Received 9 June 2024, accepted 19 August 2024, date of publication 21 August 2024, date of current version 2 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3447158

RESEARCH ARTICLE

Autonomous Scanning and Cleanliness Classification of Pharmaceutical Bins Through Artificial Intelligence and Robotics

SIMONE COMARI AND MARCO CARRICATO, (Senior Member, IEEE)
Department of Industrial Engineering, IRMA L@B, University of Bologna, 40136 Bologna, Italy

Corresponding author: Marco Carricato (marco.carricato@unibo.it)

This work was supported in part by the Italian Ministry for Economic Development and the Italian Emilia-Romagna Region through the Systems ENabling Efficient Cognitive Automation (SENECA) Project under Grant CUP B76G19000520008.

ABSTRACT In the pharmaceutical industry, bins need to be cleaned up to a critical level because the products that they contain are often incompatible with each other, and their mixture can facilitate the formation of bacterial fauna. In this work, a strategy is presented to fully automatize the procedure of cleanliness quality inspection of a pharmaceutical bin through a robotic arm and the use of both traditional and artificial-intelligence-based computer-vision techniques. An autonomous mobile robot is used to mimic the approach of a manipulator to a bin inserted in a washing cabin with an uncertain position. The manipulator is equipped with an eye-on-hand color camera and it carries out the binary classification of the bin surface status (e.g. clean vs dirty) through a convolutional neural network based on ResNet. The viewpoints from which the images are taken are the result of an optimization that, starting from the digital three-dimensional model of the bin and exploiting a virtual-twin-based planning scene, minimizes their number while maximizing the visible area of the bin from the current location of the robot. The results of this optimization are used to set up a pipeline that is entirely bin-independent. The same procedure may also be employed to generate the best washing trajectories to be performed by the cleaning robot, by simply replacing the inspection camera mounted on the robot end-effector with a washing nozzle. Though a complete tuning session is still required, preliminary experimental results are very promising, reaching a classifier accuracy (namely a capability of distinguishing clean and dirty surfaces) of 98% on conditioned data, showing that this work has the potential of becoming an effective and versatile industrial product.

INDEX TERMS Artificial intelligence, computer vision, object scanning, quality inspection, robotics, surface cleanliness, trajectory planning.

I. INTRODUCTION

In the pharmaceutical industry, the containers used for storing, manipulating or mixing powders that will eventually become pills or tablets are called *bins*.

Pharmaceutical bins need to be cleaned up to a level known as *critical*, because the products that they contain are often incompatible with each other, and their mixture can facilitate the formation of bacterial fauna.

The associate editor coordinating the review of this manuscript and approving it for publication was Davide Patti.

To avoid exposure to contaminants harmful to humans, as well as to speed up the process, it is preferred to automate the cleaning of the bins by means of specialised fully automatic washing booths, where the external and internal surfaces of the bin are cleaned by conveniently located washing nozzles. This solution guarantees high levels of cleanliness and repeatability, but it is poorly reconfigurable and can handle a limited number of bin types. This is a problem, since, in the absence of specific regulations or standards, every company uses bins with customized shapes and dimensions, adapted to its production department. A more flexible solution is a mobile module associated with a cleaning room where an operator manually carries out the cleaning operation with a washing nozzle. However, this solution decreases the efficacy of the procedure (which is not fully automatised) and increases the risk for the human operator.

Darragjati [1] proposed the use of a robotic manipulator to automate the manual operation within the cleaning room, proving that this solution is as economically viable as a washing booth, but with increased flexibility and the possibility to intensify cleaning in the critical areas of the bin, such as corners and junctions. A virtual layout of the envisioned setup is shown in Fig. 1, where a bin located in a washing cabin is cleaned by a robotic arm installed on the cabin ceiling. The robot handles the washing nozzle connected by a hose to the mobile module containing all cleaning and drying hardware (pumps, heaters, etc.). In [1], however, washing trajectories were manually calculated based on the unique shape of a single bin, which is a long and tedious job for the programmer in charge. Moreover, the system is unable to recognize whether the bin surface is clean or dirty; thus, it is not capable of autonomously assessing the quality of the cleaning operation.

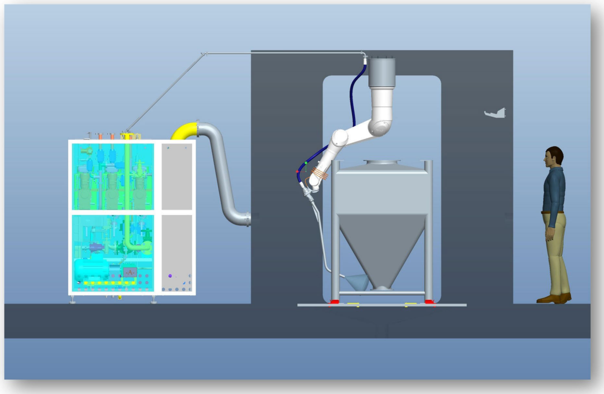


FIGURE 1. Washing module with a cleaning robotic arm inside a washing cabin [1].

This article aims to extend the autonomy and flexibility of the robotic system in Fig. 1, by introducing vision sensors and artificial-intelligence-based algorithms, in the perspective of making the cleaning and quality-inspection operations fully autonomous. The presented framework can cope with an arbitrary location of the robotic arm with respect to the bin, so as to cover a completely generic configuration of the robotized cell and thus increase versatility.

The complete cleaning process comprises five stages to automatize: identification of the bin type and pose; bin scanning to generate suitable trajectories for cleanliness inspection; classification of bin surface areas as dirty or clean; planning of robot trajectories for washing; bin cleaning. The current paper is focused on the first three stages, though it sets the basis for effectively solving the fourth stage too.

The main original contributions of the paper are:

- the implementation of a procedure for the automatic generation of a scanning path given a CAD model of a pharmaceutical bin, while accounting for visibility constraints, robot kinematic limitations and collision avoidance;
- the realization of a deep-learning-based binary classifier for assessing whether a bin (internal or external) surface is clean or dirty.

Accordingly, the main novelty lies in developing an integrated robotic vision system, which combines path-planning techniques (using the CAD model of the object to scan) with an AI-based surface inspection module (using deep learning on camera images). The application to autonomous cleanliness inspection of pharmaceutical bins is presented as a case study to validate the proposed framework.

Though the actual cleaning of the bins goes beyond the scope of the paper, the proposed path-planning procedure can easily be adapted to generate the washing trajectories to be performed by the cleaning robot, by simply replacing the inspection camera mounted on the robot end-effector with a washing nozzle and changing the camera pin-hole model with with the model describing the nozzle frustum of action.

Since the presented framework can cope with an arbitrary location of the robotic arm with respect to the object to be inspected and cleaned, the robot can be installed on an autonomous mobile platform [2], [3]. This allows the robot to approach the object from different sides and thus always guarantee optimal relative positioning, also allowing the robot to serve different robotized cells to increase versatility. To minimize execution time, a dual-arm setup could also be envisioned, so that operations could be shared between multiple robots and completed faster [4], [5].

The framework presented in the paper does not depend on the bin type and shape, thus it can be extended to any other inspection/cleaning problem, provided that the CAD model of the object to scan is available. The proposed solution can be especially favourable in all those sectors where stringent cleanliness standards apply, and the presence of human operators is discouraged, such as in medical and pharmaceutical industries, food processing and packaging, electronics and aerospace manufacturing. Other relevant applications can be found whenever an object or product needs to be scanned to assess its manufacturing quality (mechanical pieces, ceramics, etc.).

As far as the organization of the paper is concerned, Sections II introduces the adopted materials and methods. Sections III, IV and V describe, respectively, bin identification, surface scanning and cleanliness classification, presenting software framework, implementation and validation. Finally, Section VI draws conclusions and outlines future research directions.

