **dq**)
  end
end
Adjust **dq** and find an unchecked neighbor J-cell jc';
until there is no valid J-cell transition from jc;
$continuity = false$;
return $continuity$, uv_N, jc_N;

This process continues in Algorithm 1 until every reachable uv-cell pair continuity is checked via a tree search and a uv space connectivity graph, G, is built based on neighboring feasibility continuity results. If the resulting graph is a single connected component, we determine that there is at least one end-effector path that can cover the surface S continuously while maintaining constraints.

This search process is essentially a depth-first search (DFS) in the joint space. It takes full advantage of the information from the Jacobian matrix to explore the most likely path first. This is why the search with DFS can be more efficient than with breadth-first search (BFS). DFS also has the well-known advantage of space efficiency over BFS.

The worst case time complexity of the uv grid search depends on the worst case time complexity of the feasibility search between two adjacent uv-cells. The worst case time complexity of the DFS is $O(b^m)$, where b is the maximum branching factor and m is the maximum depth of search. $b \leq 3^n - 1$, where n is the degrees of freedom of the manipulator. Usually, $b \ll 3^n - 1$ because (1) not all joints move during the transition from one uv-cell to an adjacent one, and (2) many joint configurations cause a violation of constraints. m depends on the joint limits and the value of δq. Essentially, if δq is used to discretize the high-dimensional joint space into a hyper-grid, m is bounded by the number of J-cells. Thus, the worst-case time complexity of the uv grid search algorithm is $O(N \cdot b^m)$, where N is the number of uv-cells. The actual running time for feasibility check between two adjacent uv-cells takes just a few milliseconds.

The DFS search within the manipulator's joint space persists until it reaches the adjacent center of the uv-cell or exhaustively explores all viable joint configurations. Given that this is a comprehensive exploration of the joint space,



(a) Start direction failed check on raster scan lines (red). (b) Start direction passes check on raster scan lines.

Fig. 7. Illustration showing examples of suitable and non-suitable raster scan start direction for uv grid surface.

the search is complete and will provide a feasible solution if it exists.

V. Putting Coverage Pattern on uv Grid

The convexity of a flattened (planar) uv space, as shown in Fig. 7, can be first used to decide whether a coverage pattern can be applied, since for some concave region, certain coverage patterns cannot provide uninterrupted coverage for the entire surface. If no coverage pattern can provide uninterrupted surface coverage, the surface can be decomposed into smaller and preferably convex regions [41], [42] so that each smaller region can be covered by a desired coverage pattern. Alternatively, other 2D CPP methods [12], [13], [14], [15], [16] could be adapted to design a coverage pattern in the uv space.

A common raster pattern is a set of parallel, straight "scan" lines which are separated perpendicularly by a distance, ω, and connected at their ends in an alternating manner. We denote the direction of the lines as the *start direction*. Reference [40] shows how to create a raster pattern on the uv grid with a horizontal start direction. In general, we can determine if a start direction will result in a raster coverage pattern that covers the surface uninterrupted by using a systematic check on the scan lines. If each scan line intersects the surface boundaries only twice, then that start direction is suitable for the uv grid surface (see Fig. 7). If not, the start direction is rotated to a new direction, and the test is repeated. The lines are shortened to fit within the surface boundaries with end points connected in an alternating manner.

We denote $\mathbf{UV}$ as a coverage pattern on the uv grid with uv_i being the ith uv point of the pattern such that $\mathbf{UV} = [uv_1, uv_2, \ldots, uv_M]$. Each uv_i indicates a center of a uv-cell; uv_i and uv_{i+1} indicates centers of neighboring uv cells, for $1 \leq i < M$. Note that the $\mathbf{UV}$ coverage pattern should only cross between uv-cells which are connected in the connectivity graph of the uv grid.

We denote $\mathbf{H}$ as the coverage pattern $\mathbf{UV}$ expressed in Cartesian coordinates, with $\mathbf{h}_i$ being the position of uv_i on the pattern such that $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_M]$.

