2016 IEEE International Conference on Automation Science and Engineering (CASE)
Fort Worth, TX, USA, August 21-24, 2016

Planning Algorithms for Multi-Setup Multi-Pass Robotic Cleaning with Oscillatory Moving Tools

Ariyan M. Kabir† and Joshua D. Langsfeld† and Shaurya Shriyam† and Vinaichandra Sai Rachakonda† and Cunbo Zhuang† and Krishnanand N. Kaipa† and Jeremy Marvel‡ and Satyandra K. Gupta††

***Abstract*— We present planning algorithms for cleaning stains on a curved object. Removing the stain may require multiple reorientations and repositions of the object and some portions of the stain may require multiple cleaning passes. The experimental setup involves two robot arms. The first arm immobilizes the object. The second arm moves the cleaning tool. The algorithm analyzes the stain and determines the sequence of positions and orientations needed to clean the part based on the kinematic constraints of the robot arm. Our algorithm uses a *depth-first branch-and-bound* search to generate setup plan solutions. We also compute the cleaning trajectories and select the cleaning parameters to maximize the cleaning performance. The algorithm generates multi-pass trajectories by replanning based on the observed cleaning performance. Numerical simulations and cleaning experiments with two Kuka[1] robots are used to validate our approach.**

I. Introduction

Cleaning parts and tools is a common process in many applications like construction, maintenance, service, manufacturing, healthcare, and food processing. Depending upon the nature of the cleaning task, different modes of cleaning may be used. In some situations cleaning fluid is applied to the surface to either dissolve foreign particles or force the particles to separate from the surface. Cleaning kitchen utensils is an example of such a cleaning mode. Many cleaning tasks require application of an oscillatory moving cleaning tool (often with abrasive particles embedded in the cleaning surface) over the surface being cleaned. In this cleaning mode, foreign particles are removed using mechanical erosion. This paper deals with cleaning tasks that utilize an oscillatory moving cleaning tool that makes mechanical contact with the part being cleaned.

Consider the task of removing rust (or sanding off paint before re-painting) from a non-planar surface in a remanufacturing application. This task is typically done manually and requires a lot of effort on the part of the operator. This seemingly simple task is hard to automate for the following reasons:

1) It is difficult to clean the entire part from one orientation. The part needs to be often moved and regrasped to ensure that the cleaning tool can access the entire part surface.

†Maryland Robotics Center, University of Maryland, MD USA ‡National Institute of Standards and Technology, MD USA ††Center for Advanced Manufacturing, University of Southern California, CA USA [guptask]@usc.edu

[1]DISCLAIMER: Any commercial product or company name in this paper is given for informational purposes only. Their use does not imply recommendation or endorsement by NIST or the University of Maryland or the University of Southern California

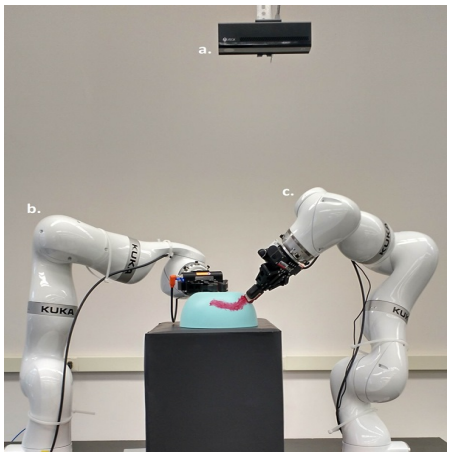

Fig. 1. Robotic setup built with two Kuka robots and Microsoft Kinect: (a) Kinect. (b) Holding robot arm. (c) Cleaning robot arm.

2) The cleaning progress needs to be constantly monitored and the plan needs to be often modified to ensure efficient cleaning. This requires a sensor-based feedback loop.
3) Cleaning with mechanical action requires application of force. Often the parts being cleaned cannot withstand arbitrarily large forces. This requires that the force being applied during the cleaning process be monitored and controlled to make sure that the part being cleaned is not physically damaged due to the application of excessive cleaning force.
4) Cleaning time is considered as a non-value-added time in manufacturing applications. Hence, it needs to be minimized by carefully selecting cleaning process parameters.
5) Cleaning of complex geometries requires complex cleaning motions.

Recent trends in perception and planning have enabled use of robots in highly non-repetitive tasks. Representative examples include kitting [1], [2], bin-picking [3], [4], [5], [6], assembly [7], and cleaning [8], [9], [10]. Traditionally, non-repetitive cleaning tasks requiring mechanical contact with a moving tool are done manually. Advances in robotics make it feasible to consider industrial robots in cleaning applications. Bi-manual setups enable one arm to hold and immobilize the object being cleaned. The second arm can be used to perform the cleaning. The bi-manual setup enables the frequent reorientation of the part without requiring part-

specific fixtures and hence offers flexibility in non-repetitive tasks to deal with a wide variety of geometries. Many robotic manipulators now include integrated high resolution force sensor and support impedance control. This enables safe cleaning operations without the risk of damaging the part. Three dimensional (3D) sensors enable estimation of the shape of the part being cleaned and enable generation of cleaning trajectories on the fly.