II. MATERIALS AND METHODS

As mentioned in the Introduction, the complete cleaning process comprises five stages:

1) **Bin identification**: assuming a discrete and known number of bins available to a pharmaceutical company, and assuming that a 3D CAD model exists for each one of them, the automatic system autonomously identifies the bin type and where the bin is positioned with respect to the robotic arm inside the cleaning room.
2) **Bin-surface scanning**: the CAD model of the identified bin and its pose relative to the robot are exploited to automatically generate suitable trajectories allowing the robot to scan the inner and outer bin surfaces via an eye-in-hand colour camera, taking into account visibility constraints, robot kinematic limitations and collision avoidance.
3) **Cleanliness binary classification**: a pre-trained convolutional neural network establishes whether the areas framed in the scanned image are dirty or clean (this operation involves collecting sample images showing dirty and clean surfaces to train the algorithm).
4) **Trajectory planning for bin cleaning**: the output of the surface-cleanliness classification is used to plan the robot trajectories for the washing operation.
5) **Bin cleaning**: the robot cleans the inner and outer bin surfaces via a hand-held washing nozzle.

Though this paper focuses on the first three stages only (in Sections III, IV and V, respectively), the procedures set up for the second and third stages can easily be adapted to generate the washing trajectories to be performed by the robot in the fourth stage, by replacing the inspection camera mounted on the robot end-effector with a washing nozzle and changing the camera pin-hole model with the model defining the nozzle frustum of action.

Pharmaceutical bins exhibit a great variety of shapes and dimensions. The bin used in this paper is represented in Fig. 2 as a case study: it is 1.030 mm high and has a roughly rectangular footprint with dimensions 880 $\times$ 740 mm. However, the procedures that will be conceived and developed will be completely bin-independent, provided that a CAD model of the bin is available. The bin is usually made of stainless steel, extensively polished inside. This means that the internal surface is much more reflective than the external one, which appears more opaque. This feature motivates the need for two neural networks to distinguish dirty and clean areas in the two scenarios (see Section V).

For simulation and experimental testing of the procedures related to the first three stages mentioned above (bin identification, scanning and cleanliness classification), the robotic platform shown in Fig. 3 is used, consisting of a 6-DOF robotic arm (UR10e by Universal Robots) equipped with an eye-in-hand 2D RGB colour camera (RealSense D435 by Intel) and mounted on an autonomous guided vehicle (MiR500 by MiR), so that an arbitrary location of the robot with respect to the bin can be reproduced. The robotic platform in Fig. 3, borrowed from the European project ROSSINI [6], has two robotic arms. Still, only one arm is used for the simulations and experiments described in the paper.

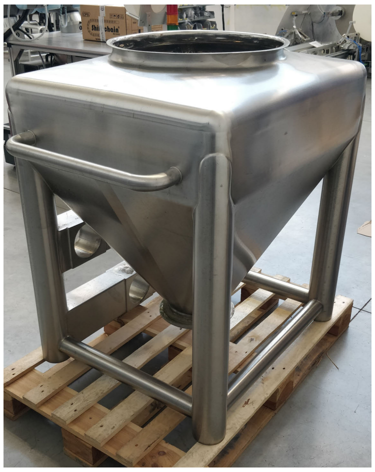
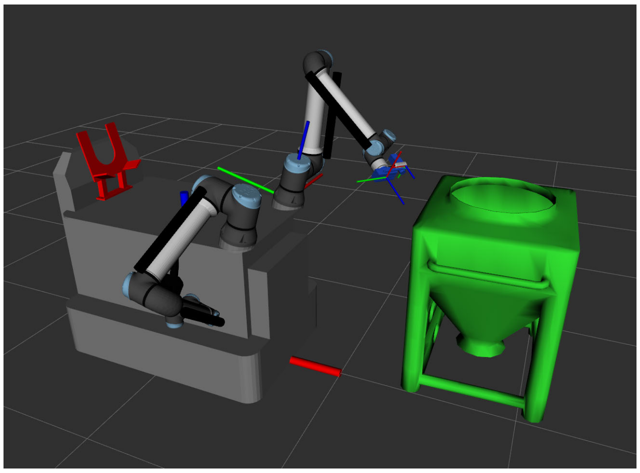


FIGURE 2. A pharmaceutical bin.

All computations reported in Sections IV and V were performed by a PC equipped with an Intel 11th-Gen i7-11800H Processor with 8 cores and 16GB RAM, and a NVIDIA GeForce RTX3060 graphic card with 6GB VRAM.


(a) Virtual twin of the robotic platform.

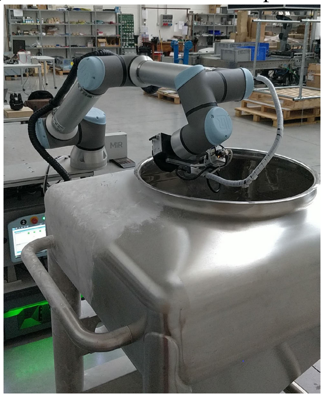

(b) Robotic platform scanning the inner surface of the bin.

FIGURE 3. Robotic platform used for simulation and experiments.

III. BIN IDENTIFICATION

A. METHODOLOGY

The main requirement for the bin-identification procedure is to be independent of the type of bin that is loaded inside the washing cabin. Two methods based on computer vision were evaluated: point cloud registration and pose estimation from a 2D image. The goal is to quickly define the exact shape of the bin at hand and its relative pose with respect to the inspection/cleaning robot.

Point cloud registration is the name used in literature to address the process of matching a polygon mesh[1] obtained from a 3D CAD model of an object with a measured point cloud representing the same object as seen from a 3D sensor such as a stereo camera or a laser scanner [7], [8], [9].

Pose estimation is generally the task of detecting the 6D pose of an object, which includes its location and orientation, from a single 2D image. Because a traditional 2D camera is essentially a bearing sensor, given an object in a framed scene, it is not possible from a single image to extrapolate its pose using only the bare data (i.e. the pixels) without additional external information or by resorting to complex and computational demanding neural-network techniques [10], [11], [12]. Feature matching and homography are common techniques that allow estimating the pose of a template object of known dimension in a cluttered scene where the object appears in an arbitrary pose.

The approach chosen in this paper belongs to the *pose-estimation* methodology and is based on *visual markers* (or *tags*), such as chessboards, that may be easily and distinctively recognized in an image. Markers have known size and dimensions and, once detected in the image, their pose can be immediately calculated with respect to the frame of the camera. Differently from point cloud registration and feature-based matching, subject to the effect of noise, light and clutter of the scene, visual-tag-based pose estimation usually yields millimetric accuracy at short ranges (i.e. $<$ 2 m) with little computational effort, low cost and very fast response. Moreover, markers can provide information not only about the pose but also about the object type, thus allowing its precise identification.

B. IMPLEMENTATION

As a first step, the mobile robot moves must move close to the bin, located in a fixed position in the work cell. The approach motion is subject to uncertainty due to drift and mapping errors and can lead to a positional error of a few centimetres. This inaccuracy is actually helpful to mimic a possible error in the placement of the bin inside the washing cabin (indeed, the latter operation can be performed by a human operator and small positioning errors are possible if a mechanical guide is not present).

A ChArUco board, namely a combination of a chessboard pattern and an ArUco board, is chosen as marker type and applied on one of the vertical sides of the bin. The marker is framed by the industrial 2D camera mounted at the end-effector of the robot, and its precise position is measured with millimetric accuracy ($<$ 2 mm positioning error [2]), thus allowing the identification of both the bin type and its location in space. This knowledge makes it possible to retrieve the bin CAD model from a remote database and spawn it in the planning scene, as shown in Fig. 3a.

IV. BIN SCANNING

A. METHODOLOGY

The algorithm used for this phase is inspired by the recent work from [13], which puts together the results from [14] about a non-voxel[2] approach for viewpoints generation, and from [15] for viewpoints sorting. Suggestions and insights from [16] are also used to obtain satisfactory results in a realistic pipeline.

The viewpoints generation is realized mainly using two open-source libraries for Python, which is the language used for these implementations: *Open3D* [17] and *PyMesh* [18].

