International Journal of Advanced Robotic Systems
January–February 2026: 1–12

DOI: 10.1177/17298806261415956
journals.sagepub.com/home/arx
Sage

Adaptive vision-based cleaning of industrial workstations using a collaborative mobile manipulator arm

Guillaume Dupoiron[1], François Michaud[2,3] and Jean-Philippe Roberge[1]

Abstract

In small- and medium-sized enterprises, machine tools generate significant quantities of various types of dust, including plastic, iron, and other materials. To facilitate efficient cleaning or recycling of these materials, workers are required to manually vacuum the machines, directly targeting the accumulated dust. This paper introduces a novel approach for automated cleaning in industrial settings, diverging from the common practice of systematic surface coverage. Our method, mobile arm vision-based adaptive cleaning, employs a two-stage adaptive strategy that necessitates only a coarse representation of the machine. First, an offline process optimizes the positions of the robot's mobile base and arm configurations to maximize the visibility of machine sections from the camera's perspective at the end effector. Then, an online process detects dust accumulations and adjusts its trajectory based on their locations. We demonstrate the capability of our approach by cleaning three machines with intricate geometries. Our results indicate that our method can clean approximately 84% of the dust heaps present on the machines.

Keywords

Automated cleaning, industrial robotics, mobile manipulation, adaptive trajectory planning

Received: 30 June 2025; accepted: 15 December 2025

Topic: Robot Manipulation and Control
Topic Editor: Carbone, Giuseppe
Associate Editor: Rajput, Ankita

Introduction

Cleaning workstations is a common yet complex task in industrial settings, typically requiring customization for each space. Factories often accumulate dust from fabrication equipment, such as cutting and grinding machines, which can slow production and adversely affect the environment. The continuous presence of dust, which often cannot be cleaned continuously, poses a potential health risk to workers and represents a form of waste that could potentially be recycled or reused. Under these conditions, cleaning must be both efficient and thorough, as dust accumulation can hinder workers' health and production goals. To overcome these problems, the factories can install static vacuum cleaners as needed along the production line, which typically presents a costly and cumbersome solution. When connected to a central vacuum system, this setup also limits reconfigurability. Automated cleaning methods have also been proposed, typically relying on systematic surface coverage.[1,2] Although these methods can ultimately achieve the cleaning objective, they are often highly inefficient in practice, as they do not distinguish between clean and dusty areas. Other works have explored artificial intelligence (AI)- or reinforcement learning (RL)-based techniques,[3–5]

[1]Command and Robotics Laboratory, École de technologie supérieure, Montreal, Quebec, Canada
[2]Interdisciplinary Institute for Technological Innovation (3IT), Université de Sherbrooke, Sherbrooke, Quebec, Canada
[3]Department of Electrical Engineering and Computer Engineering, Université de Sherbrooke, Sherbrooke, Quebec, Canada

Corresponding author:
Guillaume Dupoiron, Command and Robotics Laboratory, École de technologie supérieur, Montréal, H3C 1k3, Canada.
Email: guillaume.dupoiron.1@ens.etsmtl.ca

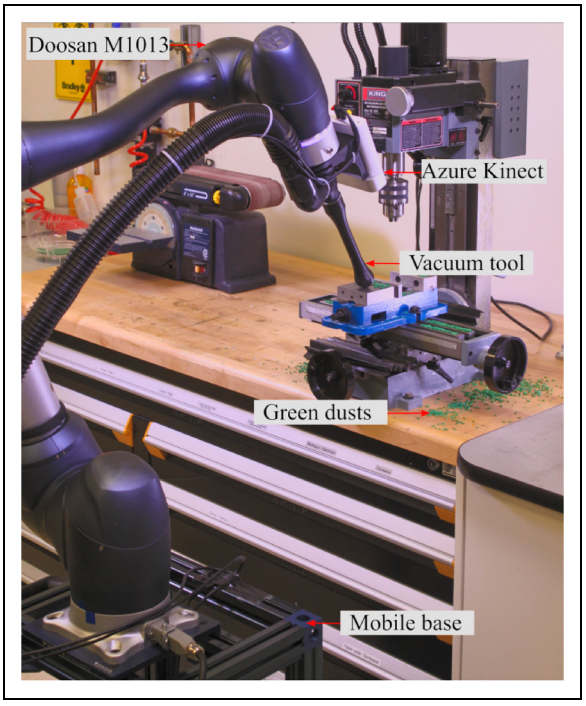

Figure 1. The system cleaning a machine using mobile arm vision-based adaptive cleaning (MAVAC).

but these approaches demand large datasets, long training times, and have been tested mainly on simple geometries such as tables or floors, making them unsuitable for industrial machines with complex structures.

To address these challenges, this article presents a robotic cleaning platform that includes a serial manipulator equipped with an RGB-D camera, mounted on an automated guided vehicle mobile base (Figure 1). This robot is specifically designed to facilitate the cleaning of industrial machines with diverse geometries.

This paper introduces mobile arm vision-based adaptive cleaning (MAVAC), a novel two-stage framework that bridges the gap between systematic coverage and AI-based methods. Unlike traditional automated cleaning approaches[1,2] that rely on monotonous, progressive surface scanning within predefined cleaning areas, our method is vision-based and adaptive: dust is detected using an onboard camera, and targeted motions are planned to vacuum the identified dust accumulations. We assume that at least a coarse representation of these machines is available. Should such a representation be lacking, our approach can use a basic environmental model obtained. As an example, a machine with an approximate volume of 1.3 m^3 and an exposed surface area of about 4.97 m^2 can be reconstructed using photogrammetry or algorithms such as RTAB-Map with a standard smartphone in about 10 min.[6] Initially, an offline process optimizes the positions of the robot's mobile base and arm configurations to maximize the visibility of machine sections from the camera's perspective at the end effector. Then, the online process uses the camera to efficiently scan the machine from strategic viewpoints. Starting with the end effector poses that offer the best coverage, vision algorithms are employed to detect dust and formulate cleaning trajectories. This adaptive, vision-based approach ensures that dust is cleaned from all detected locations, creating a comprehensive map of dust distribution on the machine. Impedance control is implemented in the robot to facilitate efficient cleaning by enabling direct interaction between the suction hose and the machine, much like a human cleaner would. Moreover, this approach provides the necessary compliance for the robot to make contact with and adapt to the machine's geometry, offering increased flexibility in handling contact with the machine's surfaces.

Overall, the contributions presented in this paper are threefold:

- We introduce an adaptive strategy in which online dust detection exploits offline path planning for inspection to adjust trajectories and target only dirty surfaces.
- We introduce a refined definition of accessible surfaces, combining reachability and visibility, to more realistically evaluate what parts of the machine can be cleaned.
- We validate MAVAC on real machines with concavities and irregular geometries, while impedance control ensures safe interaction with machine surfaces.