In this paper, we describe planning algorithms for cleaning stains on curved objects. The stain being cleaned may require multiple repositions and/or reorientations of the part and some portions of the stain may require multiple cleaning passes. The number of cleaning passes is determined by the intensity of the stain. The cleaning is performed with a moving tool with an abrasive surface. The experimental setup involves two robot arms. The first arm immobilizes the object. The second arm moves the cleaning tool. Figure 1 shows the experimental setup used to implement the results of the planning algorithm. The algorithm analyzes the stain and determines the sequence of poses (positions and orientations) needed to clean the part based on the kinematic constraints of the robot arm. Each pose is called a cleaning setup in this paper. For each cleaning setup, we also compute the cleaning trajectories by computing the tool paths and select the cleaning parameters to maximize the cleaning performance. The algorithm is capable of generating multi-pass trajectories. We use replanning to refine the plan based on the observed cleaning performance.

To the best of our knowledge, planning of cleaning tasks by changing object's pose based on robot's reachability has not been considered so far. Our focus is on problems involving hard stains that typically require multiple passes of mechanical scrubbing over the same surface area and setup change for complete coverage.

II. Related Work

Robotic cleaning has been attracting a growing interest from the research community in the recent years. Prior work has focused on different aspects of the problem including coverage path planning [11], perception [12], control [13], [14], [15], and learning [16], [17]. Most research considered cleaning with manipulators [11], [13], [14], [15], [16], [17], while some considered cleaning with mobile robots [12]. Approaches differed based on factors like the type of stains involved, shape/material of the target surface to be cleaned, and the environments in which the task was performed.

Hess et al. [11] addressed the coverage path planning for cleaning 3D surfaces using redundant manipulators. They explicitly considered the null space by treating different inverse kinematics solutions as individual nodes in a graph and modeled the problem as a generalized traveling salesman problem where the nodes of the graph are subdivided into clusters and at least one node in each cluster needs to be visited. They showed that their method outperformed Euclidean coverage algorithms in terms of manipulation effort and completion time. Sato et al. [13] presented a trajectory/force tracking controller, in a mixed position/torque control mode, for a standing humanoid robot using its right arm to clean a dry erase marker on a white board. Martinez et al. [18] presented an approach to plan high-level manipulation actions for the cleaning task. They considered stains like dry erase marker on white board and lentils (seeds) on a flat surface. They addressed changes in dirt distributions during dragging actions used while cleaning by replanning every few actions with newer perceptions. They also integrated a learning component in order to provide adaptation to changes such as different rag grasps, robots, or cleaning surfaces. Nagata et al. [15] were able to use a force and position controlled manipulator to perform mold polishing on a curved surface. King et al. [14] used equilibrium point control to generate wiping behaviors for a robot giving bed baths to patients. They considered powdered candy with food color as the stains to be cleaned. Hermann et al. [19] developed algorithms to automatically generate tool trajectories in real time for Computer Aided Design (CAD) applications. These algorithms can be used to plan trajectory on curved surfaces for robotic cleaning applications. Machine learning methods have also been applied to robotic cleaning tasks. Eppner et al. [16] developed an imitation learning framework through the use of a dynamic Bayesian network for the task of cleaning a whiteboard. Gams et al. [17] developed a two-layer framework that models a human through vision and then uses force profiling to wipe a kitchen table. In our earlier works [10], [8], [9], we used a semi-supervised learning approach to learn optimal operation parameters and a part deformation model for robotic cleaning.

III. Problem Formulation

We define the task for the robot as cleaning stains on an arbitrary curved surface $\Gamma \in \mathbb{R}^3$. Let $\ell \in \mathbb{R}^6 = \{x, y, z, \alpha, \beta, \gamma\}$ represent a general pose where (x, y, z) and (α, β, γ) represent the position and orientation, respectively in 3D. Let $\Gamma(\ell)$ represent the target surface oriented in an arbitrary pose ℓ. We approximate the stain on the surface as a set of small discrete stain patches $\mathcal{P} = \{p_i : i = 1, 2, \ldots, n\}$. Each patch p_i is a small planar triangle with an area $a_i \leq a_m$, where a_m is the surface area of the tip of the cleaning tool. Figure $2(a)$ demonstrates an example of a surface after triangulation. The red region represents $\mathcal{P}$.