VI. Coverage Path Singularity Detection and Avoidance

Let $\{R_1, R_2, \ldots, R_M\}$ denote a sequence of end-effector orientations, such that R_i is the rotation matrix of the manipulator initialized with the end-effector's z-axis along the normal $\mathbf{b}_i$ of the triangle that contains the ith waypoint uv_i of the coverage pattern. This is to best satisfy the orientation task constraint.

Algorithm 3 Manipulator Path Planner

```
Initialize end-effector orientation sequence
{R_1, R_2, ..., R_M};
Initialize joint-space path Q = {q_1, q_2, ..., q_M};
i = 2; count = 0;
while i ≤ M do
    check joint-space singularity and Cartesian-space
    singularity from q_{i-1} to q_i:
    while Singularity at q_{i-1} ≤ q* ≤ q_i and count < K
    do
        for j = 0 to i − 1 do
            smooth the orientation R_{i-j} if needed to
            reduce the difference to the orientation
            R_{(i-j)-1} while satisfying the orientation
            constraint and update p_{i-j} accordingly to
            satisfy position constraint;
            update q_{i-j} based on the updated p_{i-j};
        end
        count++;
    end
    if no singularity then
        i++;
    else
        exit with "no feasible solution";
    end
end
output Q;
```

We compute the end-effector's position $\mathbf{p}_i$ corresponding to each waypoint uv_i with position $\mathbf{h}_i$ in the Cartesian space as:

$$\mathbf{p}_i = \mathbf{h}_i - d \cdot \mathbf{a}_i, \tag{17}$$

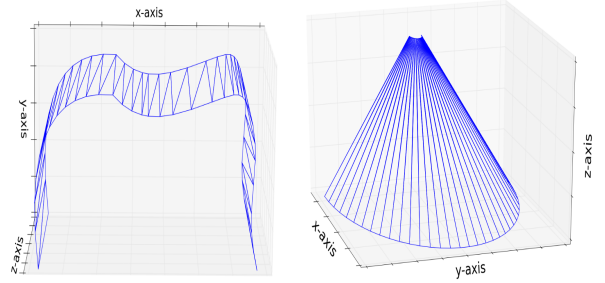
where $\mathbf{a}_i$ is the third column of R_i.

Now we initialize the end-effector coverage path $\mathbf{P}$, s.t. $\mathbf{P} = [E_1, E_2, \ldots, E_M]$, where E_i, for $i = 1, \ldots, M$, is a homogeneous transformation matrix consisting of position $\mathbf{p}_i$ and rotation matrix R_i. $\mathbf{P}$ can correspond to multiple paths in the joint space. A joint-space solution will be returned by an inverse kinematics solver, based on some optimization criteria (such as minimum joint distance, collision avoidance, and even some specific task constraints on the joints). We denote $\mathbf{Q}$ as such a solution, which is the sequence of corresponding joint states of $\mathbf{P}$, with $\mathbf{q}_i$ being the ith joint state on the path such that $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M]$. Note that, which joint-space solution is used does not really affect the task solution, because the Cartesian-space path $\mathbf{P}$ will remain the same.

If a feasible coverage path exists (as determined in Section IV), it does not mean the generated initial path $\mathbf{P}$ is a feasible coverage path. The following subsections describe singularity detection on initial coverage path $\mathbf{P}$, and if singularities are detected, how we alter path $\mathbf{P}$ to avoid singularities and best satisfy task constraints.

A. Singularity Detection

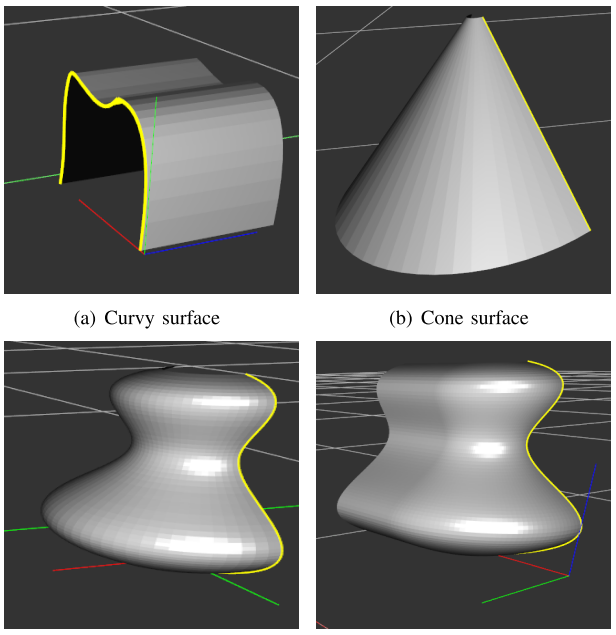
Even if the surface S is placed within the robot's workspace, depending on the shape and size of S, there can be internal



(a) Curvy surface (SoT). (b) Cone surface (SoR).