This paper is organized as follows: the "Related work" section examines existing methods and research relevant to our study, and the "Cleaning task formulation" section defines the problem of robotic cleaning. The "Mobile arm vision-based adaptive cleaning (MAVAC)" section presents our adaptive vision-based cleaning approach. The "Experimental setup" section details our experimental setup, including the description of the robotic platform, testbeds, and data collection. The "Results" section details our experimental findings. The paper concludes by summarizing our findings and their implications, along with discussing limitations and suggesting avenues for future work.

Related work

Kim et al.[7] provide an insightful review of methods utilized for cleaning with mobile manipulator robots. These challenges are commonly categorized into three subproblems: path planning, dust detection, and trajectory generation for the cleaning task.

Mobile base placement for coverage path planning

Methods determining optimal base placement for effective manipulation or coverage often employ reachability maps

to evaluate the maneuverability of the robot arm. Prior studies[8–10] have demonstrated the value of these maps, which identify positions where the robotic arm can operate in the best possible conditions. For instance, Vahrenkamp et al.[9] study the inverse reachability map to position the mobile base for object grasping. By dividing the workspace, Porges et al.[10] suggest using a capability map, which is essentially a reachability map with a quality index for every position, to provide a more accurate assessment. Makhal et al.[8] expand on this work to present a real-time robot base placement method. These studies primarily focus on mobile base placement for object grasping at a specific position. In the context of coverage path planning, the objective is not to reach specific positions but to cover a surface efficiently using a trajectory that minimizes mobile base movements. To address this issue, Paus et al.[1] present a greedy method to select multiple local positions to cover a machine entirely. By iteratively exploring different positions around the machine and estimating the accessibility of poses for the robotic arm, this approach aims to achieve the most complete coverage possible. However, this method has only been tested in simulation and never in real environments.

Other techniques, such as stochastic and genetic algorithms,[11] as well as hybrid solutions,[12] have been explored to improve machine coverage and cleaning process efficiency. However, these studies generate trajectories to cover the machine according to the action range of the tool center point (TCP).[1,2,13,14] Systematically cleaning the entire machine using only the action range of the TCP is not efficient enough, as only dirty machine parts should be vacuumed.

Industrial dust detection

Two main approaches are commonly used for dust detection. The first relies on classical image segmentation to identify visually distinctive types of dust.[14,15] These techniques typically use color-based segmentation and background subtraction. Since highly precise three-dimensional (3D) cameras cannot easily be mounted on the robot, and because dust particles are too small for reliable depth sensing, only two-dimensional images can be used for detection.

The second approach leverages convolutional neural networks (CNNs) to identify dirty or contaminated surfaces. Ramalingam et al.[16] employ a YOLOv3-based CNN to detect door handles and autonomously disinfect them. Yin et al.[5] train a DarkFlow-based CNN to detect food debris on tables and prompt wiping actions, distinguishing between liquid and solid residues. More recently, Chen et al.[17] introduce the use of S-YOLOv5s for dust detection in real-world environments.

The literature emphasizes the significance of CNN-based detection for effective cleaning, as it accommodates adaptation to different environments and lighting conditions.

Trajectory generation for cleaning

Cleaning methods using either RL or CNN have been proposed to generate robot movements. Lew et al.[4] use an RL method to wipe dust on a table. The robot is trained according to a dust model and controlled in admittance. More recently, Gaurav et al.[3] combine a time contrastive network and RL to mimic the human mopping behavior from a video. Using demonstration learning, Jaeseok et al.[18] and Cauli et al.[19] present a method to clean a table. Further methods, as Saito *et al.*,[20] extract features from video with a CNN and use them in a deep neural network with the robot force and joints information. However, these machine learning methods require a large set of data or extensive training time, and are only tested on tables or flat floors. Cleaning complex machines is most likely to require significantly more data and time.

Various methods employing classical control have also been investigated to generate cleaning trajectories using traveling salesman problem (TSP) algorithms. Wakayama et al.[14] leverage RoboDK for trajectory generations in cleaning applications. More recently, Qi et al.[2] and Vatavuk et al.[21] segment specific areas such as furniture surfaces or areas occupied by grapevines, and then apply an algorithm to cover these surfaces. Similarly, Zhang et al.[22] segment the surface to be cleaned into small circles and generate a path using a TSP algorithm. These approaches have been primarily developed for specific domestic contexts where the entire surface needs to be cleaned.

In industrial settings, dust can be segmented, and the robot can directly target dust heaps,[5] thereby improving the cleaning process by reducing unnecessary cleaning time and enhancing efficiency.

For all the aforementioned tasks, such as wiping, scrubbing, or vacuuming, the robot has to be in contact with the surface during the execution of the cleaning trajectory. With a rigid tool, the robot may damage the surface or itself. To reduce this risk, Yuang et al.[23] and Wakayama et al.[14] use highly compliant tools adapted to their environments. Another solution could be to use an impedance control as Lew et al.[4]

Cleaning task formulation

As a general observation, systematic methods presented in the "Related work" section tend to be slow because they do not specifically target regions where dust is present. AI-based methods, such as CNN or RL, use a camera to detect dust and guide robot movements. However, these methods have, so far, only been applied to cleaning simple structures, such as tables. Most industrial machines, unlike these simple geometries, are not solely made of flat surfaces but also complex shapes, including concavities and hard-to-reach surfaces.

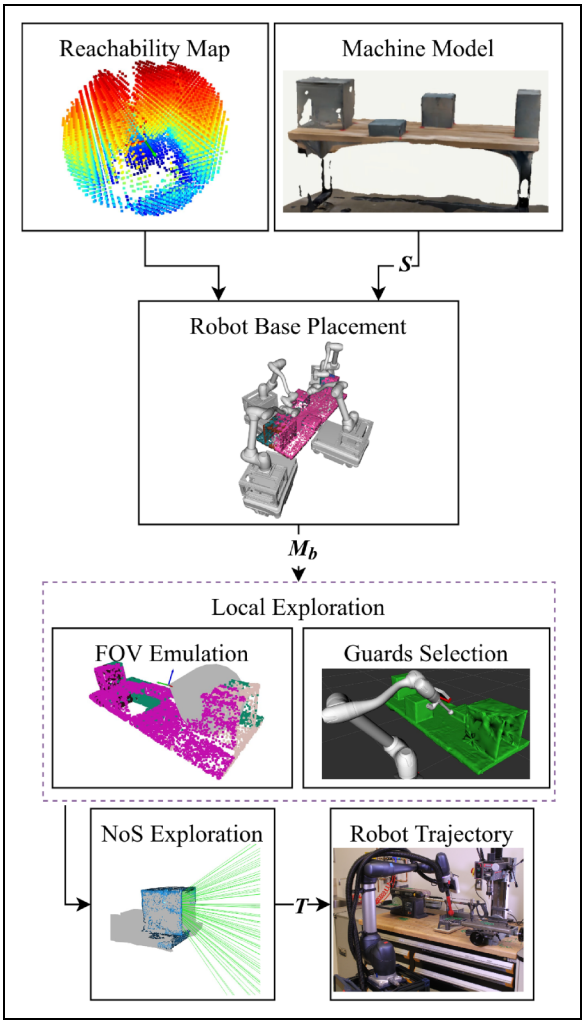

Figure 2. Overview of the offline algorithm: spots are selected from the machine model and the reachability map; for each spot, guard configurations are selected to inspect the machine; a special exploration is done for non-observed surface (NoS) to get the final trajectory.