We assume that the stain intensity is not uniform across the surface and that a single pass may not be able to clean the entire stain region completely. Let N_i represent the number of cleaning passes required to remove the stain from patch p_i. The number of passes is determined by image processing explained in section VII. We restrict the robot's motion such that its tool axis is close to the surface normal and the sweeping motions are orthogonal to the surface normal. The robot may fail to satisfy these conditions for some segments of $\mathcal{P}$, for some $\Gamma(\ell) \in \mathcal{F}$, where $\mathcal{F} \subset \mathbb{R}^6$ is the set of all such poses. For each $\Gamma(\ell)$, we can test how many patches can be reached by the robot by solving its inverse kinematics. This reachability problem can be solved by changing $\Gamma(\ell)$ in steps such that all the subsets of the target surface fall in the robot's reachability space atleast once. Therefore, we

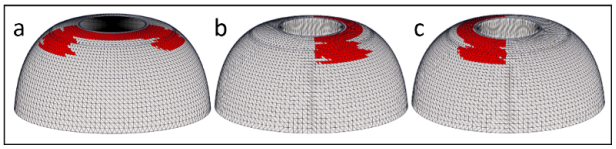

Fig. 2. (a) Representation of the surface using triangles. The red region is the target region to clean $\mathcal{P}$. (b, c) Two sample candidate setups from $\mathcal{S}$

formulate our cleaning problem as a multi-setup, multi-pass, cleaning task with setup planning for the target surface and trajectory planning for the cleaning tool.

We define a set of candidate setups $\mathcal{S} = \{s_j; j = 1, 2, \ldots, m\}$, where $s_j = \{p_i^j : i = 1, 2, \ldots, k\} \subseteq \mathcal{P}$, $k \leq n$ and the conditions on robot motion is satisfied $\forall p_i^j \in s_j$. Figure 2(b,c) demonstrates two candidate setups to clean $\mathcal{P}$. Each setup s_j corresponds to a distinct pose $\Gamma(\ell_j)$. The maximum number of passes to cover s_j is given by $N_m = \max_{i=1}^{|s_j|} N_i$, where N_i is the number of passes to clean p_i. Let t_m be the time to complete N_m passes. If $t_i \leq t_m$ is the time required for the i^{th} cleaning pass, then the total time spent to clean p_k is t^{p_k}=$\sum_{i=1}^{N_k} a_k \times t_i/(\tilde{a}_i)$, where $\tilde{a}_i$ is the total area covered by i^{th} pass. Cleaning rate for p_k is r_k^j=a_k/t^{p_k}, where a_k is the surface area of p_k. Cleaning rate for setup s_j is $R_j = (\sum_{i=1}^{|s_j|} a_i)/(t_m + t^s)$, where t^s is the setup time defined as time to change the pose of $\mathcal{P}$ from one setup to another.

We can generate different ordered setup sequences by permuting $s_i \in \mathcal{S}$. We define a valid setup plan $\mathbb{S} = \{s^i \in \mathcal{S} : i = 1, 2, \ldots, q \leq |\mathcal{S}|\}$ as an ordered sequence of setups that cleans the entire region. The path planner generates a trajectory $\tau(s^i)$ for each $s^i \in \mathbb{S}$. The robot may need to reposition the tool to cover disjoint patches in a setup and to transition between two setups. The sampling based method for generating $\mathcal{S}$ and the algorithm to find setup sequence solutions are described in Section IV and V, respectively. Our method to generate trajectories is described in Section VI. The method to select optimal operation parameters (e.g. tool speed, applied force, etc.) is described in Section VIII.

For each setup, s^i, there is an execution time $t^e(s^i) = t^c(\tau(s^i)) + t^r(\tau(s^i))$, where $t^c(\tau(s^i))$ is the cleaning time when the tool is in contact with $\mathcal{P}$ while following $\tau(s^i)$ and $t^r(\tau(s^i))$ is the repositioning time when the cleaning robot moves between setups or disjoint patches). We define the total cleaning time for a valid setup plan $\mathbb{S}$ as below:

$$\mathbb{T}(\mathbb{S}) = q \times t^s + \sum_{i=1}^{q} t^e(s^i) \tag{1}$$

The problem is formally stated as follows: *Find a setup plan $\mathbb{S}^* = \{s_1^*, s_2^*, \ldots, s_l^*\}$, where $s_1^*, s_2^*, \ldots, s_l^* \in \mathcal{S}$, such that $\mathcal{P}$ is completely clean and $T(\mathbb{S}^*)$ is minimized.*

IV. Generating Candidate Setups

We use a sampling based approach to generate the initial collection of setups. For a candidate setup s_i being evaluated, we examine how many patches in $\mathcal{P}$ are reachable by the robot in the desired orientation. We solve the inverse kinematics for the vertices for each $p_i \in \mathcal{P}$. If all vertices of a patch can be reached, then we consider that patch to be reachable by the robot and assign that patch to s_i. The patch must be relatively small compared to the tool tip's surface area. This is achieved by surface triangulation.

We begin with a coarse resolution sampling over a large range of configuration space parameters to establish the narrower feasible sampling range. We eliminate candidate setups that are not useful for cleaning any patch. We will use the notion of non-dominated setup to describe these approaches. We consider a setup s to be dominated if there exists another setup s' that contains all the patches covered by s. At the beginning of each of the following sampling approach we initialize an empty set of setups, ψ. When the non-dominated setups in ψ cover all $p_i \in \mathcal{P}$, we send them as input to setup planner.

A. Fine resolution uniform sampling

1) Set a fine resolution for each axis of the configuration space to perform uniform sampling over the narrower sampling range.
2) Generate setup samples using the resolution from Step (1) and add them to ψ.
3) Eliminate all dominated setups in ψ.

