2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)
August 23-27, 2021. Lyon, France

Efficient Feasibility Checking on Continuous Coverage Motion for Constrained Manipulation

Sean McGovern[1] and Jing Xiao[1]

2021 IEEE 17th International Conference on Automation Science and Engineering (CASE) | 978-1-6654-1873-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/CASE49439.2021.9551685

***Abstract*— Many industrial robotic applications require a manipulator to move the end-effector in a constrained motion to cover a surface region, including painting, spray coating, abrasive blasting, polishing, shotcreting. etc. The manipulator has to satisfy both task constraints imposed on the end-effector (such as maintaining certain distance and angle with respect to the target surface while traversing it) and manipulator joint constraints. Given a robot manipulator and a target surface patch, an important question is whether there exists a feasible path for the manipulator to move continuously along the surface patch to cover it entirely while satisfying both manipulator and task constraints. This question is largely open as it has not been addressed systematically, even though there is substantial literature on path planning of constrained manipulation motion. In this paper, we introduce a general and efficient method to provide answers to this question.**

I. INTRODUCTION

There are many robotic tasks that require a robot manipulator to traverse its end-effector along a constraining surface smoothly and satisfy some task constraints. Autonomous spray coating is such a task. It requires the robot end-effector to cover an entire surface while satisfying task criteria in terms of spray thickness, cycle time, and material waste [1], which translate to constraints on the configuration of the end-effector during traversal.

Surface coverage is often treated as a coverage path planning problem [2] on 3D surfaces with constraints. Many papers focus on CPP for mobile robots [3]–[6], while a few also consider CPP for manipulators [7], [8].

There is also substantial work on constrained manipulator motion planning [9]–[13], where the focus is mostly on finding a feasible, collision-free path connecting two configurations while keeping the end-effector constrained along the way. There is often an underlying assumption that such a path exists.

In the context of applications, there are many papers focusing on specific manipulator coverage tasks, such as on spray painting trajectories [14]–[18] and path planning [8] [19], shotcrete, [20], and laser ablation [21].

However, there is little research focused on the fundamental question of whether there exists a feasible manipulator path that enables the end-effector to cover an entire target surface continuously, satisfying given task constraints and manipulator constraints. The answer to this question will provide insights to whether the manipulator is suitable for the given task of covering the specific surface and under what conditions (such as relative spatial arrangements between the manipulator and the surface), whether there is a need to divide the surface patch into sub-patches and how to perform the division, or whether there is a need to change the manipulator.

In this paper, we introduce a general and efficient method to answer the open question. Our approach converts the task constraints on robot end-effector position and orientation into constraints that directly relate manipulator joint parameters and variables to task parameters (such as surface location, shape, and size) and task-specific requirements, e.g., spray nozzle offset and spray angle for a spraying job. Next our method explores the joint space that satisfy those constraints as well as joint constraints continuously and informs the feasibility of traversal in the Cartesian space by forward kinematics. The connectivity of the Cartersian-space manifold informs whether there is a feasible solution.

The rest of the paper is as follows. In Section II, we introduce notations, assumptions, and formulate the problem. We describe the method in Section III. We present implementation and testing results and discussions in Section V and conclusions in Section VI.

II. ASSUMPTIONS, NOTATIONS, AND CONSTRAINT FORMULATIONS

We attach a coordinate frame to a target spatial surface S with reference position $[x_c, y_c, z_c]^T$ and orientation R_c, where R_c is the rotation matrix of the surface frame with respect to the world frame. S can be either (a) a discretized freeform surface with indices u and v or (b) a piece-wise parametric surface with parameters u and v with respect to the surface frame. For (b), it means that S can have a single parametric representation or different parametric representations for its different pieces connected together. Without losing generality, we simply denote every point (x_s, y_s, z_s) on S as a function of (bounded) u and v: $[x_s, y_s, z_s]^T = \mathbf{g}(u, v)$.

The robot end-effector has to satisfy a *position task constraint*, such that every end-effector position $[x_e, y_e, z_e]^T$ corresponds to a surface point on S to maintain a fixed distance d from the surface; thus the end-effector position is also a function of u and v:

$$[x_e, y_e, z_e]^T = c_p(u, v) \tag{1}$$

The end-effector also needs to satisfy an *orientation task constraint*:

$$R_e = c_o(u, v) \tag{2}$$

[1]The authors are affiliated with the Robotics Engineering Department, Worcester Polytechnic Institute. smmcgovern@wpi.edu, jxiao2@wpi.edu. This work is funded by US Army Research Lab Contract W911NF1920108.

where R_e is the rotation matrix of the end-effector. A common orientation constraint is that the end-effector (or the tool mounted to the end-effector) approaches the surface S in a certain direction (e.g., the normal direction).