To design and evaluate our dust cleaning method, some information must be available regarding the cleaning task. In addition, at least a basic model of the machine should be readily available. Moreover, human presence during navigation within the factory is not considered.

The cleaning process requirements are determined as follows:

- The machine should be cleaned as thoroughly as possible.
- The robot must adapt its trajectories to avoid damaging the machine in 12 trials and the surrounding environment.
- The robot should minimize cleaning time.
- The robot must be compliant when in contact with the machine.
- The approach should be generalizable to different machines and environments.

The machine's surface is discretized into a point cloud, denoted as a set $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\} \in \mathbb{R}^{3\times n}$. We consider the dust on the machine, denoted as $\mathbf{D}$, as a subset of $\mathbf{S}$: $\mathbf{D} \subset \mathbf{S}$. One objective is to accurately identify the positions of all the dust particles on the machine. To do so, the mobile base moves to several locations, called spots, identified as $\boldsymbol{M_b} = \{\boldsymbol{m_b}i \in \mathrm{SE}(2),\ 1 \le \mathrm{i} \le \mathrm{m_{max}}\}$, where $m_{\max}$ represents the maximal number of spots. For each spot, the robot arm reaches inspection configurations, denoted as $\mathbf{G} = \{\mathbf{q}_1, \ldots, \mathbf{q}_j\} \in \mathbb{R}^{6\times j}$ and called guard configuration.

During dust detection, one of the main objectives is to minimize the amount of required movements, either by the arm or the mobile base. This is achieved by minimizing the number of spots min ($|\boldsymbol{M_b}|$) and configurations min ($|\boldsymbol{G}|$) needed for inspecting the machine. Concurrently, coverage has to be maximized and overlap minimized. Coverage $\boldsymbol{C_g}$ is defined by equation (1), which is the set of every element from the machine surface covered by the arm configurations and the mobile base positions. The overlap $\boldsymbol{O_g}$, defined by equation (2), is represented as the set of surfaces visible from the robot at least two times. In these two equations, the function cover() returns the surface elements $\boldsymbol{s}$ covered from a specific spot and arm configuration, denoted $\boldsymbol{Gm} = [\boldsymbol{m_b}^\top, \boldsymbol{q}^\top]^\top$. A set of pose around the machine is denoted $\mathbf{T} = \{\mathbf{Gm_1}, \ldots, \mathbf{Gm_h}\} \in \mathbb{R}^{9\times h}$. To efficiently detect and clean with a mobile manipulator robot, the algorithm needs to maximize $|\boldsymbol{C_g}|$ and minimize $|\boldsymbol{O_g}|$.

$$\mathbf{C_g} = \bigcup_{i=0}^{|\mathbf{T}|} \mathrm{cover}(\mathbf{S}, \mathbf{Gm_i}) \tag{1}$$

$$\mathbf{O_g} = \bigcup_{i=0}^{|\mathbf{T}|-1} \Big(\bigcup_{j=i+1}^{|\mathbf{T}|} (\mathrm{cover}(\mathbf{S}, \mathbf{Gm_i}) \cap \mathrm{cover}(\mathbf{S}, \mathbf{Gm_j})) \tag{2}$$

Mobile arm vision-based adaptive cleaning (MAVAC)

Our approach is divided into two interconnected parts, each targeting specific aspects of the cleaning process: offline computation, which generates trajectories for the mobile base and the arm to inspect the entire machine; and online execution, which adapts the arm's trajectory to the real environment and to the detected dust.

Offline computation

The objective of the offline computation is to generate a set of poses $\mathbf{T}$ around the machine for the mobile base and the robotic arm. These positions are used to detect dust on the machine. This section details the process of trajectory

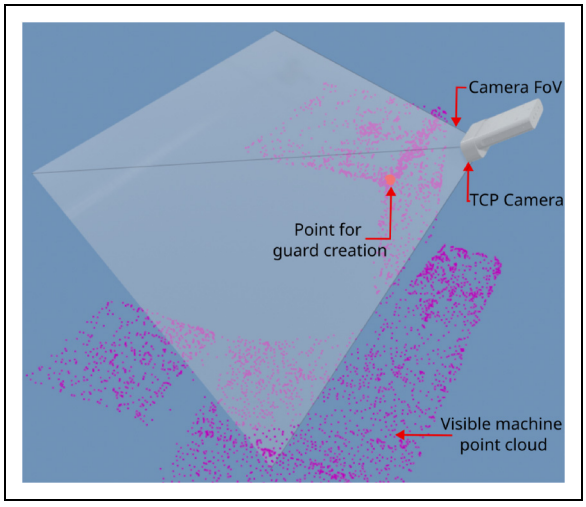

Figure 3. Illustration of the visible point definition used in the MAVAC algorithm. Purple points represent the set of surface points visible from the current camera position, as determined by the HPR algorithm. The white translucent pyramid represents the emulated camera FoV; all points within this volume are considered visible by the MAVAC method. The red point indicates the target surface used to determine the camera's guard position. MAVAC: mobile arm vision-based adaptive cleaning; HPR: hidden point removal; FOV: field of view.

generation and the exploration of complex surfaces. Figure 2 summarizes the method.

Robot base placement. The robot base placement strategy is derived from the work of Paus et al.,[1] using a relationship between potential spots and the corresponding accessible surfaces. An accessible surface from a spot is defined as a visible surface that is within the reachability map of the robot placed at this spot. The spots are chosen using a selection process that scores potential cleaning spots based on the number of accessible surface elements from a spot. This iterative selection process works in three steps:

- First, the spot with the highest score is added to the set ($\mathbf{M_b}$).
- Then, accessible surface elements belonging to this spot are removed from the accessible surfaces for all remaining spots. This ensures that only unaddressed surface elements remain for consideration.
- Potential spots are rescored, and the process is repeated until the newly highest score is lower than a predefined threshold.

The threshold is fixed at the first iteration by multiplying the highest score by a user-defined coefficient, $P_s \in [0, 1]$. This approach aims to prioritize spots that contribute significantly to surface coverage, thereby avoiding an excessive number of spots with minimal impact.