(c) Vase surface (SoR). (d) Corner surface (SoTR).

Fig. 8. Polygon mesh of four different freeform surfaces.



(a) Curvy surface (b) Cone surface

(c) Vase surface (d) Corner surface

Fig. 9. Initial curve C highlighted in yellow for each surface.

singularities to prevent the manipulator from following the initial path. Two kinds of singularities have to be considered: *joint space singularities* and *Cartesian space singularities*.

We refer to joint space singularities along a joint space path as those configurations that exceed a certain joint limit, which result in failures of motion/trajectory planning. We refer to Cartesian space singularities along a Cartesian space path as those end-effector configurations that cannot be reached with the imposed task constraints.

Our method checks the end-effector path $\mathbf{P}$ for such singularities through the corresponding manipulator joint space path $\mathbf{Q}$. As in [38], linear interpolation is performed between joint configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, for $i = 1, \ldots, M$, on path $\mathbf{Q}$, such that:

$$\mathbf{q}_{i_j} = \mathbf{q}_i + j \cdot \epsilon \cdot (\mathbf{q}_{i+1} - \mathbf{q}_i),\ j = 1, 2, \ldots \tag{18}$$

for a small ϵ. Forward kinematics yields corresponding Cartesian configurations. If the end-effector position $\mathbf{p}_{i_j}$ satisfies $|\mathbf{p}_{i_j} - \mathbf{p}_{i+1}| > |\mathbf{p}_{i_{j-1}} - \mathbf{p}_{i+1}|$, then a singularity occurs; otherwise, there is no singularity at $\mathbf{p}_{i_j}$. The joint-space path between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ should result in end-effector positions where $\mathbf{p}_{i_j}$ is closer to $\mathbf{p}_{i+1}$ than $\mathbf{p}_{i_{j-1}}$, otherwise a singularity exists between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$.

B. Singularity Avoidance

For every singularity encountered along the path (either a joint space singularity or a Cartesian space singularity), our planner alters the end-effector orientations in a neighborhood of the singularity configuration, called *orientation smoothing*, until a singularity free path is obtained.

For each singularity encountered along the end-effector initial coverage path $\mathbf{P}$, say at the ith waypoint, our method re-orients the end-effector approach vector $\mathbf{a}_{i-j}$ closer to that of the previous waypoint's approach vector $\mathbf{a}_{(i-j)-1}$ with: $\mathbf{a}_{i-j} = \frac{\mathbf{a}_v}{\|\mathbf{a}_v\|}$, where $\mathbf{a}_v = \frac{(\mathbf{a}_{i-j}+\mathbf{a}_{(i-j)-1})}{2}$, for $j = 0$ to $i-1$. The path can be smoothed up to N times, where the difference between neighboring approach vectors are reduced to 2^{-N} original difference. Satisfaction of task constraints are checked after orientation smoothing at each point. The worst case time complexity for each smoothing processes is O(M) and the total algorithm execution time is proportional to how many singularities are detected along the initial coverage path $\mathbf{P}$.