This section describes the main steps that compose the algorithm as well as the introduction of new kinematic constraints related to the use of a robotic manipulator to reach the computed viewpoints, which is the main contribution to this stage. For greater generality, a more challenging scenario is considered with respect to that shown in Fig. 1: the location of the robotic manipulator can be arbitrary with respect to the bin, not necessarily centred with its upper hole, as shown in Fig. 3a, where the robot is mounted on an autonomous vehicle that can approach the bin from any side, but not from above.

1) VIEWPOINTS GENERATION

Given a CAD 3D model of the bin, the latter is loaded as a mesh with triangular faces and converted into a uniform point cloud. The number N of points can be arbitrarily defined as a percentage p_{SP} of the number of faces. The bin mesh reference system $OXYZ$ has its origin O in the centre of the circular upper hole of the bin and is assumed to have the Z-axis pointing upwards, as shown in Fig. 4.

Each point in the cloud, called *seed point*, is identified by the position vector $\mathbf{s}_i$ in $OXYZ$ and is associated with the closest triangular face of the original mesh, called *seed face*, whose normal direction is $\mathbf{n}_i$. Seed points and faces are used to compute each *ViewPoint (VP)* pose, namely the pose of the camera that will be used to perform the bin scanning. Each viewpoint is identified with a reference system $O_i x_i y_i z_i$, whose location with respect to $OXYZ$ is expressed in the form of a homogeneous transformation $\mathbf{H} \in \mathbb{R}^4$. Two different scenarios can be analysed, respectively for internal and external scanning.

[1]A *polygon mesh* in 3D computer graphics and solid modelling is a collection of vertices, edges and faces that compose a polyhedral object. The faces are usually triangles (triangle mesh), quadrilaterals (quads) or other simple convex polygons (n-gons).

[2]In 3D computer graphics, a *voxel* is a value on a regular grid in three-dimensional space.

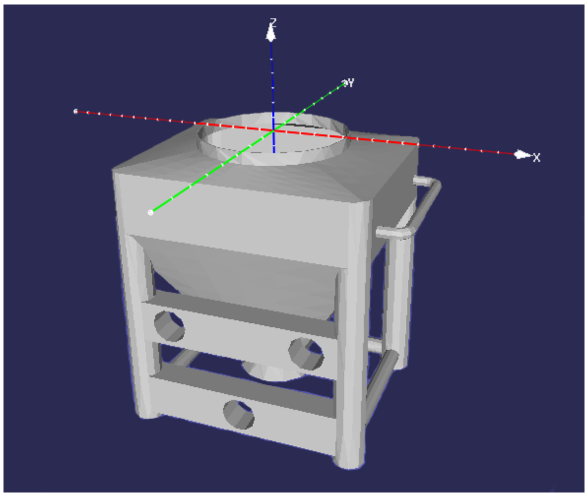


FIGURE 4. Bin mesh with the reference system *OXYZ*.

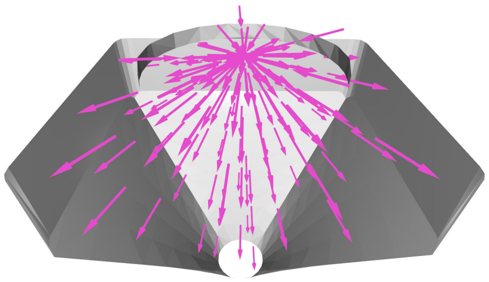


FIGURE 5. Internal viewpoints (132) irradiating from the bin mesh origin *O*.

In the case of internal scanning, all poses irradiate from a fixed point located at the origin O of the bin reference system (Fig. 5). Then, the direction $\mathbf{k}_i$ of each pose is oriented such that it points towards a different seed point (only the internal surface is used to generate the seed points in this case). In the case of external scanning, the z_i-axis of each pose is aligned with the opposite direction of the corresponding seed-face normal, namely $\mathbf{k}_i = -\mathbf{n}_i$, as shown in Fig. 6.

Once $\mathbf{k}_i$ is identified, the overall rotation matrix between $OXYZ$ and $O_ix_iy_iz_i$ is computed as [19]:

$$\mathbf{R}_i = \mathbf{I} + \lfloor \mathbf{Z} \times \mathbf{k}_i \rfloor + \frac{\lfloor \mathbf{Z} \times \mathbf{k}_i \rfloor^2}{1 + \mathbf{p}_i \cdot \mathbf{k}_i}, \tag{1}$$

where $\mathbf{Z}$ is the Z-direction of the mesh coordinate frame and $\lfloor \cdot \rfloor$ is the skew-symmetric matrix operator.

A suitable number D of viewpoints is associated with each orientation $\mathbf{R}_i$, all sharing the same z_i-axis and having their origins $\mathbf{t}_{i,j}$ located at constant distances $d_0, \ldots, d_{D-1}$ from the seed point, namely:

$$\mathbf{t}_{i,j} = \mathbf{s}_i - d_j \mathbf{k}_i, \quad i = 0, \ldots, N-1, \quad j = 0, \ldots, D-1. \tag{2}$$

The distances d_j must be selected within the range of valid field depths of the sensor and should appear in the list in order of preference because, in the end, only the first reachable viewpoint along each direction $\mathbf{k}_i$ will be kept.

Two examples of a viewpoint resulting from this process are shown in Fig. 7.

a: VIEWPOINTS FILTERING

Each viewpoint can then be kept or discarded according to any combination of the following conditions, listed in order of increasing computational effort:

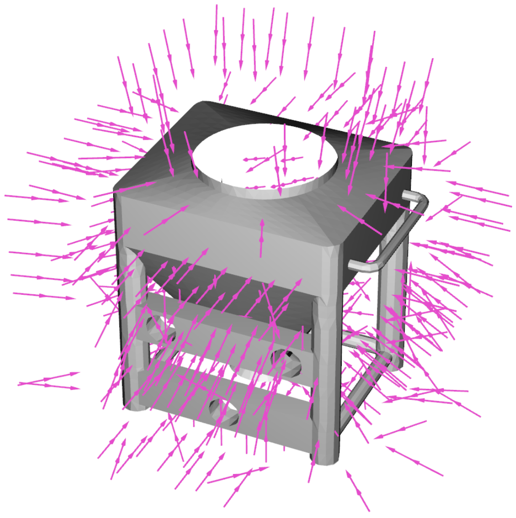
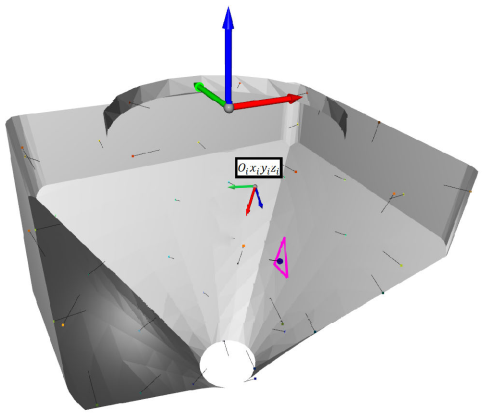


FIGURE 6. External viewpoints (406) aligned with seed-faces normals $\mathbf{n}_i$.