B. Hierarchical uniform sampling with gradient descent

1) Set a coarse resolution for each axis of the configuration space to perform uniform sampling over the narrower sampling range.
2) Generate samples of setups using above resolution.
3) Pick setups in the generated set which do not belong to ψ and refine them by using gradient descent over the configuration parameters to optimize the area covered by each setup. Add the refined setups to ψ.
4) Eliminate all dominated setups in ψ.
5) If the non-dominated setups in ψ do not cover all $p_i \in \mathcal{P}$, then refine sampling resolution and go to step (2).

C. Random sampling

1) Generate a random setup sample from the narrower sampling range.
2) Refine this setup by using gradient descent to optimize the area covered by this setup.
3) If refined setup does not belong to ψ, then add it to ψ.
4) Repeat steps (1-3) until setups in ψ cover all $p_i \in \mathcal{P}$.
5) Eliminate all dominated configurations in ψ.

D. Gradient descent to improve area coverage

We use gradient descent in the configuration space to optimize the area covered by a setup. Let $P_1 \in \mathcal{P}$ be the set of patches that were reachable by the robot for a starting configuration. We define $|P_1|$ as the score for the starting configuration. Suppose one step was taken in the configuration space by gradient descent and it is at a new setup configuration. Let $P_2 \in \mathcal{P}$ be the set of patches that are reachable by the robot for this new configuration. We use the following two score evaluation schemes:

TABLE I

GRADIENT DESCENT VARIANTS EXPLORED TO IMPROVE AREA COVERED BY A SETUP. SCORING METHODS: TYPE I - CONSERVATIVE; TYPE II - TWO ROUNDS (ROUND 1 IS CONSERVATIVE AND ROUND 2 IS ABSOLUTE)

Abbreviation	Scoring method	Gradient descent type
C_XY	Type I	Along x and y axes for fixed α
CA_XY	Type II	
C_XY_α	Type I	Along x and y axes keeping α fixed and then along α axis alone
CA_XY_α	Type II	
C_XYα	Type I	Along x, y, and α axes
CA_XYα	Type II	

Conservative scoring: In this scheme, We enforce a constraint to ensure that the improved configuration is able to cover all the patches that were present in the starting configuration. Therefore, if $P_1 \subseteq P_2$, then the score of the new configuration is $|P_2|$. Else, it is $|P_1 \cap P_2|$.

Absolute scoring: We do not enforce any constraint. The score for the new configuration is evaluated as $|P_2|$.

We explore different gradient descent variants by performing a search over the entire configuration space, over the subspaces by batch covering the entire configuration space, with conservative scoring scheme, and with both conservative and absolute scoring schemes (refer to Table I).

V. SETUP PLANNING

The sampling based method described in section IV leads to a set of refined setups. The setup planner finds a setup plan from this set of candidate setups $\mathcal{S}$.

Our algorithm uses a *depth-first branch-and-bound* search with computational time bound $\mathbb{T}_{max}$ to generate setup plans. Let $s_j^i \in \mathcal{S}$ represent a setup, where i is the iteration index and j is the node index in the i^{th} solution. Therefore, the i^{th} solution is given by $\mathbb{S}^i = \{s_1^i, s_2^i, \ldots, s_r^i : r \leq |\mathcal{S}|\}$. Initially, we select a setup $s_1^0 \in \mathcal{S}$ based on the cleaning rate and area coverage of that setup. This results in the remaining stain area $\mathcal{P}_r = \mathcal{P} - \{p_i \in s_1^0\}$ to be covered. Then we use the same basis to find other setups s_j^i, one at a time, until all the stain area is covered ($\mathcal{P}_r = \emptyset$). Once the initial solution $\mathbb{S}^0$ is found, we set the best solution $\mathbb{S}^* = \mathbb{S}^0$ and the cleaning time for the best solution $\mathbb{T}^* = \mathbb{T}(\mathbb{S}^0)$, which is the sum of all the setup time and execution time for the solution setup sequence. Then we keep branching to find an optimal solution until $\mathbb{T}_{max}$ is exceeded.

Let, $\mathbb{S}_{curr}$ be a set of setups in a solution search path. $\mathbb{S}_{curr}$ may not necessarily be a complete solution. We define $\mathbb{T}(\mathbb{S}_{curr})$ as the total execution and setup time for the setups included in $\mathbb{S}_{curr}$. We consider this as the current cost. We define $\mathbb{T}^{lb}(\mathcal{P}_r)$=$t^s + \min_j t^e(s_j^i)$, where $s_j^i \notin \mathbb{S}_{curr}$, as the lower bound for execution and setup time for the remaining stain region $\mathcal{P}_r$ and use it as the lower bound on future cost to prune sub-optimal branches in the search tree.

Input for Algorithm 1: $\mathcal{P}$: set of patches on the surface that need to be cleaned. $\mathcal{S}$: set of candidate setups.

Input for Algorithm 2: $\mathcal{P}_r$: set of remaining patches not covered in any setup so far in this solution search path. $\mathcal{S}_u$: set of candidate setups not used so far in this solution search path. $\mathbb{S}_{curr}$: current setup plan in this solution search path.