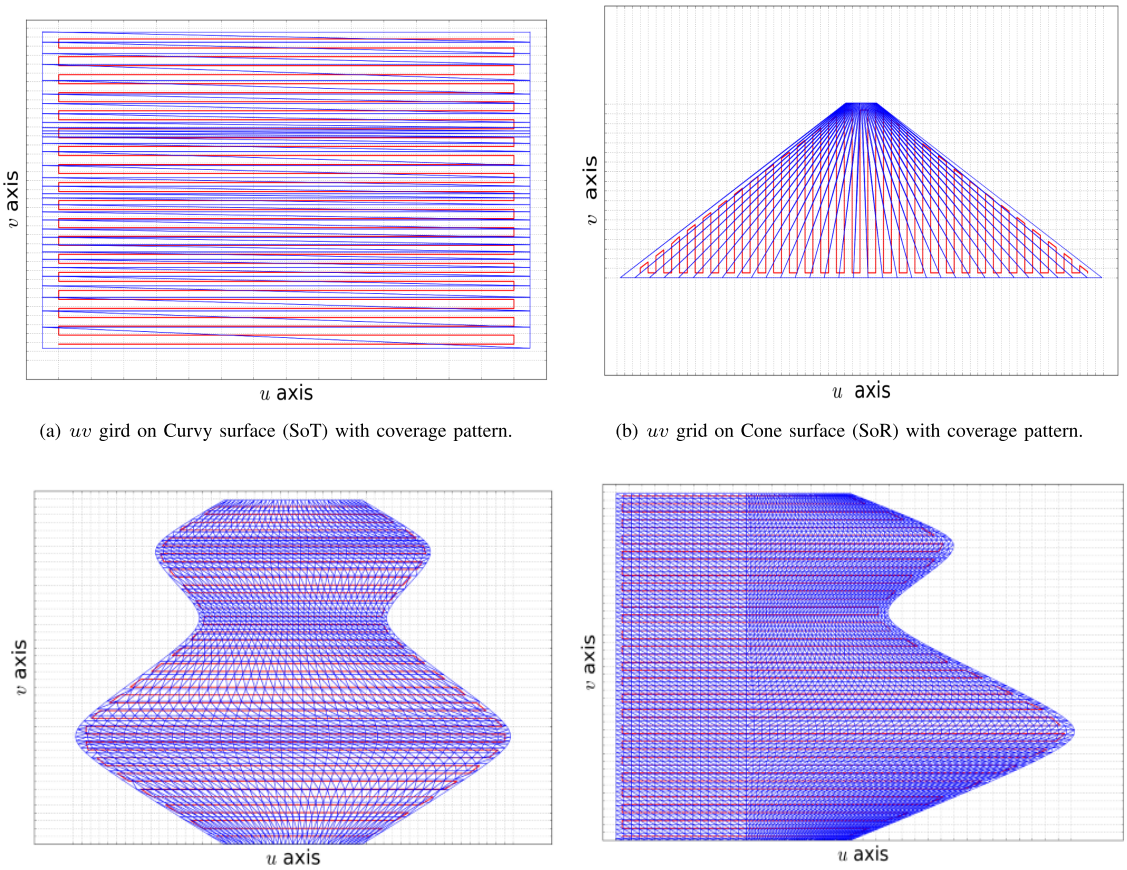
The complete manipulator path planner is shown in Algorithm 3, which alters the initial manipulator path to create a singularity free path, satisfying equality task constraints, and best satisfying inequality task constraints. If it is not possible to create such a path under the current coverage pattern, the planner returns "no feasible solution". Another coverage pattern can be tried.

VII. Implementation and Results

To test our approach, we used Solidworks to generate a polygon mesh of a Curvy (SoT), Cone (SoR), Vase (SoR), and Corner (SoTR) surface (see Fig. 8), containing 68, 32, 3743, and 9200 triangle faces respectively. We used a Franka Emika Panda manipulator which contains seven revolute joints, with the joint vector $\mathbf{q} = [\theta_1, \theta_2, \ldots, \theta_7]^T$ and link parameters $\mathbf{l} = [d_1, d_3, a_4, a_5, d_5, a_7] = [0.3, 0.3, 0.08, -0.08, 0.3, 0.08]$(m).

Fig. 9 shows the results of interactive placement of the surface frame and determination of the C curve in the virtual environment. Fig. 10 gives a flattened view of the resulting uv grid for each surface; a raster coverage pattern (red) overlayed on the top. The resulting initial coverage paths are shown in Fig. 11.

Fig. 11 (b) shows the results of a coverage pattern that was generated on a cone surface using our method. This pattern is not trivial to generate using other methods but is easily generated and evenly spaced using our uv grid. The "vertical" raster pattern is straight near the center yet spiral on either side. This is a simple example to demonstrate that using the



(a) uv gird on Curvy surface (SoT) with coverage pattern.