Denote the end-effector approach vector as the unit approach vector $\mathbf{a}$ (often along the z axis of the end-effector) and the desired approach direction to the surface as $\mathbf{b}(u, v)$. Then the equation (2) can be expressed more specifically as:

$$\mathbf{a} \cdot \mathbf{b}(u, v) = 0 \tag{3}$$

The orientation constraint can be further relaxed a bit to allow $\mathbf{a}$ to deviate from $\mathbf{b}$ within a small angle α at each position. Thus, we have the following inequality constraint:

$$\mathbf{a} \cdot \mathbf{b}(u, v) \leq \cos \alpha \tag{4}$$

We denote E as the constrained position manifold for the end-effector to cover the surface S while satisfying position and orientation task constraints.

S can be discretized into a discrete (surface) grid with u and v being indices for the cells on the grid. Each cell's center is positioned at $\mathbf{g}(u, v) = [x_s, y_s, z_s]^T$ with respect to the surface frame.

We further discretize the end-effector manifold E to form a grid with indices u and v, corresponding to the discretized S. The resolution of the grid can be depending on the task constraints. After discretization, we call each cell on the manifold E an *E-cell*.

Given an n-dimensional robotic manipulator, we denote its link parameters as $\mathbf{l} = [l_1, l_2, ..., l_n]^T$, a joint space configuration as $\mathbf{q} = [q_1, q_2, ..q_n]^T$, and joint limits as $q_{min,i} \leq q_i \leq q_{max,i}$, $1 \leq i \leq n$. A sequence of joint-space configurations can be noted as $Q = \{\mathbf{q}_1, \mathbf{q}_2.., \mathbf{q}_m\}$.

With forward kinematics, the end-effector position and orientation can be computed as functions of link parameters and joint variables, expressed in the manipulator homogeneous transformation matrix from the end-effector to the base ${}^0_nT$. Now, by substituting end-effector positions and orientations with their expressions $f_p(\mathbf{l}, \mathbf{q})$ and $f_o(\mathbf{l}, \mathbf{q})$ respectively in terms of link parameters and joint variables into equations (1), (2), and inequality (4), we can obtain *joint-space task constraints* that directly relate joint variables to task parameters:

$$f_p(\mathbf{l}, \mathbf{q}) = c_p(u, v), \tag{5}$$

and

$$f_o(\mathbf{l}, \mathbf{q}) = c_o(u, v), \tag{6}$$

or a more relaxed orientation constraint:

$$f_a(\mathbf{l}, \mathbf{q}) \leq \cos \alpha. \tag{7}$$

III. Methodology

Now we describe our approach to address the fundamental question: given a surface patch and a robot manipulator, does there exist a feasible path for constrained coverage or not? Note that feasibility refers to the manipulator path satisfying continuously both task constraints, as introduced in Section II, and manipulator constraints, which depends on the structure and dimensions of the manipulator links, the types of joints, and the joint limits. Violation of manipulator constraints causes either no solution for inverse kinematics or singularity configurations that prevent the end-effector to move smoothly along the constraining manifold E. Hence, our approach checks for existence of a feasible path in the manipulator's joint space.

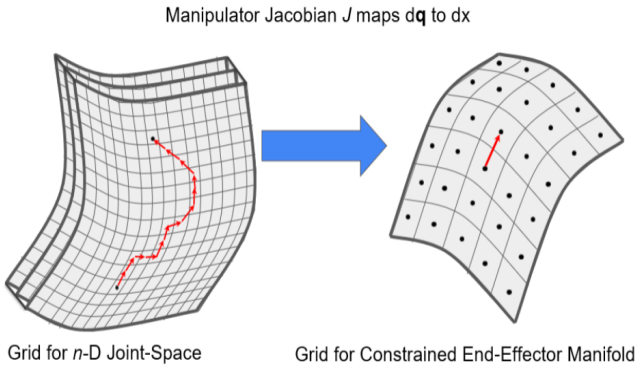


Fig. 1: Searching in joint space grid for feasible neighboring motion between E-cells.

A. Joint-space discretization and feasible J-cells

We first discretize the manipulator joint space into an n-dimensional grid and call each grid cell a *J-cell*. The joint-space distance between a J-cell and a neighboring J-cell is δq such that only one joint variable's value is increased or decreased by δq. As each joint variable's value can either remain unchanged, increase, or decrease δq, there are 3^n neighboring J-cells for each J-cell. A J-cell configuration is *feasible* if it satisfies both the joint-space task constraints, Equations (5) and (6) or Inequality (7), and the joint limits.

B. From feasible J-cell transitions to E-cell neighboring continuity

Next our method explores the feasible joint space manifold starting from a feasible J-cell jc_1 that corresponds to a feasible E-cell ec_1 on the constrained manifold E of the end-effector[1] by searching for feasible δq moves of the manipulator to the adjacent J-cells, and so on (Fig. 1). If such a move exists between jc_1 and a neighboring J-cell jc_2, then jc_2 is feasible and the transition between jc_1 and jc_2 is feasible. If, after some feasible neighboring J-cell transitions which forms a path Q, a J-cell jc_m is reached that corresponds to a neighboring E-cell ec_2 (by forward kinematics) of ec_1, then ec_2 is feasible, and there exists a feasible path Q to enable the manipulator to move from ec_1 and ec_2 smoothly (i.e., without singularity). We say that there is a *neighboring feasibility continuity* between ec_1 and ec_2.