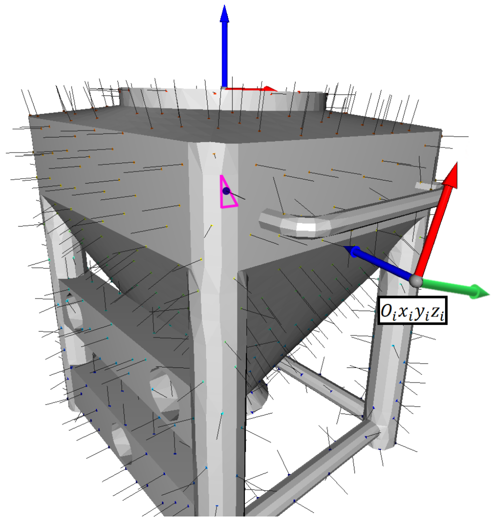
1) **height check**: it excludes all those viewpoints whose origin is below a certain threshold, resembling the impossibility of placing the camera underground or below a certain height;
2) **distance check**: it excludes all viewpoints whose origin is farther than a given distance from a convenient 3D point expressed in the coordinate frame of the bin; this helps, for instance, to discard all those points beyond the reach of a manipulator by placing the reference point at the origin of the robot base frame;
3) **occlusion check**: it excludes a viewpoint if a ray, starting at the viewpoint origin and ending at the centroid of the corresponding seed face, intersects any other triangle belonging to the mesh;
4) **collision check**: after loading the mesh of the camera and positioning it according to each viewpoint pose, the viewpoint is discarded if a collision between the camera mesh and the bin mesh is detected.

b: REACHABILITY CHECK

Once this initial filtering is complete, a simulation environment is used to test the actual feasibility of the remaining viewpoints. This is possible by spawning the bin and the robot in a realistic relative position and using a dynamic planner for online collision-free trajectory generation. After including the robotic platform and the collision objects (e.g. the bin), as shown in Fig. 3a, the virtual planning scene is kept updated using proprioceptive information, such as the manipulator joints angles, and external information, such as the location of



(a) Internal surface.

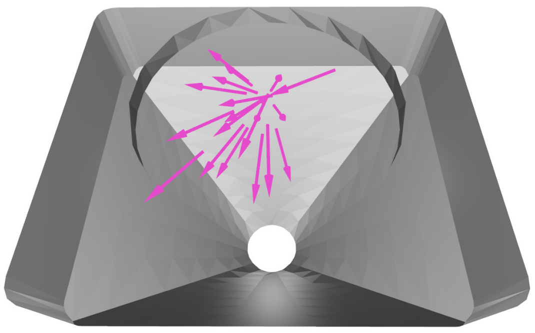


(b) External surface.

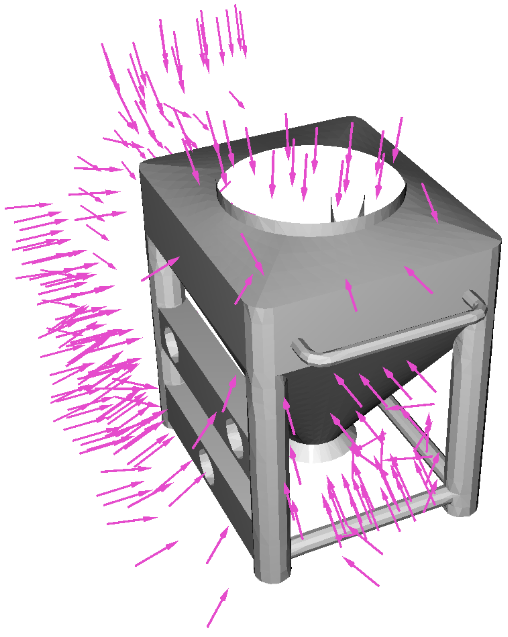
FIGURE 7. Bin mesh with normals and a viewpoint example. Coloured dots are the seed points obtained from the point cloud. The dark blue dot is the seed point used for the current viewpoint $O_ix_iy_iz_i$. The magenta triangle around the seed point is the corresponding seed face.

the bin with respect to the robot, obtained through computer vision techniques as described in Section III. Given a target in either joint or Cartesian space and an updated planning scene that incorporates the starting robot state, a collision-free trajectory is dynamically generated on the fly based on the *Open Motion Planning Library (OMPL)* [20] and, in particular, on the RTT* algorithm [21]. The procedure may be long, but it is a readily available solution to exclude unreachable poses considering both overall dimensions and the robot kinematic constraints. It is based on the simple evidence that if a pose is reachable, the planner will find a solution to the planning problem. If not, that pose is discarded.

An example of the application of the filtering procedures described in this subsection is shown in Fig. 8 (where the virtual twin of the robotic platform in Fig. 3a is used). The kinematic constraints and the interference problems of the robot with the bin play a major role in excluding the viewpoints facing one side of the internal surface (Fig. 8a). Likewise, the external surface of the bin that lies on the opposite side of the robot cannot be reached, and thus no viewpoints appear there (Fig. 8b). In general, for the same point-cloud-sampling resolution, more viewpoints remain for the external surface, due to a higher number of faces and the larger freedom of movement that the robot has compared to the narrow space inside the bin.



(a) Internal surface (20/132).



(b) External surface (218/406).

FIGURE 8. Resulting viewpoints after filtering and reachability check.

c: VISIBLE FACES

The last step of this phase consists of computing the visible faces from each remaining viewpoint. This is done by collecting all visible faces inside the camera frustum using ray casting techniques (the camera intrinsic parameters are used to define the cone of view according to the camera pin-hole model). A face is considered properly visible if its distance from the camera origin is within a predefined range expressing the depth of field of the sensor, and the glancing angle is above a threshold heuristically defined according to the characteristics of the sensor and the inspected surface (see an example in Fig. 9). This helps ignore faces that are too far or too close and, therefore, probably out of focus or too tilted to be properly observed.

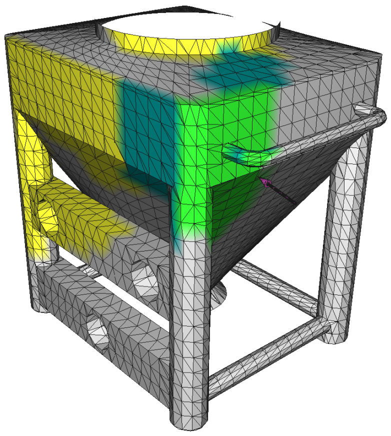


FIGURE 9. Bin mesh with visible faces from a viewpoint. The viewpoint is represented by the magenta arrow indicating the z_j-axis direction, and whose base coincides with the viewpoint origin. Green triangles are the valid visible faces. Cyan triangles are faces with an invalid glance angle. Yellow triangles are the faces that are outside the visible range. Grey triangles are the faces which are not visible from the viewpoint.

2) VIEWPOINTS SELECTION

The viewpoints selection is based on two optimization algorithms, called *Greedy Area (GA)* and *Simulated Annealing (SA)*, respectively, better discussed in [13].

Both methods rely on the existence of a *measurability matrix*, $\mathbf{C} \in \mathrm{R}^{m\times n}$, a two-dimensional binary array where the m rows correspond to all the faces of the mesh and the n columns to the viewpoints from the original set, i.e. the set of viewpoints that are left after the filtering and the reachability check. Each entry $c_{i,j}$ is equal to 1 if the i-th face is properly visible from viewpoint j; otherwise, it has a value of 0. This matrix can also be used to calculate the total visible area from all viewpoints and the visible area from every viewpoint.

In the GA method, the viewpoint that maximizes the objective function is selected at each iteration. The objective function, in this case, is the sum of the area of the visible faces from a single viewpoint. Therefore the selected viewpoint is the one from which it is possible to measure the largest surface, hence its name (Greedy Area). After the selection, the rows corresponding to the faces that were already measured by the latest best viewpoint are zeroed, so that those faces will play no role in the next iteration. The same routine continues until the measurability matrix is filled with only zeros. Consequently, this optimization process outputs the minimum number of viewpoints necessary to cover the entire surface that would be visible using the whole original set of viewpoints. An example of the application of this optimization is illustrated in Fig. 10.

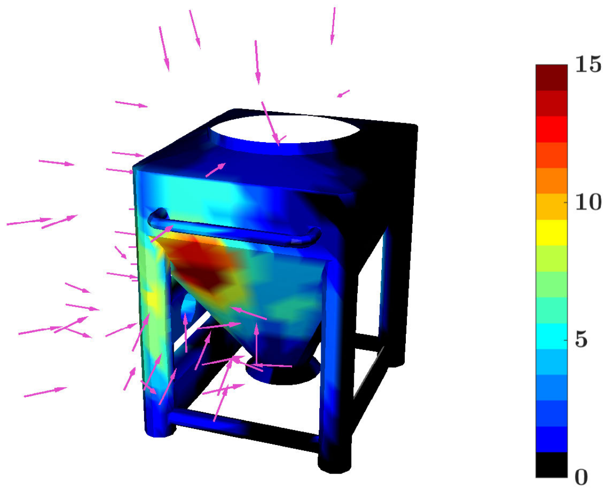


FIGURE 10. External viewpoints selected by the GA method (61/218). The colour map describes how many times a face was seen from a different viewpoint. The maximum value depends on the mesh and the settings.