(b) uv grid on Cone surface (SoR) with coverage pattern.

(c) uv grid on Vase surface (SoR) with coverage pattern.

(d) uv grid on Corner surface (SoTR) with coverage pattern.

Fig. 10. Generated uv grid (blue) with applied raster coverage patterns **UV** (red) on each freeform surface.

TABLE I
NUMBER OF SINGULARITIES FIXED BY PATH PLANNER

	Curvy (SoT)	Cone (SoR)	Vase (SoR)	Corner (SoTR)
Arr.	# Sing.	# Sing.	# Sing.	# Sing.
1)	0	0	0	0
2)	3	4	12	4
3)	16	8	35	22

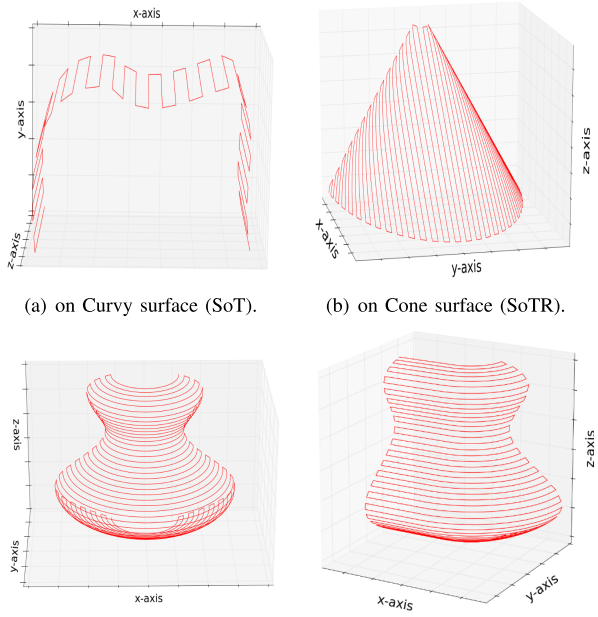
uv grid generated by our method, non-trivial coverage patterns can be produced on 3D freeform surfaces.

Table I presents the number of singularities fixed by Alg. 3 for each surface in three spatial arrangements relative to the robot.

Table II shows the corresponding results of the coverage paths and executed trajectories. The coverage path for the first spatial arrangement for each surface is shown in the accompanying video.

For each surface, the first spatial arrangement resulted in no singularities for the end-effector coverage path. These arrangements are within a more desirable workspace of the given manipulator. The other spatial arrangements resulted in singularities, most likely a result of joint values nearing the manipulator joint limits. Our orientation smoothing method outlined in Alg. 3 successfully smoothed the coverage path orientations within task constraints and produced a feasible path for each spatial arrangement.

The attached video first shows the Panda manipulator traversing the coverage pattern (in red) for the Curvy, Cone, Vase, and Corner surface in the virtual environment and then shows the real Panda manipulator traversing the coverage pattern on the 3D printed Corner surface. Fig. 12 gives snapshots of the virtual Panda manipulator traversing a surface



(a) on Curvy surface (SoT).

(b) on Cone surface (SoTR).

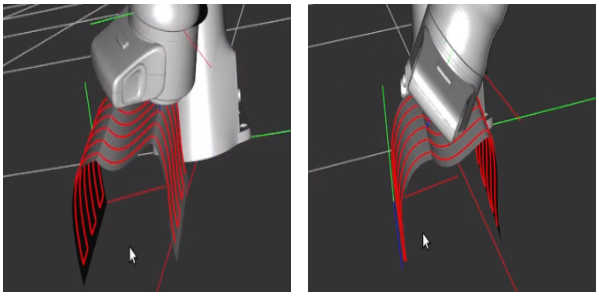
(c) on Vase surface (SoR).

(d) on Corner surface (SoTR).

Fig. 11. Raster scan coverage patterns on different surfaces.