Algorithm 1 FindSetupPlan $(\mathcal{P}, \mathcal{S})$

```
Initialize 𝕊* = ∅
Initialize 𝕋* = ∞
Call AddSetUp (𝒫, 𝒮, ∅)
Return 𝕊*
```

Algorithm 2 AddSetUp$(\mathcal{P}_r, \mathcal{S}_u, \mathbb{S}_{curr})$

```
If computation time exceeds 𝕋_max then abort search.
If 𝒫_r = ∅ and 𝕋(𝕊_curr) > 𝕋* then Return
If 𝒫_r = ∅ and 𝕋(𝕊_curr) ≤ 𝕋*, then
    update 𝕊* = 𝕊_curr, 𝕋* = 𝕋(𝕊_curr) and Return
If 𝕋(𝕊_curr) + 𝕋^lb(𝒫_r) ≥ 𝕋(𝕊*), then Return
Otherwise,
    If ∃p ∈ 𝒫_r associated with only one s ∈ 𝒮_u, then
    Find 𝒫(s) by patches that are present in s
    Call AddSetUp (𝒫_r − 𝒫(s), 𝒮_u − s, 𝕊_curr ⋃ s)
    Otherwise,
      Sort 𝒮_u by highest to lowest cleaning rate.
      For every s in 𝒮_u in decreasing order of
      cleaning rate
        Find 𝒫(s) by patches that are present in s
        Call AddSetUp (𝒫_r − 𝒫(s), 𝒮_u − s, 𝒮_curr ⋃ s)
```

VI. TRAJECTORY PLANNING

Cleaning trajectory is planned for each pass of each setup. A setup $s_j^* \in \mathbb{S}^*$ may require N_m^j passes to complete cleaning. Correspondingly, there are total N_m^j number of trajectories generated for the setup s_j^*. For example, if s_j=$\{p_1^j, p_2^j, p_3^j\}$ and number of passes required for p_1^j, p_2^j, and p_3^j are 1, 2, and 3, respectively, then the first trajectory will pass through all the patches p_1^j, p_2^j and p_3^j. The second trajectory will pass through p_2^j and p_3^j, and the third trajectory will cover p_3^j alone. To generate the trajectory, we intersect the target surface by equally spaced parallel planes and sample the resulting curves to generate waypoints. We generate each trajectory $\tau(s_j)$ to cover s_j as a spline curve $\ell(s_j, \rho) \in \mathbb{R}^6$, where $\rho \in \mathbb{I}$ is an arc-length parameter over a set $\mathbb{I}$. Figure 3(a) demonstrates an example of a cleaning trajectory for one sample setup. The robot may not be able to travel through all the desired points on a continuous smooth spline curve. In these cases we segment the trajectory in multiple spline curves and reposition the arm after completing each spline such that it can travel through the next spline. We also reposition the arm to avoid traveling through already cleaned regions, which may come in between unclean regions. Repositioning is similar to jumps because the arm comes out of contact with the $\mathcal{P}$ during these moves, hence incurring repositioning time.

The cleaning trajectory is generated by creating a spline through the connected triangles in a setup. Therefore, the

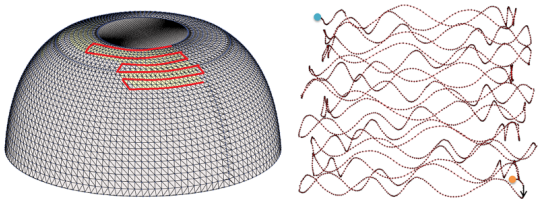

Fig. 3. (a) An example trajectory generated by parallel plane slicing for a setup. (b) Overlay pattern generated by an example set of parameters. The normalized values for the parameters used in this example is v=0.10, f_x=0.50, F_x=1.0, S_x=1.0, F_z=0.50

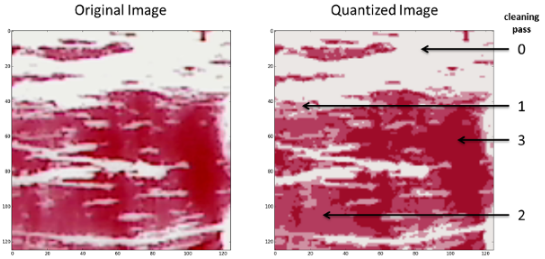

Fig. 4. Pixel-wise vector quantization of stain image

cleaning trajectory consists of multiple splines for a single setup. We overlay a force oscillation on top of the sweeps to expedite cleaning. This is described in Section VIII. Figure 3(b) illustrates an example of a trajectory with overlaid force oscillation. The repositioning trajectory is generated as a spline curve that connects the current pose of the tool, an intermediate pose where the tool is not in contact with $\mathcal{P}$, and the starting pose of the tool for the next cleaning spline. We use the spline motion primitives of the Kuka Robotics Application Programming Interface (API), which takes a set of way points $\mathcal{L} = \{\ell_i \in \mathbb{R}^6\}$ as input and creates a smooth spline curve for the tool to follow if the points are in the valid reachable space. We denote the trajectory for the ith pass on the setup s_j by $\tau_i(s_j) = \{\tau_i^{cl}(s_j), \tau_i^{repo}(s_j)\}$, where $\tau_i^{cl}(s_j)$ is the cleaning trajectory and $\tau_i^{repo}(s_j^*)$ is the repositioning trajectory for the ith pass.