Our method searches the discretized joint space for feasible J-cell transitions via a tree search. For each feasible

[1]If there are multiple candidates for the starting J-cell jc_1, the selection of jc_1 may affect the results of search. Thus, one can run our method with every possible candidate.

Algorithm 1: Continuity Check in Joint Space J for Neighboring E-$Cells$

Input J-cell jc with configuration $\mathbf{q}$, satisfying joint-space task constraints and joint limits; its corresponding E-cell is ec
input target neighboring E-cell ec_T with end-effector position $\mathbf{p}_T$;
call CONT(jc, ec_T) to obtain $Continuity$ and jc_T;
return jc_T and $Continuity$ between ec and neighboring ec_T, which is either $true$ or $false$.

procedure CONT(jc, ec_T)
$Continuity$=$false$;
repeat:
from jc, find an unchecked neighboring J-cell jc_N by adjusting a q_i of $\mathbf{q}$ by $\pm\delta q$ to obtain $\mathbf{q_N}$;
if $\mathbf{q_N}$ *satisfies joint-space task constraints and joint limits* **then**
  obtain corresponding E-cell ec_N via forward kinematics with end-effector position p_N;
  if $ec_N == ec_T$ **then**
    $Continuity$=$true$;
    return *continuity and* jc_N
  end
  if $\mathbf{p}_N$ *is closer to* $\mathbf{p}_T$ *than* $\mathbf{p}$ **then**
    call *CONT*(jc_N, ec_T)
  end
end
until there is no unchecked neighboring J-cell of jc.

Algorithm 2: Continuity Check on Manifold E

Input ec_1 and corresponding jc_1;
initialize Tree T at ec_1;
initialize graph G containing E-cells with no edges;
T = TREESEARCH(T, jc_1);
if G *is a connected component* **then**
  return "$\exists$ feasible path"
else
  return "no feasible path"
end

procedure TREESEARCH(T, jc)
ec is obtained from jc by forward kinematics;
$Neighbors$ = set of neighboring E-Cells of ec;
for all $ec_N \in Neighbors$ **do**:
if *no edge exists between* ec *and* ec_N *in* G **then**
  call Algorithm 1 with jc and ec_N to obtain $Continuity$ and jc_N;
  if $Continuity$ **then**
    Add edge between ec and ec_N if not in G.;
    if ec_N *is not in* T **then**
      Add ec_N as child to ec in T;
      T = TREESEARCH(T, jc_N)
    end
  end
end
return T

transition to a neighboring J-cell, we find the corresponding end-effector configuration achieved via instantaneous forward kinematics. That is, through searching feasibility in the joint-space, the search of feasibility in neighboring E-cell transitions in the end-effector space is achieved, as illustrated in Fig. 1. This method avoids solving inverse kinematics, which often requires numerical solutions for redundant manipulators, and greatly increases the efficiency.

C. Feasibility continuity on task-constrained manifold E

Starting from an E-cell ec_1, our method can establish whether there is a path of feasibility continuity, i.e., a continuously feasible path, between ec_1 and a non-neighboring E-cell by finding feasibility continuity to a neighboring E-cell ec_2 and repeating the process to find feasibility continuity from ec_2 to its neighboring cell ec_3, and so on. By conducting a tree search from ec_1 to cover the entire E, our method can establish a *continuity graph* on E to show all feasibility continuity among all E-cells.

Specifically, our method considers each neighboring E-cell of ec_1 in a systematic way, such as following the clockwise order from the upper-left neighboring cell of ec_1 on the manifold E. To check for feasibility continuity between two neighboring E-cells, Algorithm 1 is used. To find all feasibility continuity on manifold E, our method conducts a tree search as shown in Algorithm 2.

As the tree search traverses E, we add E-cells to the tree and also record the connectivity of E (which indicates continuity in constrained motion) in a graph, G. If the resulting G is a single connected component, we determine that there is at least one path that can cover the full surface of S while maintaining constraints. We can also analyze G to determine other feasible paths on E.

IV. Implementation and Testing Results

To test our method, we use a PUMA 560 robotic manipulator to determine feasibility for autonomous constrained manipulation coverage. We considered spray coating as an example constrained coverage problem. For this problem, the end-effector position task constraint is to keep the end-effector a constant distance d from the target surface as it traverses the surface, and the end-effector orientation task constraint is to maintain its approach vector within a small angle α from the normal of the surface.