TABLE II
COVERAGE PATH/TRAJECTORY RESULTS

Curvy (SoT)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	48.6	7.1	1.4	1.42
2)	4.9°	0.28°	45.3	6.8	1.3	1.56
3)	5.7°	0.43°	44.3	6.6	1.3	1.74
Cone (SoR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	47.6	8.3	1.5	0.9
2)	2.2°	0.15°	45.8	7.9	1.2	1.13
3)	6.1°	0.21°	45.2	7.5	0.9	1.24
Vase (SoR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	43.5	13.6	1.5	0.41
2)	4.7°	0.42°	41.7	12.7	1.4	0.54
3)	8.9°	0.72°	39.2	12.5	1.2	0.61
Corner (SoTR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	33.8	12.3	2.0	0.42
2)	3.4°	0.41°	34.3	11.6	1.9	0.48
3)	5.5°	0.61°	31.2	11.1	1.7	0.53



(a) (b)

Fig. 12. Snapshots of the Panda manipulator moving along a coverage pattern (red) on the Curvy (SoT) surface.

coverage pattern along the curvy (SoT) surface (as shown in accompanying video). Note the distance between raster scans, ω, is larger in the video than as shown in Fig. 11 to reduce the coverage execution time.

Note that we placed a different coverage pattern in the demo of Fig. 12 from the one shown in Fig. 11(a). Both are valid coverage patterns. This shows that our method can place different coverage patterns on a freeform surface with ease.

A. Discussion

A human worker can divide a surface into segments that align with the primitive surface definitions discussed in Section III through the simulation interface. In this way, a lot of surfaces in real-world applications can be nicely represented and handled by our approach. However, for extremely rugged and irregular surfaces, such as the surface of a complex sculpture, there can be too many segments for our method to be practical.

In Fig. 10, we flattened the uv space and used straight-line axes to represent u and v to produce the 2D view. Note that this 2D view results in some distortion of the polygon meshes, which do not exist in the 3D surface. Our seamless representation of the surface uv space enables the use of common 2D coverage patterns on it.

Other suitable 2D coverage patterns can be applied depending on the resulting uv grid to cover the surface. For example, for the SoT example surface, we applied two raster scan coverage patterns as shown in Fig. 11(a) and Fig. 12. The key point is that our generation of the uv grid allows evenly spaced application of (different) coverage patterns.

To produce a smooth coverage along a uniform coverage pattern on the surface S, the corresponding end-effector path $\mathbf{P}$ has non-uniform transitions from one configuration to the next. This is a result of Eq. (17), which relates the position and orientation constraints, such as the end-effector's position offset from the surface along its approach direction. This is important to produce a smooth and uniform surface coverage (application).

VIII. CONCLUSION

This paper introduced a general robotic approach for smooth and even coverage of 3D freeform surfaces of three types represented in polygonal mesh. For a given surface and a robot manipulator, it consists of all the necessary steps to assist a task-domain operator who is not a robotics expert to automatically (1) check the feasibility and (2) if feasible, generate a feasible and smooth coverage robot motion to provide even coverage of the surface that satisfies the task and manipulator constraints. One key aspect for ensuring even coverage is our novel method to automatically generate a uv grid of uniform cells directly on a freeform surface. We considered general task constraints in terms of related position and orientation constraints on a manipulator end-effector. Our automatic feasibility checking algorithm ensures the existence of a feasible manipulator path through efficient joint-space search and facilitates efficient planning of such a path following a coverage pattern on the uv grid. Our path planner finds a feasible manipulator path for a given coverage pattern through an orientation smoothing method.

The introduced methods have been implemented and tested on example surfaces and demonstrated the effectiveness of our general approach.

Future research includes expanding the approach to enable effective division of an arbitrary freeform surface into those well-defined surface types and the division of a large surface into patches, such that each patch allows feasible coverage by a continuous coverage motion. Applying the general approach to specific tasks, such as spraying, can further test the approach.

REFERENCES

[1] C. Chen et al., "A novel spiral trajectory for damage component recovery with cold spray," *Surf. Coatings Technol.*, vol. 309, pp. 719–728, Jan. 2017.
[2] W. Chen, J. Liu, Y. Tang, and H. Ge, "Automatic spray trajectory optimization on Bezier surface," *Electronics*, vol. 8, no. 2, pp. 168–184, 2019.
[3] M. V. Andulkar and S. S. Chiddarwar, "Incremental approach for trajectory generation of spray painting robot," *Ind. Robot, Int. J.*, vol. 42, no. 3, pp. 228–241, May 2015.