VII. Perception

We detect the object using the Iterative Closest Point (ICP) algorithm by matching the depth image of the object with its CAD (Computer-aided Design) model. We use image processing to detect the stain on the object, evaluate the cleaning performance after each cleaning pass, and estimate the required number of cleaning passes. After each pass we must know how much dirt was removed to measure cleaning rate. We use K-means clustering to classify the background color and stain. This method performs pixel-wise vector quantization. The pixels are represented in 3D space of Red-Green-Blue (RGB) colors and all pixels are clustered into two colors− white to represent the cleaned area and black to represent the remaining stains. Let C_0 and C_1 be the number of black pixels in the binary image of the target surface before and after the cleaning pass, respectively. Let I_h and I_w be the height and width of the target region in pixels. Then we define cleaning performance $C_p = (C_0 - C_1)/(I_h \times I_w)$.

Our algorithm needs to know the number of cleaning passes required to clean each triangulated stain patch. We quantize the pixel colors of the image of the stain into N_c number of color clusters using k-means algorithm. N_c can be determined through supervised learning methods with a rich database of images of stains. In our current implementation N_c is user defined. The color value of the clusters are considered as points in RGB 3D space and their distance from the origin determine the required number of cleaning passes. Fig. 4 illustrates an example for N_c=4. The RGB color value furthest from the origin, i.e., from black, represents no cleaning pass and the point nearest to the origin represents three cleaning passes.

VIII. Operation Parameter Selection

We tune the following robot motion parameters to find optimal cleaning performance: (1) Robot tool speed v, (2) Force f applied by the tool in orthogonal direction to the surface, (3) Stiffnesses S_x and S_y in the x and y directions of the tool reference frame, (4) Frequency f_x (or f_y) of the forced oscillation in the x (or y) direction with respect to the tool reference frame, and (5) Amplitude of overlaid force F_x and F_y for Lissajous force oscillation mode. It is difficult to express cleaning performance as a closed form function of these parameters. We adapted the Gaussian Process Regression based learning method developed in our previous work [10] to estimate the optimal cleaning parameters. The method requires to conduct some initial exploration experiments and then it selects candidate points in the parameter space to conduct exploitation experiments. The points are selected based on a probability of achieving desired cleaning performance.

IX. Results

A. Synthetic test cases

We tested the setup planner by conducting simulated experiments using the Puma 560 robot for cleaning. Figure 5 shows four arbitrary curved surfaces used as synthetic objects to clean in the simulations. The curved surfaces were represented as triangulated mesh where each face has an area less than 1.5 cm^2. There are about 4000 to 5000 stain patches (faces marked as red) on these objects. Stain intensity was set such that each patch could be cleaned by one to three cleaning passes. In our experiments we have used a robot to immobilize the object and another robot to manipulate the cleaning tool. The two arms can also be used to change the pose of the object. We have considered a fixture-free work environment on a plane. Therefore we had a 3D configuration space in our experiments (x, y, and α, where α is the rotation about z). The narrowed down sampling region was [0, 1m] $\times$ [0, 1m] $\times [0, 2\pi]$.

1) Baseline results: To find the number of minimum possible setups needed to clean the four test objects, we uniformly sampled the configuration space with fine resolution, eliminated the dominated setups, and ran setup planner on the

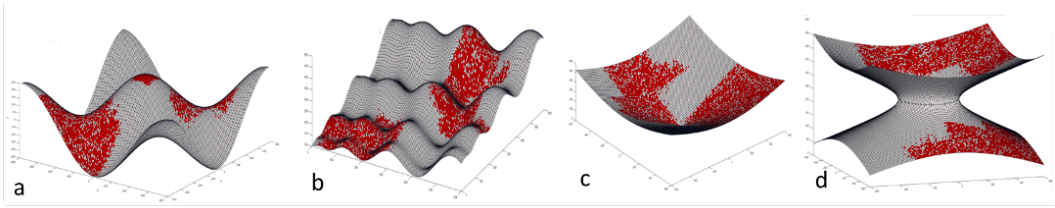

Fig. 5. Four curved surfaces used as synthetic test objects: (a) Sine function (1600mm x 1600 mm x 500 mm), (b) Schwefel function (600mm x 600 mm x 350 mm), (C) Concave bowl (1200mm x 1200 mm x 530 mm), and (d) Hyperboloid of one sheet (1200mm x 1200 mm x 412 mm).