Instead, the SA method, starting from the subset obtained by the GA method, aims to reduce it further by a desired percentage. Let A be the set of all the n viewpoints left after the filtering and the reachability check, and C the subset A obtained by the GA method, with dimension $n_{GA} \leq n$. B is a subset of C initialized as a desired percentage p_{SA} of randomly picked elements in C; the dimension of B is thus $n_{SA} = p_{SA}\, n_{GA}/100$.

At each iteration k, n_{swap} viewpoints are swapped between the set B^*, which is the current best combination of viewpoints found, and the set of unused viewpoints $\Delta = A \setminus B^*$, thus obtaining a new solution B_k, which is evaluated through a cost function that calculates the loss of visible area with respect to A:

$$Cost(B_k) = 1 - \frac{Area(B_k)}{Area(A)}, \tag{3}$$

where $Area(\cdot)$ is a function that computes the total amount of visible area using the viewpoints in the given set. Clearly, $Cost(B_k)$ is bounded between 0 and 1, since $B_k \subset A$. If the new set of viewpoints contained in B_k provides a larger visible surface (i.e. a lower cost) with respect to B^*, it replaces it, i.e. $B_k \mapsto B^*$. On the contrary, if the cost is higher, B_k is kept or discarded depending on the so-called MAC value [22]. The latter is given by:

$$0 < MAC = e^{\frac{\delta}{T_k}} < 1, \quad \delta = Cost(B_k) - Cost(B^*) > 0, \tag{4}$$

with T_k being the *temperature* parameter at iteration k, usually heuristically chosen or resulting from an optimal tuning session. The MAC value is then compared to a real number randomly sampled from a uniform distribution from 0 to 1, and the new solution B_k replaces the current one B^* if the random number is lower than the MAC value.

Both the values of n_{swap} and T follow an exponential decay $\gamma \in \mathbb{R}^+$ to allow a higher exploration at the beginning, namely

$$n_{swap,k} = n_{swap,0}\, e^{-\gamma k}, \quad T_k = T_0\, e^{-\gamma k}. \tag{5}$$

The algorithm stops after a number of iterations predefined by the user, unless a zero cost is obtained at any step. The SA method decreases the number of viewpoints at the expense of a very small loss of visible area (ideally 0%), but its efficacy strongly depends on the tuning parameters of the MAC number. An example of application of the SA optimization is illustrated in Fig. 11.

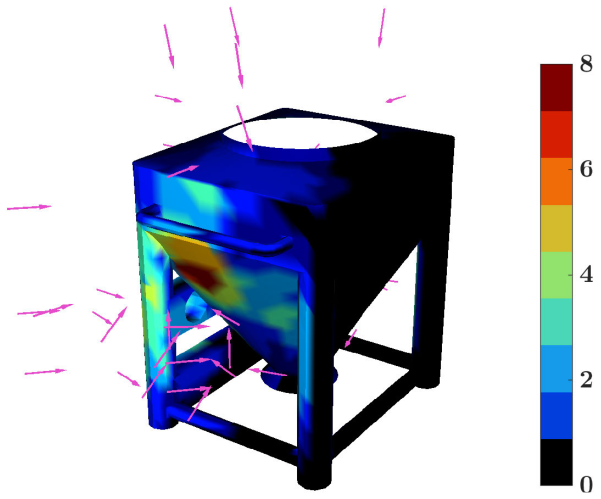


FIGURE 11. Viewpoints selected by SA method with 15% reduction (51/61). The colour map describes how many times a face was seen from a different viewpoint. The maximum value depends on the mesh and the settings.

3) VIEWPOINTS SORTING

Given the resulting subset of viewpoints, the last step consists of finding the best order in which they should be visited to minimize the overall path length. This problem is known in the literature as the *Traveling Salesman Problem*, and there are several solutions to it [23], [24]. The chosen one is the same as proposed by [13], that is, the *Ant Colony Optimization (ACO)* [15].

This optimization only requires a *distance matrix* as input which registers the cost of moving from viewpoint i (row-wise) to viewpoint j (column-wise). To populate this matrix the procedure is similar to the reachability test carried out resorting to the simulation environment. In fact, all combinations i-to-j and vice-versa are tested, and the length of the planned path covered by the end-effector in the Cartesian space is used as the cost and, thus, the entry value.

Given the distance matrix, the outcome of the ACO is the order of filtered and selected viewpoints that provides the shortest path passing through all viewpoints only once.

B. IMPLEMENTATION AND RESULTS

For a particular bin, the viewpoints can, in principle, be computed once and for all. However, the integration of robot kinematics and obstacle avoidance in the reachability of those configurations requires the knowledge of the relative pose between the robot and the bin. This pose is subject to changes since the location of the bin in the robotized cell is not always the same. As a consequence, a hybrid method was considered. When the robot finds the bin for the first time, it aborts the operation and requests the offline generation of the view poses and optimal path considering the bin in that specific location. The offline procedure may last several minutes (or even more than one hour) according to the user settings and the computing power, and it saves a model that can then be loaded and used online. At the next iteration, the process checks for the presence of this model, and if found, it compares the current measured relative pose of the bin with the one used in the offline simulation. Up to a certain tolerance, the viewpoints are adjusted according to the new bin location, whereas if the pose difference is too large, the process aborts and requires a new round of offline simulation. The reason for this latter behaviour is due to the fact that when the pose difference is large, the optimal path calculated offline may be sub-optimal since it was originally generated considering the bin obstacle in a different location with respect to the robotic arm base.

In short, in the offline mode, the viewpoints are generated, filtered and optimized, and the shortest path passing through the resulting ones is calculated. In the online mode, instead, the viewpoints are adjusted, and the robot moves to each view pose, possibly in the optimal order, to shoot a picture of the bin surface and, if desired, request the classification of the visible surface. The parameters involved in the two modes and the corresponding results are discussed in Sections IV-B1 and IV-B2.

1) OFFLINE MODE

The first viewpoints computation, which is independent of robot kinematics and dimensions, is based on a few user-defined parameters used in the steps described in Sect. IV-A1.

The point-cloud sub-sampling step takes a given percentage value p_{SP} to sample a subset of uniformly distributed seed points from the set of points obtained through direct mesh-to-point conversion: the larger the value, the more seed points (and therefore seed faces and viewpoints) will be considered. While including more viewpoints can improve surface coverage, it also significantly increases computational time, particularly during optimal path searches.

Increasing D (see Sect. IV-A1) also linearly increases the number of viewpoints to be tested, but it may be beneficial according to the application. In fact, recalling that the order of $d_0, \ldots, d_{D-1}$ is relevant, it may be convenient to prioritize closer or farther perspectives, but at the same time, those positions may entail a collision or generally be unfeasible.

For this reason, more options, i.e. $D > 1$, increase the chance of using each direction $\mathbf{k}_i$.

The filtering steps described in Sect. IV-A1a may apply up to four filters:

- *height check*: to implement the impossibility of the camera being underground, all viewpoints whose origin's z-component is below the mesh bottom-most coordinate minus the maximum dimension[3] of the camera mesh (if given) are removed;
- *distance check*: the reference point for the distance calculation is the origin of the robot (base) frame expressed in the bin coordinate system;
- *occlusion check*: this is particularly useful when dealing with concave objects;
- *collision check*: from a physical point of view, each viewpoint coincides with the optical frame of the camera; this filter may use the homogeneous transformation matrix that expresses the optical frame location with respect to the camera mesh frame; if this matrix is unknown, the camera mesh frame is considered coincident with the optical frame.

During the reachability check described in Sect. IV-A1b, the virtual robot tries to align the optical frame of the camera with each view pose. Since the camera orientation about the z-axis is irrelevant, the planner is free to reach that pose with any rotation around it. If the pose is reached with a different orientation than the ideal one, the view pose is updated.