[4] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, "CAD-based automated robot trajectory planning for spray painting of free-form surfaces," *Ind. Robot, Int. J.*, vol. 29, no. 5, pp. 426–433, Oct. 2002.

[5] T. Girbacia, F. Girbacia, and G. Mogan, "Virtual planning of robot trajectories for spray painting applications," *Appl. Mech. Mater.*, vol. 658, pp. 632–637, Oct. 2014.

[6] G. Trigatti, P. Boscariol, L. Scalera, D. Pillan, and A. Gasparetto, "A new path-constrained trajectory planning strategy for spray painting robots," *Int. J. Adv. Manuf. Technol.*, vol. 98, pp. 2287–2296, Oct. 2018.

[7] H. Chen, T. Fuhlbrigge, and X. Li, "A review of CAD-based robot path planning for spray painting," *Ind. Robot, Int. J.*, vol. 36, no. 1, pp. 45–50, Jan. 2009.

[8] G. Liu, X. Sun, Y. Liu, T. Liu, C. Li, and X. Zhang, "Automatic spraying motion planning of a shotcrete manipulator," *Intell. Service Robot.*, vol. 15, no. 1, pp. 115–128, Mar. 2022.

[9] X. Ye, L. Luo, L. Hou, Y. Duan, and Y. Wu, "Laser ablation manipulator coverage path planning method based on an improved ant colony algorithm," *Appl. Sci.*, vol. 10, no. 23, pp. 8641–8660, 2020.

[10] V. Champagne and D. Helfritch, "Critical assessment 11: Structural repairs by cold spray," *Mater. Sci. Technol.*, vol. 31, no. 6, pp. 627–634, Apr. 2015.

[11] A. Moridi, S. M. H. Gangaraj, S. Vezzu, and M. Guagliano, "Number of passes and thickness effect on mechanical characteristics of cold spray coating," *Proc. Eng.*, vol. 74, pp. 449–459, Jan. 2014.

[12] C. H. Chen and K. T. Song, "Complete coverage motion control of a cleaning robot using infrared sensors," in *Proc. IEEE Int. Conf. Mechatronics*, Taipei, Taiwan, Jul. 2005, pp. 543–548.

[13] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robot. Auto. Syst.*, vol. 59, no. 10, pp. 801–812, Oct. 2011.

[14] J. Song and S. Gupta, "ε^*: An online coverage path planning algorithm," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 526–533, Apr. 2018.

[15] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[16] 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.

[17] A. Bouman et al., "Adaptive coverage path planning for effecient exploration of unknown environments," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*, Kyoto, Japan, 2022, pp. 11916–11923.

[18] K. Schmid, H. Hirschmuller, A. Domel, I. Grixa, M. Suppa, and G. Hirzinger, "View planning for multi-stereo 3D reconstruction using an autonomous multicopter," *J. Intell. Robotic Syst.*, vol. 65, nos. 1–4, pp. 309–323, 2012.

[19] E. N. Johnson and J. G. Mooney, "A comparison of automatic nap-of-the-earth guidance strategies for helicopters," *J. Field Robot.*, vol. 31, no. 4, pp. 637–653, Jul. 2014.

[20] M. Na, H. Jo, and J.-B. Song, "CAD-based view planning with globally consistent registration for robotic inspection," *Int. J. Precis. Eng. Manuf.*, vol. 22, no. 8, pp. 1391–1399, Aug. 2021.

[21] J. Jin and L. Tang, "Coverage path planning on three-dimensional terrain for arable farming," *J. Field Robot.*, vol. 28, no. 3, pp. 424–440, May 2011.

[22] L. C. Santos, F. N. Santos, E. J. S. Pires, A. Valente, P. Costa, and S. Magalhães, "Path planning for ground robots in agriculture: A short review," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2020, pp. 61–66.

[23] P. Atkar, H. Choset, A. Rizzi, and E. Acar, "Exact cellular decomposition of closed orientable surfaces embedded in R," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, Seoul, (South) Korea, 2001, pp. 699–704.