TABLE II

BASELINE RESULTS FOR THE FOUR TEST CASES IN SIMULATION

Case	No. of setups in solution	Non Dominated Setups
Sine	4	36
Schwefel	1	1
Concave bowl	5	531
Hyperboloid	4	751

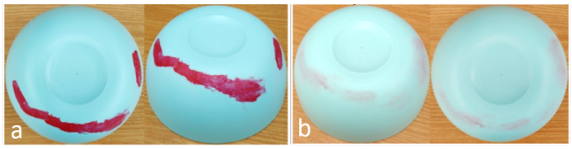

Fig. 6. The object's surface (a) before cleaning and (b) after cleaning. Images taken from two different angles.

non-dominated setups. Gradient descent was not used here (Section IV-A). The largest resolution for uniform sampling that gave set cover for the stain patches for all the four test cases was 50 mm in both x and y axes and 10° in α axis. The baseline result is summarized in table II. The reachability test function determines how many stain patches are reachable by the robot for a particular setup configuration. We use the *total reachability function calls* n_{rfc}, the number of calls to the reachability function, as one of the performance measures as this function takes the most time to evaluate. For all the cases in Table II, n_{rfc} = 16317.

2) Comparison between different sampling and gradient descent approaches: We explored hierarchical uniform sampling and random sampling approaches as mentioned in Sections IV-B and IV-C. We tested the performance of gradient descent variants described in Section IV-D on these approaches. For hierarchical uniform sampling, we used different resolutions at different levels of hierarchy. The resolution was doubled along one axis at a time in the hierarchies. The results for the four test cases using hierarchical uniform sampling are given in Table III. The system was able to find the optimal setup solution for the four test cases by refining the input setup configurations for the setup planner with the CA_XY variant (refer to Table I). For the random sampling approach, we ran 100 trials for each test case, with each type of gradient descent. The likelihood of finding the optimal solution for each case is summarized in Table IV. We can see that the likelihood is reasonable with the CA_XY variant for all the four test cases. We can see that hierarchical uniform sampling with the CA_XY variant can guarantee an optimal solution. The random sampling based approach has three fold reduction in n_{rfc} compared to that of hierarchical uniform sampling. But with a certain type of gradient descent, it can only guarantee optimal solution with likelihood less than 0.5.

3) Comparison of heuristics for setup planner: We used two heuristics on the branch-and-bound-depth-first search for faster convergence towards optimal solution. The first heuristic (step 5 of algorithm 2), sorting candidate setups based on cleaning rate, provides guidelines for choosing a potential branch with optimal solution. The second heuristic (step 4 in algorithm 2), is to prune branches based on the estimated lower bound on future cost. We tested the planner performance by toggling these two heuristics on and off. Performance is summarized in Table V for Sine-object with 36 uniform initial samples, among which 12 are non-dominated setups, and the optimal solution comprised five setups.

B. Physical Testing Results

We conducted robotic cleaning experiments on a bowl and used acrylic paint as a surrogate for stain as shown in Fig. 6. Stain intensity was controlled by applying multiple layers of paint. The map among the coordinate frames of the robots and visual sensors can be found by tracking markers attached to the end effectors of the robots. The location of the object with respect to the visual sensor can be found using the ICP algorithm.

We segmented the bowl as a discrete set of 15644 triangles, making a total surface area of about 1472.8 cm^2. The stain region consisted of about 1000 triangles with surface area of 113.45 cm^2. The surface area of each triangle was less than 10 mm^2. We found the following parameters using our parameter selection method (described in Section VIII) for optimal cleaning of the acrylic paint stain on the bowl surface and used them in the experiment: (1) f_x=7Hz (2) S_x, S_y=5000 N/m, (3) F_x, F_y = 30 N, (4) v = 0.3 (where maximum joint velocity v_{max} =1), and (5) F_z=10 N. We used Y_{th}=0.60, P_{th}=0.10 and N=10. We sampled the candidate setups in 3 Degrees of Freedom(DOF) x, y and α, where α is the angle about z. Hierarchical uniform sampling with gradient descent was used to generate the candidate setups. We found a solution sequence of five setups for the cleaning task generated by our setup planner. It took three passes to clean the region in setup 1 and two passes for others. The

TABLE III

RESULTS OF HIERARCHICAL UNIFORM SAMPLING FOR FOUR SYNTHETIC OBJECTS

Gradient Descent Type	Sine		Schwefel		Concave Bowl		Hyperboloid	
	Setups in solution	n_{rfc}	Setups in solution	n_{rfc}	Setups in solution	n_{rfc}	Setups in solution	n_{rfc}
w/o grad. dsc.	5	36	1	36	6	36	5	36
CA_XY	4	536	1	264	5	574	4	457
CA_XY_α	5	1074	1	811	5	1179	4	1089
CA_XYα	5	659	1	426	7	1209	5	671

TABLE IV

LIKELIHOOD OF FINDING OPTIMAL SOLUTION BY RANDOM SAMPLING

Grad. Dsc. Type	**Sine**	**Schwefel**	**Concave bowl**	**Hyperboloid**
Without Grad. Dsc.	0	0.13	0.01	0.05
CA_XY	0.19	0.35	0.21	0.32
CA_XY_T	0.07	0.54	0.04	0.01
CA_XYT	0.05	0.43	0.05	0.08

TABLE V

COMPARISON OF HEURISTICS FOR SETUP PLANNER

Heuristic		**Node expansion in search tree before convergence**		
Branch Guiding	**Branch Pruning**	**Sine**	**Concave bowl**	**Hyperboloid**
ON	**ON**	20286	353630	569
OFF	**ON**	39822	362208	679
ON	**OFF**	1176109	16240590	3104
OFF	**OFF**	1188175	16242862	3138

overall cleaning rate was about 0.167 cm^2/s. Figure 6 shows the object before and after the cleaning task is performed.