The viewpoint selection process in Sect. IV-A2 begins with this new set of filtered and reachable view poses. The first two parameters involved at this stage are the Distance of View (DoV), which defines the visible range of the sensor in terms of depth, and the maximum glance angle ($\alpha_{g,\max}$), which defines the maximum inclination that a view ray may have with respect to a surface to see it properly. With this information, it is possible to compute the measurability matrix used by the GA method. With the parameter p_{SA}, the user may define the percentage of viewpoints that should be removed by the SA method from the viewpoints resulting from the previous GA method. If the value is set to 0, the SA optimization is skipped. From experience, when the viewpoints obtained with the GA method are an adequate number (e.g. ≥ 15), a reduction of 15% usually entails a loss in the visible surface of less than 1%, which is a reasonable trade-off.

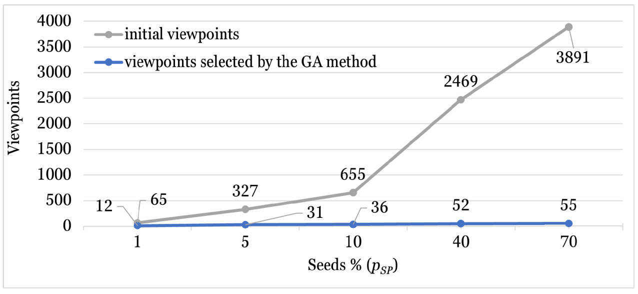
The last step discussed in Sect. IV-A3 consists in sorting the viewpoints according to the best path. This is the most time-consuming stage since it must test all motions from pose i to pose j, with $i = 0, \ldots, N_{opt} - 1$, and $j = 0, \ldots, N_{opt} - 1$, with N_{opt} being the number of viewpoints selected by the previous optimization steps.

An extensive simulation campaign was conducted in order to find the optimal parameter tuning, providing the best trade-off between computational time and accuracy.

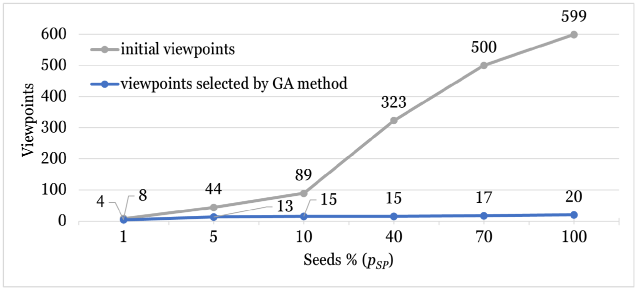
[3]The term *dimensions* refers to the sizes of the 3D object in the Cartesian space, often addressed as height, width, and depth.

Table 1 provides the parameters value used in a meaningful application of the approach. Table 2 shows the effect of viewpoint decimation due to filtering, reachability check and selection. It can be noticed that filtering plays a major role in viewpoint decimation, with an average reduction of -69%.

This is mainly due to the fact that a great portion of the external surface is out of reach when approaching the bin from one side, and the presence of the bin-supporting legs often causes a collision between the camera mesh and the bin mesh. Another interesting piece of information is related to the amount of viewpoints selected by the GA method. Noticeably, the actual number of viewpoints that are necessary to effectively see the same surface does not follow a linear trend, meaning that increasing the initial amount of viewpoints does not add much information (i.e. visible area) to the whole process, as shown in Fig. 12. On the contrary, Table 3 shows that computational time linearly grows as p_{SP} increases, with a major contribution of the reachability-check and the distance-matrix computation steps (see also Fig. 13). In conclusion, from experience, a seed percentage value p_{SP} between 10 and 40 is found to be a good trade-off between area coverage and computational time. Nonetheless, if the objective is to maximize the visible area disregarding the time and effort, all available seeds should be taken (i.e. $p_{SP} = 100\%$) as an initial attempt.



(a) External surface.



(b) Internal surface.

FIGURE 12. **Number of viewpoints against different values of the percentage p_{SP}.**

The last piece of data worth analysing is the amount of area loss when applying the SA method with a reduction percentage of 15%. The average loss is below 0.5% except for the case where $p = 1\%$, that is when the amount of starting viewpoints (obtained from GA method) is already small with respect to the total area size. In this case, even removing a

TABLE 1. Bin scanning parameters. The translation block in the homogeneous transformations is expressed in meters.

Parameter	Symbol	Value(s)
distance-of-view range	DoV	$[0.175 : 0.6]$ [m]
cobot max reach	$\bar{r}$	1.3 [m]
camera mesh frame to optical frame transformation	${}^{opt}\mathbf{H}_{cam}$	$\begin{bmatrix} 1 & 0 & 0 & 0.118 \\ 0 & 1 & 0 & 0.058 \\ 0 & 0 & 1 & -0.035 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
bin mesh frame to ChArUko marker frame transformation	${}^{tag}\mathbf{H}_{bin}$	$\begin{bmatrix} 0.9997 & -0.0233 & 0.0105 & 0.0277 \\ -0.0095 & 0.0391 & 0.9992 & 0.1491 \\ -0.0237 & -0.999 & 0.0388 & -0.3838 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
maximum glance angle	$\alpha_{g,\max}$	$60°$
SA reduction percentage	p_{SA}	15%
number of VPs for each seed pt.	D	2
distances of the VP origins from the seed pt. - ext	d_0, d_1	$[0.3, 0.4]$ [m]

TABLE 2. External viewpoints decimation with different seeds percentage values and SA reduction of 15%.

Seeds % (p_{SP})	VP	VP after filtering	VP after reachability check	GA VP	SA VP (% visible area loss)
1	65	22	13	12	$10(-1.76\%)$
5	327	105	68	31	$26(-0.19\%)$
10	655	195	137	36	$30(-0.1\%)$
40	2469	714	484	52	$44(-0.11\%)$
70	3891	1097	749	55	$46(-0.38\%)$

TABLE 3. Execution times with different seeds percentage values and SA reduction of 15% for the external-surface case.

Seeds % (p_{SP})	VP generation and filtering	Reachability check	VP selection	Distance matrix	Ant Colony Optimization
1	0.4s	33s	0.1s	54.3s	0.1s
5	2s	2m 34.4s	0.6s	6m 31s	0.2s
10	4.2s	4m 15.8s	0.3s	8m 41.6s	0.4s
40	13.8s	15m 56s	1.3s	19m 22.3s	0.8s
70	22.9s	23m 58.6s	2.6s	24m 48.1s	0.9s

single viewpoint can lead to a significant loss in the visible area.

2) ONLINE MODE

The online mode can be used for two different purposes: data collection for the classifier (see Sect. V) or surface-cleanliness evaluation.

In the first case, all reachable viewpoints are used, ignoring the selection and sorting steps in order to maximize the number of images that can be autonomously collected for the training of the classifier. To this end, a perturbation of the target pose may be added, if desired, to introduce some noise and help the training process by increasing the data variance.

In the evaluation case, only the selected and sorted viewpoints are used. In this scenario, no perturbations are applied. After stopping at a view pose, a request is sent to the classifier described in Sect. V, and the results are published. Before the request is sent, the procedure waits 1 second to let any arm oscillation disappear and focus the image. Image classification inference lasts roughly 1 second. The entire process typically lasts a few seconds per viewpoint, including the robot movement.

V. BIN-SURFACE CLASSIFICATION

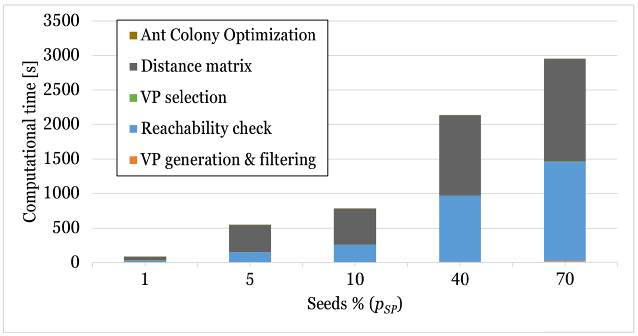
In machine learning, *classification* refers to a predictive modelling problem where a class label is guessed for given input data. There are four main types of classification:

- *binary classification*, predicting one of two classes;
- *multi-class classification*, predicting one of many classes;
- *multi-label classification*, predicting one or more classes for each sample;
- *imbalanced classification*, referring to classification tasks where the distribution of examples across the classes is not equal; this is typically the case for anomaly detection, which often belongs to uneven binary classification, given the usual low occurrences of the anomaly with respect to the nominal case in the training samples.