The PUMA robot contains six revolute joints, with the joint vector $\mathbf{q} = [\theta_1, \theta_2, ..., \theta_6]^T$ and link parameters $\mathbf{l} = [a_2, a_3, d_3, d_4] = [0.7, 0.1, 0.1, 0.7]$(m). The manipulator transformation matrix ${}^0_6T$, see [22], and joint limits are used to obtain $f_p(\mathbf{l}, \mathbf{q})$, $f_o(\mathbf{l}, \mathbf{q})$, and $f_a(\mathbf{l}, \mathbf{q})$ in the joint-space task constraint equations (5)-(7). We considered several sets of joint limits for the PUMA, see Table I, and different task constraint parameter values for each surface, see Table II.

TABLE I: Joint Limits Sets for PUMA

Set 1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-180°$	$-180°$	$-180°$	$-180°$	$-180°$	$-180°$
θ_{max}	$180°$	$180°$	$180°$	$180°$	$180°$	$180°$
Set 2	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-170°$	$-170°$	$-180°$	$-135°$	$-100°$	$-90°$
θ_{max}	$170°$	$75°$	$175°$	$190°$	$90°$	$90°$
Set 3	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-90°$	$-170°$	$-180°$	$-135°$	$-100°$	$-90°$
θ_{max}	$10°$	$75°$	$90°$	$170°$	$90°$	$90°$
Set 4	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-110°$	$-110°$	$-110°$	$-150°$	$-45°$	$-180°$
θ_{max}	$145°$	$75°$	$40°$	$120°$	$140°$	$180°$

TABLE II: Task Parameters and Values

Set 1	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
All Surfaces	0	0	0	$45°$	5
Set 2	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
Sphere	0	0	0	$5°$	5
Cylinder	0	0.055	0	$5°$	5
Hyperboloid	1.095	0	0	$10°$	5
Set 3	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
Sphere	0.005	0	0	$10°$	5
Cylinder	0	0.055	0	$10°$	5
Hyperboloid	1.15	0	0	$20°$	5

TABLE III

Surface Parametric Equations					
S	u	v	x	y	z
Sphere	θ	ψ	r*$sin\theta$*$cos\psi$	r*$sin\theta$*$sin\psi$	r*$cos\theta$
Cylinder	θ	z	r*$cos\theta$	r*$sin\theta$	z
Hyperboloid	θ	z	$r(z)$*$cos\theta$	$r(z)$*$sin\theta$	z
Surface Parameter Ranges					
S	u min	u max	v min	v max	r
spherical	$10°$	$90°$	$10°$	$180°$	1m
cylindrical	-60°	$60°$	0	0.9m	0.5m
Hyperboloid	$130°$	$220°$	0	0.5m	$r(v)$
where $r(v_{min}) \leq r(v) \leq r(v_{max})$					

A. Parametric surfaces

We tested our method with various surface patches, including a spherical, a cylindrical, and a hyperbolic surface patch. The parametric representation of each surface, as shown in Table III, is expressed with respect to its origin $[x_c, y_c, z_c]^T$. For each surface frame, its origin is at the center (of symmetry) of the surface, and its orientation is aligned with the robot base frame.

The spherical surface is a quadrant of a hemisphere, and parameters u and v denotes the polar angle and azimuthal angle θ and ψ of the spherical coordinate system respectively. The cylinder and hyperboloid surfaces are constrained by u and v which denote angle θ (about z-axis of surface frame) and height z (with respect to origin) respectively.

B. Deriving joint-space task constraints

We derive the joint-space task constraints for each surface by first defining the task constraints for the end-effector in Cartesian space. To obtain equations (5)-(7), we substitute end-effector position $[x_e, y_e, z_e]^T$, as defined in equation (1), by their functions of joint parameters and variables from the PUMA manipulator transformation matrix:

$$\begin{aligned} x_e &= c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 \\ y_e &= s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 \\ z_e &= -a_3s_{23} - a_2s_2 - d_4c_{23} \end{aligned} \tag{8}$$

where $s_{123} = sin(\theta_1 + \theta_2 + \theta_3)$, $c_{123} = cos(\theta_1 + \theta_2 + \theta_3)$.

The detailed derivation for each surface used for testing can be found in Appendix.

C. Testing results with different sets of joint limits

We first tested the effects of different joint limits, while keeping the task parameter values as in the parameter set 1 in Table II. Fig. 2 shows an example of how Algorithm 1 searches the joint-space grid for a continuous constrained end-effector motion between two E-cells for the spherical surface, using the set 1 joint limits. Note that θ_1 and θ_2 are the horizontal and vertical axes here since PUMA's first three joint variables θ_1, θ_2, and θ_3, exclusively define the end-effector position and θ_1 and θ_2 are most significant (θ_3 does not change in this example). δq, which is $\delta\theta$ for this case, is $0.01°$. The figure shows that it took 36 $\delta\theta$ steps in the joint space to complete the neighboring transition in the Cartesian space. The first 35 J-cells correspond to one E-cell and the last J-cell corresponds to the neighboring E-cell.