Local exploration. In order to thoroughly explore the machine using a camera fixed on the robot's TCP, a set of guard configurations per spot is established. The machine is observed from these guard configurations to identify and remove potential dust particles. A guard point is assigned to each surface of the machine, and the tool orientation is determined using the surface's normal vector, computed with the Open3D library.[24] The guard point is placed along this normal at a distance corresponding to the reliable detection limit of the camera. To generate a guard configuration, a collision-free configuration is calculated using inverse kinematics.

Camera field of view (FOV). To determine the most suitable guard points, the observable surface elements are estimated by emulating the camera's FOV. The camera FOV is modeled as a pyramid with a rectangular base, referred to as $E\,fov$, and defined by two angles θ_a and θ_b. Three conditions must be met to determine if a machine surface is included in the camera's FOV. To check these conditions, the surface is transformed into the camera frame and denoted $p_m = [x_c, y_c, z_c]$. The coordinate x_c has to be within the camera's detection limit, denoted L_d. The point p_m is projected on camera planes XY and XZ. For each projection, the angles between p_m and the X-axis, denoted, respectively, θ_{xy} and θ_{xz}, are evaluated to be included in $[-\theta_a, \theta_a]$ and $[-\theta_b, \theta_b]$, respectively. The emulated camera's FOV may extend beyond the machine walls, leading to incorrect classification of observable surfaces. To overcome this issue, the hidden point removal (HPR) method[25] is used. Figure 3 illustrates the position and orientation of the camera and surface elements seen by it.

Guard selection. Guard selection process is similar to the spot selection described in the "Offline computation" section. For each guard configuration, a relationship is established between the robot configuration and the surface elements visible from this configuration. To select the best guards, a score is attributed based on the maneuverability index of the configuration and the number of visible surface elements N, normalized by $|\mathbf{S}|$ as described in equation (3):

$$\text{Score} = \mathrm{I}_{\mathrm{m}}^{\mathrm{k}_1} \times \left(\frac{\mathrm{N}}{|\mathbf{S}|}\right)^{\mathrm{k}_2} \tag{3}$$

where $k_1 \in [0, 1]$ and $k_2 \in [0, 1]$ are two weighting factors that influence the score. When $k_1 = 0$ and $k_2 = 1$, the score is primarily influenced by the number of accessible surface elements. The maneuverability index $I_m \in [0, 1]$, introduced by Vahrenkamp et al.,[26] is defined as the product of the robot's manipulability—representing the distance from singular configurations—and its distance to joint limits. A value of $I_m = 0$ indicates that the robot is completely constrained and unable to move within the task space.

$$J_l(\boldsymbol{\theta}) = 1 - \exp\left(-k \prod_{j=1}^{|\boldsymbol{\theta}|} \frac{(\theta_j - l_j^-)(l_j^+ - \theta_j)}{(l_j^+ - l_j^-)^2}\right) \tag{4}$$

where k is a scale factor and $\{l^+, l^-\}$ are the upper and lower joint limits.

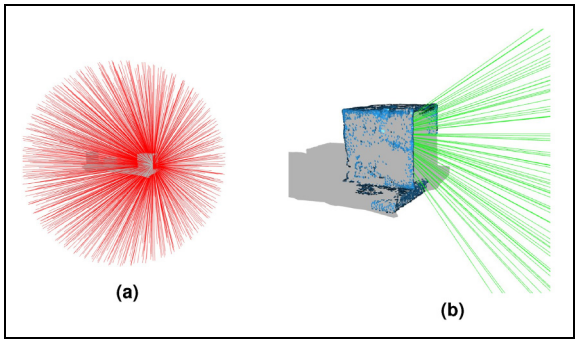

Figure 4. Illustration of the method to find the best gateway for a non-observed surface (NoS). (a) Rays (red) cast outward the machine mesh (gray) and (b) rays (green) exiting from the best gateway of the NoS (blue).

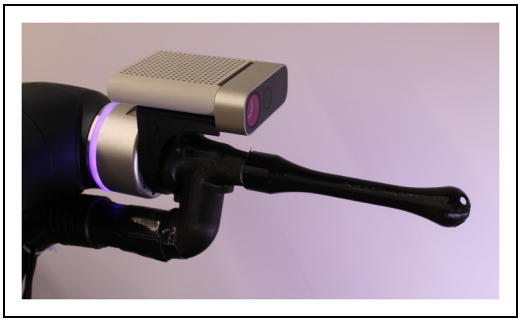

Figure 5. Robot tool with the Azure Kinect camera on the top.

Algorithm 1 Online algorithm

Require: Offline pose set : T
for All Gm in T **do**
  moveMobileBase($Gm[0]$)
  moveArm($Gm[1]$)
  $dClustList$ = detectClusterizeTspDust()
  for all D in $dClustList$ **do**
    $succes, p_a$ = findValidApprochPoint(D)
    if $succes$ **then**
      moveArm(p_a)
      for all d in d **do**
        setImpedance()
        moveArm(d)
      end for
      moveArm(p_a)
      stopImpedance()
    end if
  end for
end for

Steps of the iterative selection process are applied similarly for the guards' selection. The threshold is defined at the first iteration by multiplying the highest score by the user-defined coefficient $P_g \in [0, 1]$. Selected guards are stored in Gm.

Non-observed surface (NoS). After the evaluation and selection of all guards, it is possible that some surfaces remain non-observable, for example, due to concavities. To address this problem, these surfaces are clustered using a DBscan clustering algorithm.[27] Then, the cluster's center is determined, and multiple rays are projected outward from this center across the machine mesh, as depicted by Figure 4(a). Only rays that do not encounter collisions with the machine mesh are considered, and they are grouped to determine gateways to these surfaces. The gateway with the highest number of rays for a surface cluster is regarded as the best gateway, as shown in Figure 4(b). To generate a trajectory leading to these surfaces, the closest spot is selected considering the average orientation of rays, projected on the machine floor, exiting from the gateway.

Subsequently, a collision-free path is sought for the guard point resulting from this operation, called NoS guard. If a path is valid, the NoS guard is incorporated into the offline trajectory.

Online execution

Using the offline trajectory, the online execution follows several steps essential for efficient inspection and cleaning: dust detection, local trajectory generation, and dust collection. Algorithm 1 summarizes these steps.

Dust detection. This article does not primarily focus on dust detection, as it is more of a perception/segmentation problem; therefore, a simplified representation is used, with green dust to facilitate segmentation. The green dust particles are segmented from the images, and the central points of these segmented blobs are identified as nodes, denoted by $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_d\} \in \mathbb{R}^{3\times d}$, which the vacuum targets for cleaning. To ensure the accuracy of dust particle detection, various filters are applied to eliminate non-reachable dust particles and false detections. These filters are specifically designed to remove any dust particles that are beyond the robot's theoretically reachable workspace or are incorrectly identified due to noise.