[24] T. Yang, J. V. Miro, Q. Lai, Y. Wang, and R. Xiong, "Cellular decomposition for nonrepetitive coverage task with minimum discontinuities," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 4, pp. 1698–1708, Aug. 2020.

[25] P. Atkar, H. Choset, and A. Rizzi, "Towards optimal coverage of 2-dimensional surfaces embedded in R: Choice of start curve," in *Proc. IROS*, Taipei, Taiwan, 2003, pp. 3581–3587.

[26] Z. Kingston, M. Moll, and L. Kavraki, "Decoupling constraints from sampling-based planners," in *Proc. Int. Symp. Robot. Res.*, Puerto Varas, Chile, 2017, pp. 1151–1178.

[27] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, USA, Nov. 2007, pp. 3074–3081.

[28] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 625–632.

[29] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 105–117, Feb. 2013.

[30] T. McMahon, S. Thomas, and N. M. Amato, "Sampling-based motion planning with reachable volumes: Theoretical foundations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 6514–6521.

[31] C. Yuksel, S. Lefebvre, and M. Tarini, "Rethinking texture mapping," *Comput. Graph. Forum*, vol. 38, no. 2, pp. 535–551, May 2019.

[32] B. Purnomo, J. D. Cohen, and S. Kumar, "Seamless texture atlases," in *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Process.*, Jul. 2004, pp. 65–74.

[33] N. Aigerman, R. Poranne, and Y. Lipman, "Seamless surface mappings," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–13, Jul. 2015.

[34] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human–robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, 2017.

[35] M. C. Bingol and O. Aydogmus, "Practical application of a safe human–robot interaction software," *Ind. Robot, Int. J. Robot. Res. Appl.*, vol. 47, no. 3, pp. 359–368, Jan. 2020.

[36] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Hierarchical segmentation of surfaces embedded in $\mathbb{R}^3$ for auto-body painting," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 572–577.

[37] C. Cao, J. Zhang, M. Travers, and H. Choset, "Hierarchical coverage path planning in complex 3D environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3206–3212.

[38] A. Saric, J. Xiao, and J. Shi, "Robotic surface assembly via contact state transitions," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Madison, WI, USA, Aug. 2013, pp. 954–959.

[39] S. McGovern and J. Xiao, "Efficient feasibility checking on continuous coverage motion for constrained manipulation," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*, Lyon, France, Aug. 2021, pp. 189–195.

[40] S. McGovern and J. Xiao, "UV grid generation on 3D freeform surfaces for constrained robotic coverage path planning," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, Mexico City, Mexico, Aug. 2022, pp. 1503–1509.

[41] L. Nielson, I. Sung, and P. Nielson, "Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges," *Sensors*, vol. 19, no. 19, pp. 4165–4179, 2019.

[42] M. Ramesh, F. Imeson, B. Fidan, and S. Smith, "Optimal partitioning of non-convex environments for minimum turn coverage planning," in *Proc. IROS*, Kyoto, Japan, 2022, pp. 4529–4536.

Sean McGovern received the M.S. degree in robotics engineering from Worcester Polytechnic Institute, Worcester, MA, USA, in 2016, and the M.S. degree in computer science from the Georgia Institute of Technology, Atlanta, GA, USA, in 2021. He is currently pursuing the Ph.D. degree in robotics engineering with Worcester Polytechnic Institute.

Jing Xiao (Fellow, IEEE) received the Ph.D. degree in computer, information, and control engineering from the University of Michigan, Ann Arbor, MI, USA. She is currently the Deans' Excellence Professor, the William B. Smith Distinguished Fellow of Robotics Engineering, a Professor of Computer Science, and the Head of the Robotics Engineering Department, Worcester Polytechnic Institute (WPI). She joined WPI from the University of North Carolina at Charlotte. Her current research interests include robotics, haptics, and intelligent systems. She has coauthored a monograph, held one patent, and published extensively in major robotics journals, conferences, and books. She was a recipient of the 2015 Faculty Outstanding Research Award of the College of Computing and Informatics, University of North Carolina at Charlotte.