Fig. 3(a) shows the graph G results from Algorithm 2 using joint limits set 1 for the spherical surface. In this case, G is a single connected component and with all nodes connected to their neighbors that covers all E-cells on manifold E. This means not only that there exists a feasible path for the robot to continuously move the end-effector to cover all the E-cells, but also that there are many feasible paths to do so. Fig. 3(b) shows the graph G using joint limits set 2 this time, resulting in one connected component (in blue) that does not include all E-cells in manifold E and single-cell components from the remaining E-cells (in red). The red dots show E-cells that cannot be visited by the robot end-effector in continuous constrained motion. This means that there is no feasible continuous motion path to cover all E-cells. Note how some E-cells in the (blue) connected component in Fig. 3(b) cannot reach some of their neighboring E-cells. This shows that there are limited directions of feasible motion to reach those E-cells, which is informative to constrained path planning.

We can also display the graph G directly on the manifold E, for a given surface. Fig. 4 illustrates the graph G on the manifold E for the spherical, cylindrical, and hyperbolic surfaces using two different joint limit sets in Table I. For each surface, when joint limit set 1 is applied, the graph G is a connected component, as shown in the left figure. However, when joint limit set 2 is applied, the graph G does not have a connected component that covers all E-cells, as shown in the right figure, i.e., there is no feasible motion path to traverse the entire E in those cases.

D. Testing results with different task parameter values

Different task parameter values can also affect the feasibility of a coverage path. Fig. 5 illustrates graphs G on

the manifold E for the spherical, cylindrical, and hyperbolic surfaces using joint limit set 1 and task parameter value set 2 then set 3 respectively. Using joint limit set 1 with the spherical surface, which previously resulted in a single connected component in G (see Fig. 4(a)), Fig. 5 (a) and (b) show that decreasing α can greatly reduce the size of the multi-cell connected component in G. These figures also show that when the sphere center is moved a very small distance along the x-axis of the robot base frame, the multi-cell connected component in G shrinks much more even though α is increased. In this case, the spatial arrangement of the surface relative to the robot seems to have a more significant effect on feasibility than the α value.

Similarly, using joint limit set 1 with the cylindrical surface, previously resulted in a G of a single connected component (see Fig. 4(c)), but in Fig. 5 (c), moving the surface along the y-axis of the robot base frame for a large distance reduced half of the E-cells from the (multi-cell) connected component of G. After changing α from 5° to 10°, a relatively small amount, we again see a graph G with a single connected component (Fig. 5 (d)). In this case, the spatial arrangement of the cylinder with respect to the robot does not seem to have as much of an effect on feasibility than the value of α.

There are also scenarios where a change in parameter values can result in a loss of E-cells and edges in one section of the connected component in G while gaining E-cells and edges in another part of the connected component. This is visibly noticeable from changes in Fig. 5 (e) and (f). The multi-cell connected component of G looses edges near the center of the hyperboloid (where the curvature is greatest) as the surface moves further away from the robot base along the x-axis,and at the same time, E-cells near the corner of the hyperboloid are added to the connected component of G.

The testing results show that the effects of different parameter values on feasibility of continuous coverage can vary from case to case and may be inter-related, which confirms the need for a systematic approach and the utility of our method to determine whether there is a feasible path to cover a surface continuously given different manipulator and task constraints.

E. Performance data

Each test case had a runtime of building graph G well below 1 second. For example, the runtime of building the graph G for the spherical surface with joint limit set 1 and task parameter set 1 was 540 ms. The example case contained 90 E-cells and called Algorithm 1 for 305 times. Hence, the average joint-space search time between two neighboring E-cells is less than 2 ms. This example represents the worst-case scenario where there is a single connected component in G with the maximum number of edges. For cases where G contains multiple components (such as in Fig. 3(b)), i.e., there is no feasible solution to cover the entire manifold E, the runtime is much lower still. The runtime of Algorithm 2 will increase as the resolution of the grid for manifold E increases, but the average joint-space search time between two neighboring E-cells will decrease at the same time.

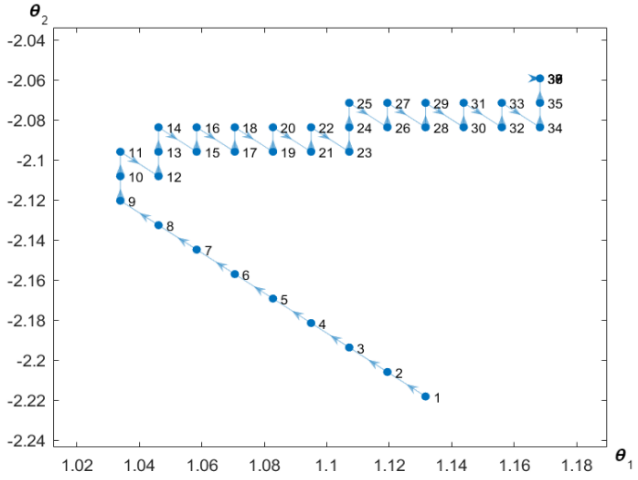
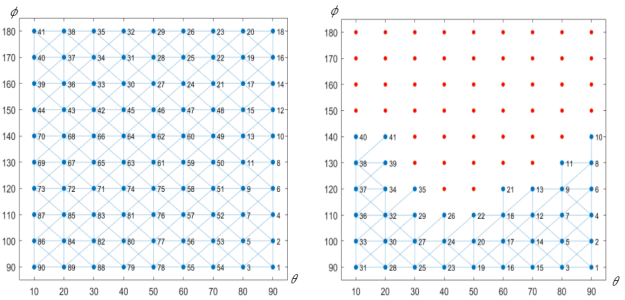


Fig. 2: Joint-space motion continuity search between two neighboring E-cells for the spherical surface (in degrees).