Trajectory generation. Nodes in $\mathbf{D}$ are split into subsets with a clustering algorithm, DBScan,[27] to determine areas to clean. For each node of an area, an approach point $\mathbf{p}_a$ is defined to position the robot vacuum tool above the node. This approach point is calculated using equation (5), where roll_g, pitch_g and yaw_g represents the orientation of the current guard point. The aim is to position the tool with the same orientation as the corresponding guard point with a tolerance of α to have a greater leeway to find a feasible path. Our method assumes that a node observed at a specific guard position and orientation should be accessible with the same orientation during the cleaning process. To reach $\mathbf{p}_a$, a trajectory is generated using a path planner that considers obstacle avoidance based on the octomap

Table 1. C, O, and the number of spots as a function of P_s.

P_s	Number of spots	C	O
0.01	11	92.8%	5.7%
0.02	9	91.9%	5.5%
0.05	6	87.4%	5%
0.10	4	82.4%	4%
0.15	4	82.4%	4%
0.20	4	82.4%	4%
0.30	2	62.8%	1.9%

derived from the 3D camera and the robot's geometry. That path planner is RRT-connect,[28] as considered by Ahmadi et al.,[29] one of the fastest algorithms for path planning in complex environments in the Open Motion Planning Library. If the approach point is not reachable, another dust heap from the area is used to create a new approach point $\mathbf{p}_a$, until a $\mathbf{p}_a$ is reachable. If no $\mathbf{p}_a$ of the area is reachable, the area is considered as unreachable from this guard and may be accessible from another guard point.

$$\mathbf{p}_a = \begin{bmatrix} d_x \\ d_y \\ d_z - \delta \\ \text{roll}_g \pm \alpha \\ \text{pitch}_g \pm \alpha \\ \text{yaw}_g \pm \alpha \end{bmatrix} \tag{5}$$

Dust collecting. Once the robot reaches $\mathbf{p}_a$, it performs a linear movement to directly engage with the dust and vacuum it. To mitigate risks of damage or collision with the machine, the robot is controlled in impedance during dust collection. This part of the dust collection could be improved using feedback regarding the quality of the cleaning. A specialized tool has been developed to enable omnidirectional vacuuming by the robot. Illustrated in Figure 5, this tool is designed with an elongated tube and a perforated sphere at its end. This tool enhances the robot's capabilities, enabling it to reach into confined spaces. After cleaning an area, the robot returns to the approach point and proceeds to the next area. This strategy ensures that the robot exits an area in the same manner it entered, thereby eliminating the risk of becoming trapped in a concavity.

Experimental setup

The robot mobile manipulator used in the trials consists of a MiR200 mobile robot and a Doosan M1013 manipulator arm. The manipulator arm has been strategically positioned at the center of the MiR200, thereby avoiding forward tilting and maximizing its motion range. Our approach is implemented on a computer equipped with an i9-12900H processor and an Nvidia RTX A2000 8 GB graphics card. The operating system used is Ubuntu 20.04. For real-time control, the system relies on Moveit! library, which runs on the ROS Noetic middleware. Moveit! is employed with real-time collision detection, using an octomap with a resolution of 2 cm. The TCP-mounted camera configuration, together with the offline exploration strategy (including the NoS method for concavities), ensures that all surfaces/obstacles are observed from appropriate viewpoints before trajectory generation. This approach enables the robot to safely vacuum even in geometrically challenging areas.

To ensure both the safety of the robot and the integrity of the machine, we use impedance control when operating close to the machine.

To benchmark our approach, a self-made test bench measuring 2 m×1 m×1 m has been used. This test bench has been designed based on a real industrial machine and is made of four boxes, representing varying levels of machine complexity, including concavities. Additionally, two machines, a sanding machine and a milling machine in a fabrication laboratory, also serve as testbeds in our trials. They typically generate a significant amount of dust and feature complex structures with concavities and hard-to-reach areas.

For evaluation purposes, green dust heaps are evenly distributed at intervals of 10 cm on the surface, including within the concavities. Green dust detection is performed using an Azure Kinect camera and a simple HSV filter with fine-tuned parameters. For real-world industrial cleaning applications, other detection methods could be implemented, as outlined in the "Industrial dust detection" section.

Results

This section evaluates the offline computation method used to configure the parameters for the online execution phase, which is also analyzed.

Offline computation

Offline computation relies on four key parameters: P_s, P_g, k_1, and k_2. To fine-tune these parameters, the percentage of coverage $\mathbf{C}$ and overlap $\mathbf{O}$ are estimated from $|\mathbf{C_g}|$ and $|\mathbf{O_g}|$, respectively, divided by $|\mathbf{S}|$. It is important to note that these parameters should be tuned for each machine, as optimal values depend on the machine's geometry, size, and complexity. The test bench setup is used here as a reference example to demonstrate the tuning methodology.

To evaluate P_s, values ranging from 0.01 to 0.3 were sampled. This range was selected based on preliminary observations: values below 0.01 generate an excessive number of spots, while values above 0.3 result in insufficient coverage. For each P_s value, the iterative spot selection process described in Section "Offline computation" was executed to determine the resulting number of spots,

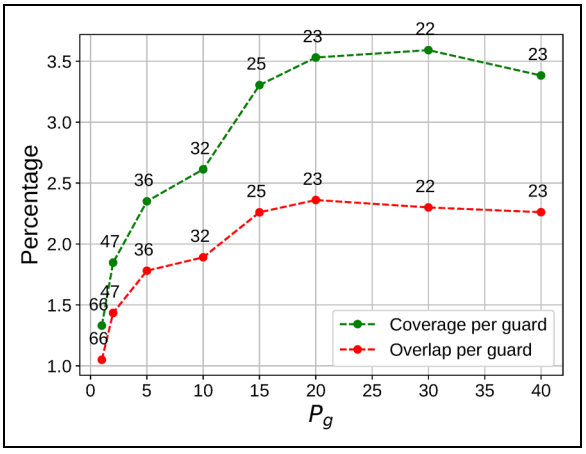

Figure 6. Average percentage of coverage (green) and overlap (red) per **guard point** in relation to percentage score limit, P_g. The number of guards for P_g is indicated above the experimental points.

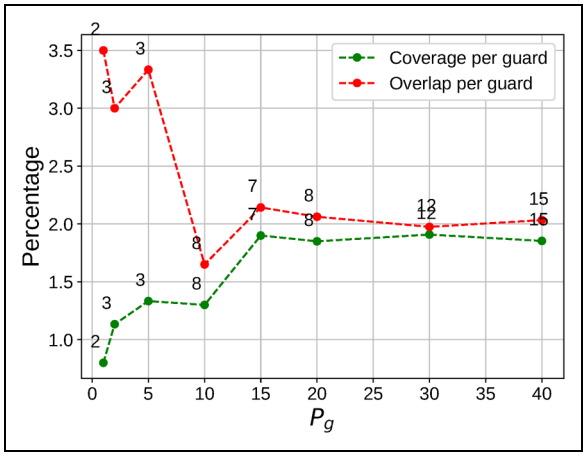

Figure 7. Average percentage of coverage and overlap per non-observed surface **(NoS) guard** point in relation to percentage of P_g. The number of guards for P_g is indicated above the experimental points.

coverage **C**, and overlap **O**. Table 1 presents the average results obtained for each value of P_s over 12 trials.