X. CONCLUSIONS

This paper presents algorithms for automated cleaning of objects with curved surfaces. The method is equally effective for paint stripping and polishing tasks. The algorithm is capable of generating the optimal number of setups to maximize the cleaning rate. The system identifies the best cleaning parameter settings by a semi-supervised learning scheme. It is capable of adjusting the parameters based on the observed performance. The system was able to successfully remove stains that were difficult and tedious for humans to remove. In the current implementation, setup change is performed manually. In the future, we will automate setup changing using two manipulators.

REFERENCES

[1] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. K. Gupta. An industrial robotic knowledge representation for kit building applications. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1365–1370, Oct 2012.

[2] A. G. Banerjee, A. Barnes, K. N. Kaipa, J. Liu, S. Shriyam, N. Shah, and S. K. Gupta. An ontology to enable optimized task partitioning in human-robot collaboration for warehouse kitting operations. In *Proc. SPIE, Next-Generation Robotics II; and Machine Intelligence and Bio-inspired Computation: Theory and Applications IX, 94940H*, 2015.

[3] K. N. Kaipa, S. S. Thevendria-Karthic, S. Shriyam, A. M. Kabir, J. D. Langsfeld, and S. K. Gupta. Resolving automated perception system failures in bin-picking tasks using assistance from remote human operators. In *Proc. of IEEE International Conference on Automation Science and Engineering*, August 2015.

[4] K. N. Kaipa, S. Shriyam, N. B. Kumbla, and S .K. Gupta. Automated plan generation for robotic singulation from mixed bins. In *IROS Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.

[5] K. N. Kaipa, N. B. Kumbla, and S .K. Gupta. Characterizing performance of sensorless fine positioning moves in the presence of grasping position uncertainty. In *IROS Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.

[6] K. N. Kaipa, A. S. Kankanhalli-Nagendra, and S. K. Gupta. Toward estimating task execution confidence for robotic bin-picking applications. In *AAAI Fall Symposium: Self-Confidence in Autonomous Systems*, 2015.

[7] C. Morato, K. N. Kaipa, and S. K. Gupta. Improving assembly precedence constraint generation by utilizing motion planning and part interaction clusters. *Computer-Aided Design*, 45(11):1349 – 1364, 2013.

[8] J. D. Langsfeld, A. M. Kabir, K. N. Kaipa, and S. K. Gupta. Online learning of part deformation models in robotic cleaning of compliant objects. In *ASMEs 11th Manufacturing Science and Engineering Conference*, June 2016.

[9] J. D. Langsfeld, A. M. Kabir, Kaipa K. N., and Gupta S. K. Robotic bimanual cleaning of deformable objects with online learning of part and tool models. In *Proc. of IEEE International Conference on Automation Science and Engineering*, August 2016.

[10] A. M. Kabir, J. D. Langsfeld, C. Zhuang, Kaipa K. N., and Gupta S. K. Automated learning of operation parameters for robotic cleaning by mechanical scrubbing. In *ASMEs 11th Manufacturing Science and Engineering Conference*, June 2016.

[11] J. Hess, G. D. Tipaldi, and W. Burgard. Null space optimization for effective coverage of 3D surfaces using redundant manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1923–1928, 2012.

[12] R. Bormann, F. Weisshardt, G. Arbeiter, and J. Fischer. Autonomous dirt detection for cleaning in office environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1260–1267, May 2013.

[13] F. Sato, T. Nishii, J. Takahashi, Y. Yoshida, M. Mitsuhashi, and D. Nenchev. Experimental evaluation of a trajectory/force tracking controller for a humanoid robot cleaning a vertical surface. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3179–3184, Sept 2011.

[14] C. H. King, T. L. Chen, A. Jain, and C. C. Kemp. Towards an assistive robot that autonomously performs bed baths for patient hygiene. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (1):319–324, 2010.

[15] F. Nagata, T. Hase, Z. Haga, M. Omoto, and K. Watanabe. CAD/CAM-based position/force controller for a mold polishing robot. *Mechatronics*, 17(4-5):207–216, 2007.

[16] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *IEEE International Conference on Robotics and Automation*, pages 3968–3974, May 2009.

[17] A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann. On-line periodic movement and force-profile learning for adaptation to new surfaces. In *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 560–565, Dec 2010.

[18] D. Martínez, G. Alenyà, and C. Torras. Planning robot manipulation to clean planar surfaces. *Engineering Applications of Artificial Intelligence*, 39:23–32, 2015.

[19] G. Hermann. Algorithms for real-time tool path generation. *Geometric Modeling for CAD Applications*, pages 295–305, 1988.