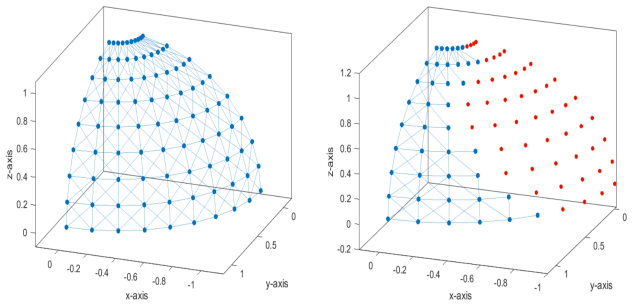
(a) based on joint limit set 1 (b) based on joint limit set 2

Fig. 3: Graph G for the spherical surface using task parameter value set 1 and different joint limits, where red dots indicates unreachable nodes.

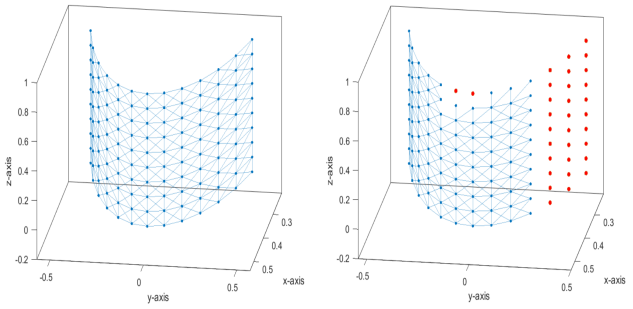
V. Conclusions

Although there are many applications that require continuously moving a robot manipulator's end-effector across a surface patch under some task constraints, the question whether this is always possible, given a specific manipulator, target surface, and task constraints, has not been answered systematically. This paper has introduced an efficient and general approach to address this question systematically, based on converting task constraints from the Cartesian space to the joint space and searching for feasible joint-space paths directly without inverse kinematics, which can be computationally expensive, especially for redundant manipulators.

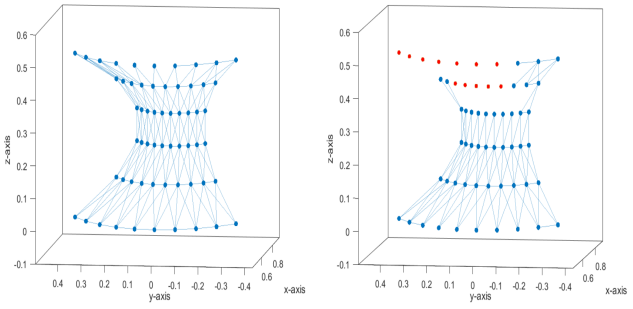
The introduced algorithms have been implemented and tested on example tasks involving a PUMA robot and parametric surface patches. The results have shown that manipulator joint limits and values of task constraint parameters both can greatly affect whether or not there is a feasible solution, i.e., a continuous path for the robot manipulator to cover the target surface patch under task constraints. If the answer is yes, the graph generated by applying the introduced