The optimal value of P_s must balance three competing objectives: maximizing coverage **C**, minimizing overlap **O**, and minimizing the number of spots. As shown in Table 1, lower P_s values (0.01 to 0.02) achieve higher coverage (92%) but require significantly more spots (9 to 11), which increases inspection time and overlap. Conversely, higher P_s values (0.30) drastically reduce the number of spots (2) and overlap (1.9%) but at the cost of unacceptably low coverage (62.8%). Based on these results for the test bench, P_s values of 0.10, 0.15, or 0.20 provide the best tradeoff, all converging to four spots with 82.4% coverage and 4% overlap. This configuration balances efficient inspection (few spots) with adequate coverage for effective dust detection. For the test bench experiments, $P_s = 0.15$ was selected.

The parameter P_g affects the number of guard points assigned to each spot. To evaluate this parameter, the number of spots was fixed at 4, as determined from the optimal P_s value. Figure 6 shows the average percentage of coverage and overlap per guard point as a function of P_g. Eight values of P_g were sampled based on empirical observations. The curves exhibit quasi-asymptotic behavior, converging to approximately 3.5% coverage and 2.4% overlap per guard point. Similarly, beyond $P_g = 15$, the total number of guards stabilizes in the range of 22 to 25, indicating diminishing returns for higher threshold values.

Following the same tradeoff principles applied to P_s selection, $P_g = 15\%$ was chosen for the test bench, as it provides near-maximal coverage efficiency while maintaining a reasonable number of guards. While the detailed analysis above describes the tuning methodology using the test bench, similar evaluations were performed for the sanding and milling machines to determine their respective optimal parameters. Although the exact parameter values varied slightly between machines due to differences in their geometries and complexities, the optimal configurations followed similar trends to those observed for the test bench. Machine-specific tuning is necessary to achieve optimal performance, though the general tradeoff principles and selection methodology remain consistent across different machines.

Parameters k_1 and k_2 influence the maneuverability of the robot at guard points. However, these parameters do not affect C or O. A free-moving robot at guard configurations is needed to ensure a successful cleaning. Therefore, k_1 should not be equal to zero. For online execution tests, they are fixed to $k_1 = k_2 = 1$, to give the same weight to both.

For a fixed P_s, the number of NoS guards depends on the variable P_g. Figure 7 shows the average percentages of C and O per NoS guards as a function of P_g. We observe that having an adequate number of NoS guards is important to ensuring observation of each concavity and increasing the chances of accessing and cleaning these regions. However, it is preferable to limit their number, as they do not account for overlap.

As a result, to inspect machines following the requirements from the "Cleaning task formulation" section, we suggest to set $P_s = 0.02$, $P_g = 15$, and $k_1 = k_2 = 1$.

Online execution

Ten trials were carried out on three scenarios as shown in Figures 8(a) to 8(c). Table 2 presents the percentage of dust heaps vacuumed calculated at the end of each trial. The milling and sanding machines have been cleaned during the same cleaning test. To provide a general indication of the task completion time, a few trials were timed and are presented as illustrative examples. Machine sizes are reflected by the number of dust heaps since they are evenly

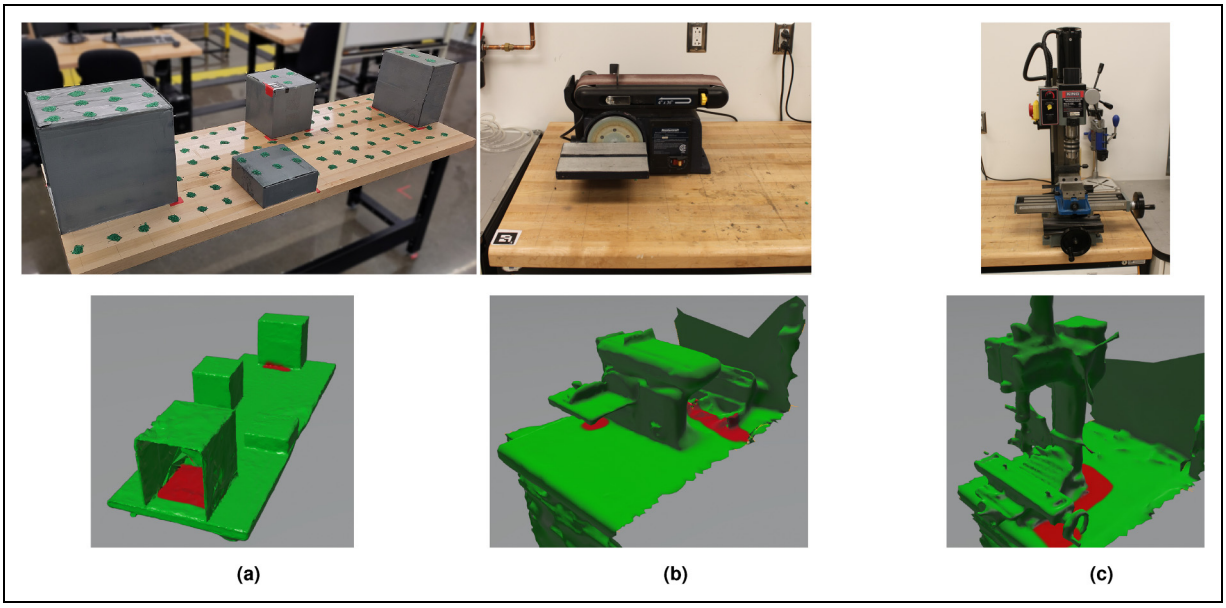

Figure 8. Result of the cleaning process. The top images show the tested machines. Lower images show the cleaned areas in green and uncleaned areas in red. (a) Test bench, (b) sanding machine, and (c) milling machine.

distributed. For these cleaning trials, the robot achieved an average cleaning efficiency of 84%, indicating that a significant portion of the dust on the machines was removed, although some areas remained uncleaned. The 4% standard deviation of cleaning efficiency across 10 trials conducted on three machines with significantly different morphologies suggests that our method can adapt to various types of machines.

While MAVAC is slower than manual cleaning, direct comparison with other robotic methods is difficult, as most studies report only simulated coverage rather than real cleaning times.