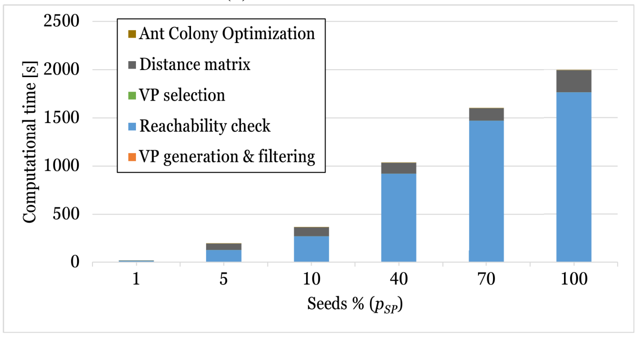
The use case is a binary-classification problem since the goal is to distinguish a clean surface from a dirty one, and training data are evenly distributed. The samples, in this case, are images of a portion of the bin surface taken at a suitable distance in the visible range of the RGB camera. Since the images may include some unrelated background, they are focused, cropped and resized to be independent of the original resolution.

A. METHODOLOGY

Machine Learning (ML) is a subset of *Artificial Intelligence* focusing on allowing computers to perform tasks without the need for explicit programming [25]. ML problems are, as a matter of fact, optimization problems where the solution is not given in an analytical form.



(a) External surface.



(b) Internal surface.

FIGURE 13. Computational times against different values of the percentage p_{SP}.

Deep Learning (DL) is a branch of ML that may target all problems suitable to ML, but it truly excels in those involving myriads of features such as images, speech and text processing [26]. DL modelling introduces a sophisticated approach based on complex, multi-layered *Neural Network*, built to allow data to move through nodes (like neurons) in highly connected ways.

A *Convolutional Neural Network (CNN)* is a particular type of *Deep Neural Network* employing *convolutions* in at least one layer [27], [28]. CNNs proved to be extremely suited for image processing thanks to their ability to assign importance to different features and/or objects in a very efficient way [29].

A CNN can capture both spatial and temporal dependencies in an image through the consecutive application of suitable filters. Differently from primitive methods, where filters are hand-engineered, CNNs are able to autonomously learn dependencies by training on labelled data, that is, in this case, images with an associated ground-truth description/label. This technique, therefore, falls under the realm of *data-driven supervised learning*, where the quality, distribution and availability of labelled data play a key role in the robustness and efficacy of this method.

1) RESNET-BASED APPLICATION

ResNet18[4] is chosen as the backbone for the bin surface binary classifier for its availability as open-source software and excellent performance. Thanks to the so-called *transfer learning* technique, it is, in fact, possible to exploit the pre-training of this sophisticated CNN over millions of image samples and use the small amount of application-specific pictures to replace the classifier section and finely tune the feature extractor layers. This method is based on the idea that, especially when dealing with images, a common processing part extracts high-level, generic features from a picture. These features can then be suitably combined to solve a specific task, for instance, a binary surface classification (e.g. clean vs dirty). The input of the customized ResNet-based network is an RGB image framing a portion of the bin surface, and the output is a label indicating whether the surface is clean or not. Another advantage of this method is the possibility to easily re-train the network whenever there are changes in the specifications, such as a different powder or a different bin surface are used. This characteristic can also be exploited to re-train the network with a larger and/or better data set.

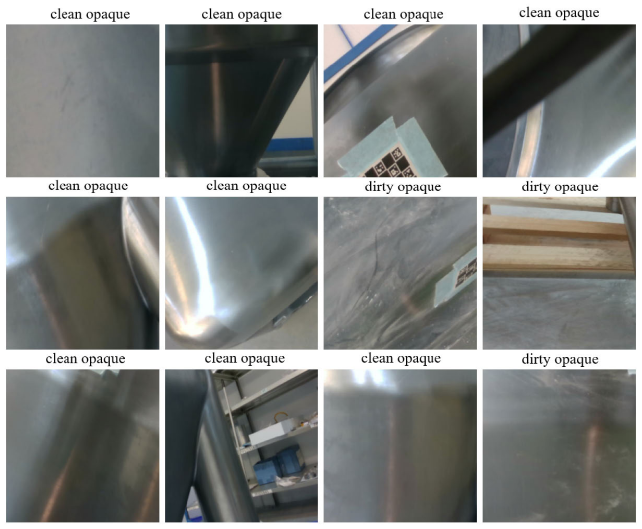
B. IMPLEMENTATION AND RESULTS

The software developed for this work is implemented resorting to the well-known *PyTorch*[5] [31] Python library, and includes two main items: a stand-alone script to arrange the data set and train the network, and a program that uses the trained model, connects it to a camera stream, and predicts the class of the latest grabbed image.

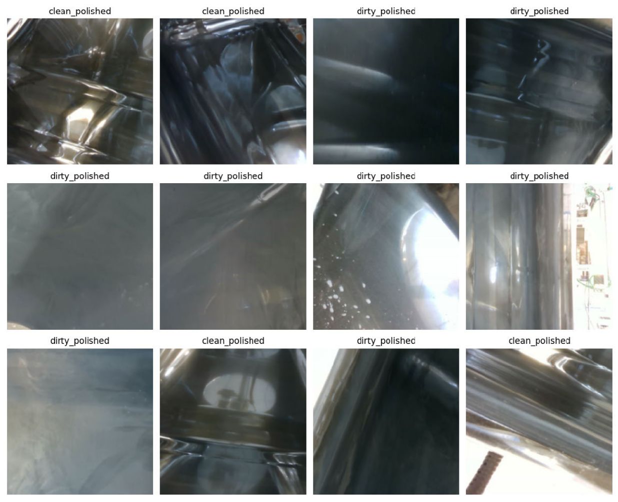
Fig. 14a shows a random subset of images obtained with the procedures explained in Section IV, with the corresponding ground truth (i.e. the correct label) used to train the network tailored to the external surface. Similarly, Fig. 14b shows a random subset used to train the network for the internal surface. Testing data were collected under typical conditions of an industrial warehouse with skylights, namely diffused artificial light mixed with natural light, a bin with matt external surfaces and glossy internal ones, and white powder randomly scattered on bin surfaces. Fig. 15 displays the distribution of training, validating and test sets for external and internal surfaces. Around 1000 samples per class were automatically collected for the external surface, whereas around 500 samples were collected for the internal surface.

[4]*ResNet* [30] won the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* in 2015 and was the first CNN to achieve predictions above the human level for object detection and image classification. Subsequent winners of the challenge are based on more complicated and power-consuming algorithms, but with a negligible gain in performance for the application at hand.

[5]*Pytorch* is an open-source ML framework that accelerates the path from research prototyping to production deployment by providing blocks, models, pre-trained parameters and functionalities related to ML and DL algorithms.

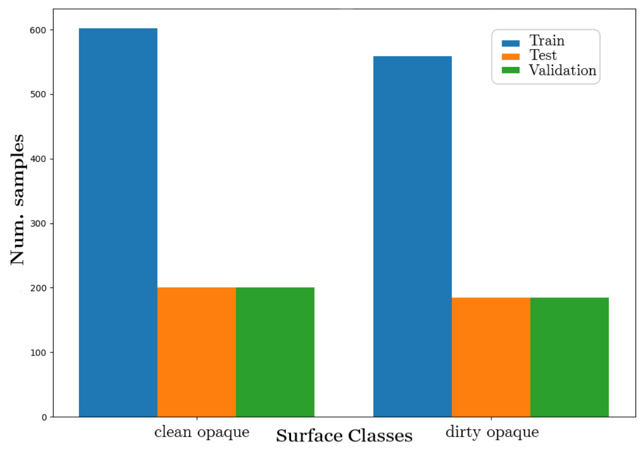


(a) External surface.