(a) spherical, joint limit set 1 (b) spherical, joint limit set 2

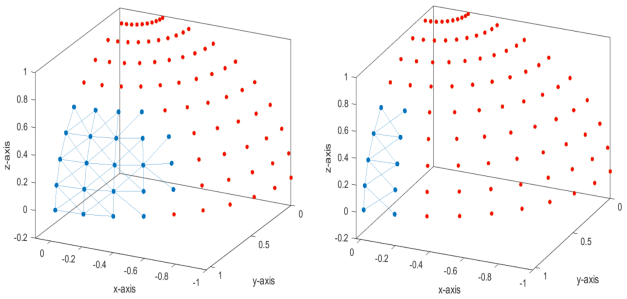


(c) cylindrical, joint limit set 1 (d) cylindrical, joint limit set 3

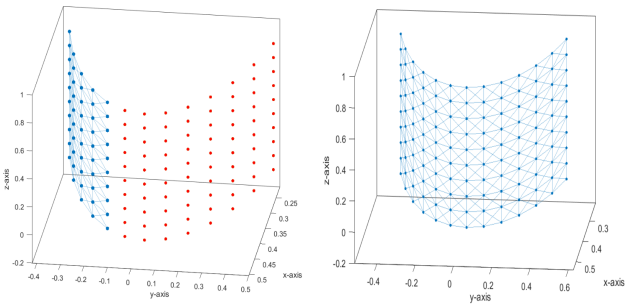


(e) hyperbolic, joint limit set 1 (f) hyperbolic, joint limit set 4

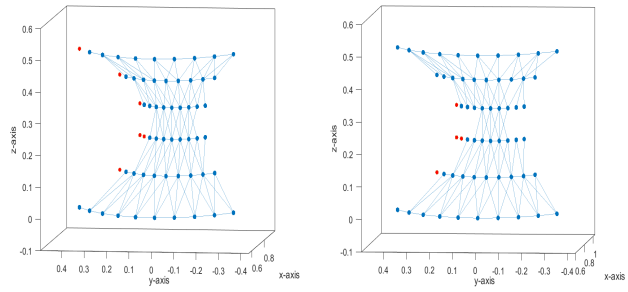
Fig. 4: Graph G on manifold E of different surface patches using task parameters value set 1 and different PUMA joint limit sets.



(a) spherical, α=5° (b) spherical, α=10°, x_c=5mm



(c) cylindrical, α=5°, y_c=55mm (d) cylindrical, α=10°, y_c=55mm



(e) hyperbolic, α=10°, x_c=1.095m (f) hyperbolic, α=20°, x_c=1.15m

Fig. 5: Graph G on manifold E for different surface patches using joint limit set 1 and varied values of task constraint parameters.

method can further convert the problem of planning feasible manipulator paths for coverage into a much simpler graph search problem. If the answer is no, the method can help revealing the cause(s) of infeasibility, which could range from mismatched manipulator and task to mis-positioned target surface with respect to the manipulator. If the latter is the cause, the method can be used to adjust the relative position and pose of the surface with respect to the manipulator to find the relative arrangement that results in a feasible coverage solution. Alternatively, the method can be used to determine an acceptable manipulator for a given task.

We plan to further test the approach on different manipulators and for different tasks and constraints and to develop autonomous robotic solutions for manipulation tasks involving surface coverage for many applications.

APPENDIX

In the following, we show the derivation of joint-space task constraints for three surfaces: spherical, cylindrical, and hyperbolic, as shown in Table III, assuming that the robot base is on the same side of the surface as the surface center (of symmetry).

A. *Spherical surface*

For the spherical surface, the position task constraint, as presented in equation (1), can be found by subtracting the radius r by distance d, which is the end-effector distance from the surface:

$$(x_e - x_c)^2 + (y_e - y_c)^2 + (z_e - z_c)^2 = (r - d)^2 \qquad (9)$$

The corresponding joint-space task constraint (5) can be expressed by replacing x_e, y_e, z_e with their joint functions in equation (8).

The orientation task constraint of inequality (7) can be expressed in terms of the dot product of the unit vector along the end-effector z axis $\mathbf{a}$ and the normal vector $\mathbf{b}$ at each

position on the constrained manifold E, such that:

$$\mathbf{a} = [r_{13}, r_{23}, r_{33}]^T \tag{10}$$

where r_{**} is from the rotation matrix of the manipulator transformation matrix, and:

$$\begin{bmatrix} x_e - x_c \\ y_e - y_c \\ z_e - z_c \end{bmatrix} = \vec{b}_s, \quad \mathbf{b} = \frac{\vec{b}_s}{\|\vec{b}_s\|}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{11}$$

B. Cylindrical surface

For the cylindrical surface, the position task constraint is as follows:

$$(x_e - x_c)^2 + (y_e - y_c)^2 = (r - d)^2 \tag{12}$$

The corresponding joint-space task constraint (5) can be expressed by replacing x_e, y_e, z_e with their joint functions in equation (8).

The *orientation task constraint* of inequality (7) can be expressed in terms of the dot product of the unit vector along the end-effector a axis $\mathbf{a}$ and the normal vector $\mathbf{b}$ at each end-effector position on the constrained position manifold E such that $\mathbf{a}$ is as defined in equation (10) and:

$$\begin{bmatrix} x_e - x_c \\ y_e - x_c \\ 0 \end{bmatrix} = \vec{b}_s, \quad \mathbf{b} = \frac{\vec{b}_s}{\|\vec{b}_s\|}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{13}$$

C. Hyperbolic surface

For the hyperbolic surface, the end-effector position task constraint is shown in equation (14). This is similar to the position task constraint for the cylinder, except that the radius, r, is a function of coordinate z of the surface frame:

$$(x_e - x_c)^2 + (y_e - y_c)^2 = [r(z) - d]^2. \tag{14}$$

The orientation task constraint of inequality (7) can be expressed in terms of the dot product of the unit vector along the end-effector z axis $\mathbf{z}$ and the unit normal vector $\mathbf{b}$ at each end-effector position on the constrained position manifold E such that $\mathbf{a}$ is as defined in equation (10) and:

$$x = x_e - x_c, y = y_e - y_c, z = z_e - z_c$$

so that $[x, y, z]^T$ indicates the end-effector position expressed in the surface frame. The unit normal vector $\mathbf{b}$ can be expressed in the following:

$$\mathbf{b} = \frac{\mathbf{c_1} \times \mathbf{c_2}}{\sqrt{\|\mathbf{c_1}\|^2 \|\mathbf{c_2}\|^2 - \|\mathbf{c}_1 \cdot \mathbf{c}_2\|^2}}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{15}$$

where

$$\mathbf{c_1} = \begin{bmatrix} -y \\ x \\ 0 \end{bmatrix}, \mathbf{c_2} = \begin{bmatrix} \frac{\partial r(z)}{\partial z} \frac{x}{r(z)} \\ \frac{\partial r(z)}{\partial z} \frac{y}{r(z)} \\ 1 \end{bmatrix} \tag{16}$$