Two implementation issues were identified. First, during the milling machine cleaning, the robot detected dust inside the box. However, the path planner failed to generate a valid route due to the tool's large size. Extending the suction tube could allow access to this area. However, this might introduce new issues, such as the camera's blind spot widening. The second point concerns the RRT-connect path planning algorithm, which is highly central processing unit-intensive and requires a significant amount of time to generate a trajectory in densely cluttered environments. Currently, newly designed path planners for highly cluttered environments, such as the one proposed by Balakumar et al.,[30] could provide a potential solution. These planners appear to be specifically tailored for such environments, incorporating real-time collision detection with an improved resolution of around 1 cm.

Table 2. Results of cleaning tests.

Machine	Cleaning percentage	Dust heaps	Uncleaned dust heaps	Cleaning time (min)
Sanding and milling machine	79%	150	31	–
	80%	158	31	–
	82%	158	28	–
	79%	155	33	–
	81%	160	30	45
Test bench	89%	144	16	36
	84%	146	23	38
	88%	146	17	39
	90%	145	15	–
	88%	147	17	–

Discussion

The primary research contribution of MAVAC is the combination of offline reachability/visibility-based planning with online dust-adaptive cleaning, validated experimentally on real machines with complex geometries.

While a direct quantitative comparison of time savings and motion reduction against systematic coverage methods would be valuable, such benchmarking is challenging due to the lack of established baselines in the literature. Previous work on robotic cleaning typically reports coverage percentages in simulation rather than actual cleaning times or motion metrics.[1,11,22] Qualitatively, MAVAC's adaptive approach provides inherent efficiency advantages over systematic coverage methods: the robot only generates cleaning trajectories to detected dust locations rather than sweeping entire surfaces. By avoiding redundant motion over clean areas, the method prioritizes efficiency without

sacrificing cleaning effectiveness. This distinction becomes increasingly significant for large or complex machines where dust distribution is sparse or localized. The primary tradeoff lies in the computational overhead of dust detection and adaptive trajectory planning. However, as demonstrated in our experiments, the online detection and planning process operates efficiently enough to maintain practical cleaning times (36 to 45 min for machines with 144 to 160 dust heaps distributed across surfaces ranging from 5 to 6 m^2), while the offline computation (performed once per machine) requires minimal setup time. Future work should establish standardized benchmarks for comparing cleaning efficiency across different methods, including metrics such as cleaning time per unit area, total trajectory length, and energy consumption, to enable more rigorous quantitative comparisons.

Another important contribution of MAVAC lies in its definition of "accessible surfaces." By evaluating surfaces that can truly be reached and cleaned, MAVAC provides a more realistic assessment than methods relying solely on mesh collision checks or nominal reachability maps. This refined definition explains why MAVAC's reported coverage percentages may be lower than those typically found in the literature. Part of this discrepancy is due to occasional misclassifications by the HPR method, which can incorrectly distinguish between visible and non-visible surfaces. However, these lower coverage values do not necessarily mean that dust remains uncleaned; on the contrary, our experiments show that MAVAC can reach challenging areas (e.g. inside concavities) that standard methods might miss.

A further strength of MAVAC is its reliance on observation trajectories planned with respect to joint configurations, rather than solely on Cartesian paths or static assumptions about machine geometry. This design choice, differing from many existing methods,[1,22] enables the cleaning trajectories to adapt seamlessly to real-world changes in the workspace. Such adaptability is particularly beneficial in small- and medium-sized enterprises, where machine layouts are frequently modified. Indeed, between our tests in the fabrication laboratory, the machines were slightly rearranged, yet no additional scanning was required to update the cleaning plan.

In contrast to AI- or RL-based cleaning approaches,[3,4] MAVAC does not require large datasets or long training phases, and it has been validated on industrial machines with complex geometries. Previous learning-based approaches have primarily been demonstrated on simple flat surfaces such as tables or floors, whereas MAVAC addresses the challenges of concavities and irregular machine structures. This distinction highlights MAVAC's potential for practical deployment in real industrial environments.

While the current implementation still leaves room for improvement, particularly in visibility classification and trajectory computation, these challenges are shared by most state-of-the-art methods. The main opportunities for enhancement are:

- **Visibility classification:** Refining or replacing the HPR algorithm could reduce the rate of misclassifications, thereby increasing both precision in coverage estimation and confidence in uncleaned areas.
- **Trajectory optimization:** In our setup, RRT-connect required several seconds per plan due to high obstacle density and fine resolution. Future work could incorporate graphics processing unit-based planners such as Curobo,[30] which demonstrate subsecond planning in cluttered environments, or hybrid approaches combining precomputed offline paths with online adaptation.
- **Dust detection**: In practical industrial settings, dust exhibits a wide variety of textures, colors, and lighting conditions, making perception significantly more challenging than our controlled evaluation with green dust. To enhance MAVAC's practical relevance, it can be integrated with advanced vision-based dust detection methods. For instance, recent work by Chen et al.[17] demonstrates reliable dust detection in real-world environments. Similarly, Singh et al.[31] proposed a vision-based system using YOLOv5 with DeepSORT to generate dirt distribution maps from complex dirt types and surfaces. These examples illustrate how MAVAC's adaptive cleaning logic can be paired with robust perception pipelines to support deployment in varied industrial scenarios.

Overall, MAVAC advances robotic cleaning by introducing an adaptable and experimentally validated framework that directly addresses the limitations of both systematic coverage and learning-based approaches.

Conclusion

This paper introduces an approach for cleaning machines in industrial environments. By cleaning only the detected dust, this method is designed to be inherently more efficient than systematic covering methods and suitable for complex machines. The approach involves the inspection of machines using an RGB-D camera attached to a mobile robot manipulator's end effector. The offline inspection path, informed by a coarse model of the machine, facilitates precise dust detection and the generation of an online cleaning path. The approach has been successfully tested on a test bench as well as on two real machines from a fabrication laboratory. The experimental results demonstrate that the cleaning task can be accomplished with an 84% average success rate. By developing suitable detection techniques,

this method could be adapted to various types of dust or even extended to other tasks, such as painting or sanding.

Future work will focus on solving the problem of unreachable areas. A solution could be to adapt the reachability map of the robot arm by moving the mobile base during the cleaning process.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the Collaborative REsearch And Training Experience (CREATE) program, specifically the Collaborative Robotics for Manufacturing (CoRoM) program, Sycodal Electrotechnique Inc., and by Mitacs via the Mitacs Accelerate program.

ORCID iDs

Guillaume Dupoiron https://orcid.org/0009-0007-8420-4789
François Michaud https://orcid.org/0000-0002-3639-7770
Jean-Philippe Roberge https://orcid.org/0000-0003-0265-9020

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the CREATE Program on Collaborative Robotics for Manufacturing (CoRoM), and by Mitacs and Sycodal Electrotechnique Inc. through the Mitacs Acceleration program (Grant IT25843).

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Paus F, Kaiser P, Vahrenkamp N, et al. A combined approach for robot placement and coverage path planning for mobile manipulation. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Vancouver, BC, Canada, 2017, pp.6285–6292. DOI: 10.1109/IROS.2017.8206531.
2. Qi L, Gan Z, Hua Z, et al. Cleaning of object surfaces based on deep learning: A method for generating manipulator trajectories using RGB-D semantic segmentation. *Neural Comput Appl* 2023; 35: 8677–8692. DOI: 10.1007/s00521-022-07930-x.
3. Gaurav S, Crookes A, Hoying D, et al.Robot learning to mop like humans using video demonstrations. In: *2023 IEEE/RSJ international conference on intelligent robots and systems (IROS). Detroit, MI, USA* 2023, pp.9947–9954. DOI: 10.1109/IROS55552.2023.10342231.
4. Lew T, Singh S, Prats M, et al.Robotic table wiping via reinforcement learning and whole-body trajectory optimization. In: *2023 IEEE international conference on robotics and automation (ICRA)*. London, UK, 2023, pp.7184–7190. DOI: 10.1109/ICRA48891.2023.10161283.
5. Yin J, Apuroop K, Tamilselvam Y, et al. Table cleaning task by human support robot using deep learning technique. *Sensors* 2020; 20: 1698.
6. Labbé M and Michaud F. RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *J Field Robot* 2019; 36: 416–446.
7. Kim J, Mishra A, Limosani R, et al. Control strategies for cleaning robots in domestic applications: A comprehensive review. *Int J Adv Robot Syst* 2019; 16(4): 1–22.
8. Makhal A and Goins A. Reuleaux: Robot base placement by reachability analysis. *CoRR* 2017; abs/1710.01328.
9. Vahrenkamp N, Asfour T and Dillmann R. Robot placement based on reachability inversion. In:*2013 IEEE international conference on robotics and automation*. Karlsruhe, Germany, 2013, pp.1970–1975. DOI: 10.1109/ICRA.2013.6630839.
10. Porges O, Stouraitis T, Borst C, et al. Reachability and capability analysis for manipulation tasks. In: *ROBOT2013: First iberian robotics conference, Madrid, Spain. Advances in intelligent systems and computing*. Vol. 253. Cham: Springer. DOI: 10.1007/978-3-319-03653-3_50.
11. Hassan M, Liu D and Paul G. Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements. In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Daejeon, Korea (South), 2016, pp.2978–2984. DOI: 10.1109/IROS.2016.7759461.
12. Kalawoun R, Lengagne S and Mezouar Y. Optimal robot base placements for coverage tasks. In: *2018 IEEE 14th international conference on automation science and engineering (CASE)*. Munich, Germany, 2018, pp.235–240. DOI: 10.1109/COASE.2018.8560402.
13. Ricardez GAG, Koganti N, Yang PC, et al. Adaptive motion generation using imitation learning and highly compliant end effector for autonomous cleaning. *Adv Robot* 2020; 34: 189–201.
14. Wakayama T, Fujiura E, Yamaguchi M, et al. Versatile cleaning service robot based on a mobile manipulator with tool switching for liquids and garbage removal in restrooms. *Adv Robot* 2022; 36: 967–981.
15. Martinez D, Alenyà G and Torras C. Planning robot manipulation to clean planar surfaces. *Eng Appl Artif Intell* 2015; 39: 23–32.
16. Ramalingam B, Yin J, Rajesh Elara M, et al. A human support robot for the cleaning and maintenance of door handles using a deep-learning framework. *Sensors* 2020; 20: 1–18.
17. Chen LW, Zhu J, hui Zhang H, et al. Dust detection and cleanliness assessment based on s-yolov5s for npp reactor containment wall-climbing cleaning robot. *Heliyon* 2024; 10: 1–11. DOI: 10.1016/j.heliyon.2024.e24220.
18. Jaeseok K, Nino C, Pedro V, et al. "iCub, clean the table!" A robot learning from demonstration approach using deep neural networks. In: *2018 international conference on*

autonomous robot systems and competitions (ICARSC). Torres Vedras, Portugal, 2018, pp.3–9. DOI: 10.1109/ICARSC.2018.8374152.

19. Cauli N, Vicente P, Kim J, et al. Autonomous table-cleaning from kinesthetic demonstrations using deep learning. In: *2018 Joint IEEE 8th international conference on development and learning and epigenetic robotics (ICDL-EpiRob)*. Tokyo, Japan, 2018, pp.26–32. DOI: 10.1109/DEVLRN.2018.8761013.
20. Saito N, Wang D, Ogata T, et al. Wiping 3D-objects using deep learning model based on image/force/joint information. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Las Vegas, NV, USA, 2020, pp.10152–10157. DOI: 10.1109/IROS45743.2020.9341275.
21. Vatavuk I, Vasiljevic G and Kovacic Z. Task space model predictive control for vineyard spraying with a mobile manipulator. *Agriculture* 2022; 12: 1–20. DOI: 10.3390/agriculture12030381.
22. Zhang H, Mi K and Zhang Z. Base placement optimization for coverage mobile manipulation tasks. *arXiv*: 230408246, 2023.
23. Yang PC, Koganti N, Alfonso GG, et al. Context dependent trajectory generation using sequence-to-sequence models for robotic toilet cleaning. In: *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*. Naples, Italy, 2020, pp.932–937. DOI: 10.1109/RO-MAN47096.2020.9223341.
24. Zhou QY, Park J and Koltun V. Open3D: A modern library for 3D data processing. *arXiv*: 180109847, 2018.
25. Katz S, Tal A and Basri R. Direct visibility of point sets. *ACM Trans Graph* 2007; 26: 1–12.
26. Vahrenkamp N, Asfour T, Metta G, et al. Manipulability analysis. In: *2012 12th IEEE-RAS international conference on humanoid robots (Humanoids 2012)*. Osaka, Japan, 2012, pp.568–573. DOI: 10.1109/HUMANOIDS.2012.6651576.
27. Ester M, Kriegel HP, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceeding of the second international conference on knowledge discovery and data mining*. Portland, OR: AAAI Press, 1996, pp.226–231.
28. Kuffner J and LaValle S. RRT-connect: An efficient approach to single-query path planning. In: *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No.00CH37065)*. Vol. 2. San Francisco, CA, USA, 2000, pp.995–1001. DOI: 10.1109/ROBOT.2000.844730.
29. Ahmadi S. *Real-time motion planning of 6 DOF collaborative robot*. Master's Thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2022.
30. Sundaralingam B, Hari SKS, Fishman A, et al. cuRobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv*: 2310.17274, 2023.
31. Singh IS, Wijegunawardana ID, Samarakoon SBP, et al. Vision-based dirt distribution mapping using deep learning. *Sci Rep* 2023; 13: 1–11 https://api.semanticscholar.org/CorpusID:260681108.