REFERENCES

[1] H.Chen, T. Fuhlbrigge, X. Li, "A review of CAD-based robot path planning for spray painting," *Industrial Robot: An International Journal*, vol. 36, no. 1, pp. 45-50, 2009.

[2] E. Galceran, M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258-1276, 2013.

[3] Chen, C.H.; Song, K.T, "Complete coverage motion control of a cleaning robot using infrared sensors," in *Proc. IEEE International Conference on Mechatronics (ICM)*, Taipei, Taiwan, 10–12 July 2005, pp. 543–548.

[4] L. Santos, F. Santos,S. Pires, E.J. Pires, A. Valente, P. Costa, "Planning for ground robots in agriculture: A short review," in *Proc. of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Ponta Delgada, Portugal, 15–17 April 2020; pp. 61–66.

[5] T. Lee, S. Baek, Y. Choi, S. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 801-812, 2011.

[6] M. Hassan and D. Liu, "PPCPP: A Predator–Prey-Based Approach to Adaptive Coverage Path Planning," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 284-301, 2020.

[7] P. Atkar, H. Choset, A. Rizzi, E. Acar, "Exact Cellular Decomposition of Closed Orientable Surfaces Embedded in R," *International Conference on Robotics and Automation*, vol. 1, pp. 699 - 704, 2001.

[8] P. Atkar, H. Choset, A. Rizzi, "Towards Optimal Coverage of 2-Dimensional Surfaces Embedded in R: Choice of start Curve," in *Proc. International Conference on Intelligent Robotics and Systems*, October 2003, vol. 4, pp. 3581-3587.

[9] Z. Kingston, M. Moll, and L. E. Kavraki, "Decoupling constraints from sampling-based planners," in *Proc. Int. Symp. of Robot.* Res., 2017, vol. 38, no. 10-11, pp. 1151-1178.

[10] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[11] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," *Proc. IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 625–632.

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

[13] T. McMahon, S. Thomas, and N. M. Amato, "Sampling-based motion planning with reachable volumes: Theoretical foundations," in *Proc. IEEE International Conference on Robotics and Automation*, 2014, pp. 6514–6521.

[14] C. Chen, S. Gojon, Y. Xie, S. Yin, C. Verdy, Z. Ren, H. Liao, S. Deng, "A novel spiral trajectory for damage component recovery with cold spray," *Surface and Coatings Technology 309*, vol. 309, pp. 719-728, 2017.

[15] W. Chen, J. Liu, Y. Tang, H. Ge, "Automatic Spray Trajectory Optimization on Bezier Surface," *Electronics*, vol.8, no. 2, pp. 168-184, 2019.

[16] M. Andulkar, S. Chiddarwar,"Incremental approach for trajectory generation of spray painting robot," *Industrial Robot: An International Journal*, vol. 42, no. 3, pp.228-241, 2015.

[17] H.Chen, N.Xi, W. Sheng, M. Song, Y. Chen, "CAD-based automated robot trajectory planning for spray painting of free-form surfaces," *Industrial Robot: An International Journal*, vol. 29, no.5, pp. 426-433, 2002.

[18] G. Teodora, G. Florin, M. Gheorghe, "Virtual Planning of Robot Trajectories for Spray Painting Applications," *Applied Mechanics and Materials*, vol. 658, pp. 632-637, 2014.

[19] G.Trigatti, P.Boscariol, L. Scalera, D. Pillan, A. Gasparetto, "A new path-constrained trajectory planning strategy for spray painting robots," *The International Journal of Advanced Manufacturing Technology*, vol. 98, pp. 2287-2296, 2018.

[20] G. Liu, X. Sun, Y. Liu, C. Li, X. Zhang, "Automatic spraying motion planning of a shotcrete manipulator," *Intelligent Service Robotics (2021)*, https://doi.org/10.1007/s11370-021-00348-9.

[21] X. Ye, L. Lui, L. Hou, Y. Duan, Y. Wu, "Laser Ablation Manipulator Coverage Path Planning Method Based on an Improved Ant Colony Algorithm," *Applied Sciences*, vol.10, no. 23, pp.8641, 2020.

[22] John J. Craig, *Introduction to Robotics: Mechanics and Control* 3rd ed.., Pearson Education, Inc. 2005.