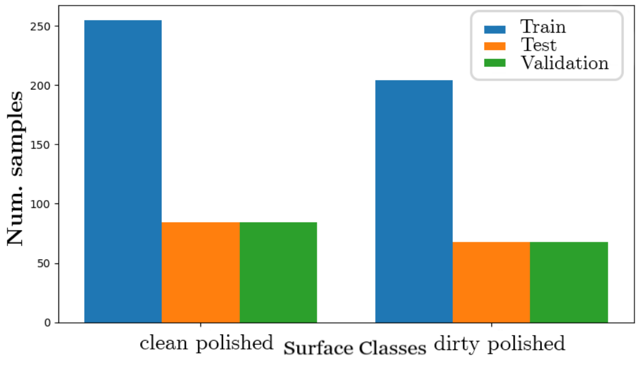


(b) Internal surface.

FIGURE 14. Random subset of sample training images for the surface classifier.



(a) External surface.



(b) Internal surface.

FIGURE 15. Data distribution.

This discrepancy is due to the greater difficulty for the robotic arm to access the internal side of the bin to take pictures during the data-collection phase. Moreover, the total area of the internal surface is smaller than the external one, leading to a smaller number of seed points and, thus, viewpoints. Fig. 15 shows that data sets are approximately evenly distributed, so that there is no significant class imbalance (i.e. more clean samples than dirty ones, or vice-versa).

The hyper-parameters listed in Table 4 were used to train the model in two consecutive steps: first, only the classifier layers are trained by freezing the parameters of the feature extractor; then, at the fine-tuning stage, the whole network is trained by unfreezing the feature-extractor parameters and by using a lower learning rate, but a larger number of epochs.

To augment training data, images are randomly cropped and/or horizontally flipped, whereas to enhance batch normalization, images are normalized using mean values and standard deviations for each one of the three colour channels, directly calculated on the image sets built for training and validation. Stochastic gradient descent is used as optimizer, with momentum equal to 0.9 and weight decay equal to 0.0001. A learning rate scheduler, instead, decays the learning rate of each parameter group by 0.1 every 5 epochs.

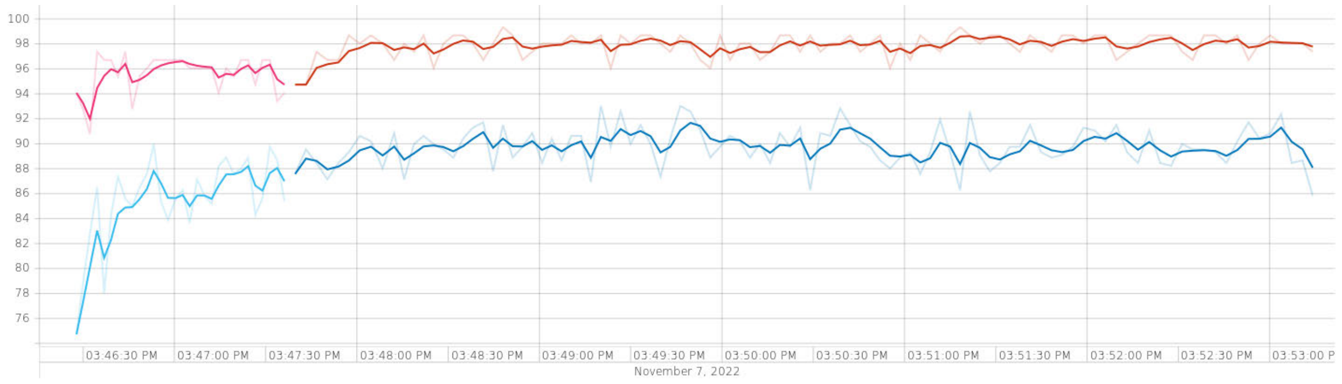
TABLE 4. Hyper-parameters used at training time.

Name	Value(s)
learning rate	0.001
num. of epochs	30
image size	244 $\times$ 224 [pixels]
batch size	32
momentum	0.9
weight decay	0.0001

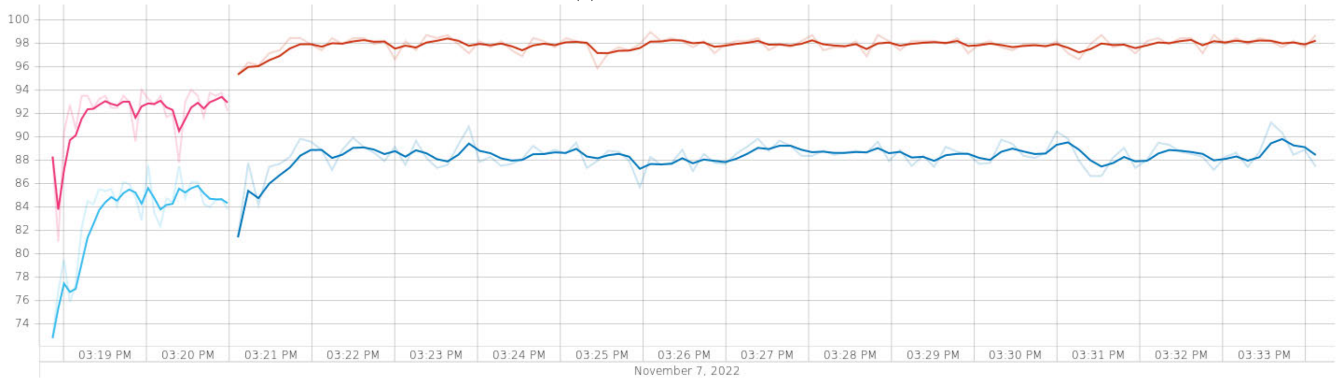
(a) Classifier.

Name	Value(s)
learning rate	0.0001
num. of epochs	100
image size	244 $\times$ 224 [pixels]
batch size	32
momentum	0.9
weight decay	0.0001

(b) Fine tuning.



(a) External surface.



(b) Internal surface.

FIGURE 16. Accuracy plot for bin cleanliness classifier. Magenta and light blue lines represent, respectively, the accuracy of training data and validation data during the classifier training. Orange and blue lines represent, respectively, the accuracy of training data and validation data during fine-tuning. The light lines in the back represent the raw data whose moving average is displayed by the darker overlapping lines.

Finally, the classifier accuracy during training and fine-tuning is shown in Fig. 16 for the training and validation data sets (Fig. 16a shows the accuracy for the external-surface classifier, Fig. 16b the accuracy for the internal-surface classifier). A typical training session lasts between 10 and 15 minutes, whereas image classification inference requires roughly 1 second.

Despite the relatively low amount of training data, results show accuracy on the test set around 98%. This excellent outcome, however, must be taken with care since the testing data were collected under conditions similar to those of the training set. This means that the combination of light conditions, type and status of the inspected surface and the camera used to collect testing data is not significantly different from the one used to collect training data. Hence, a larger variance in the working scenarios would be necessary to test the quality of the model, and this will be the subject of future works.

VI. CONCLUSION

This paper presented a strategy to fully automatize the procedure of cleanliness inspection of a pharmaceutical bin through a robotic arm and traditional/AI-based computer vision techniques.

An autonomous mobile robot was used to mimic the approach of a manipulator to a bin with arbitrary relative positioning. An eye-on-hand RGB camera was installed on the robot end-effector to carry out the classification of the surface status (e.g. clean vs dirty). This classification was performed by a convolutional neural network based on ResNet18 and transfer learning.

The viewpoints from which the images are taken are the result of an optimization that, starting from the CAD model of the bin, a virtual planning scene and a few parameters, minimizes the number of viewpoints while maximizing the visible area from the current position of the robot with respect to the bin. The results of this optimization can be used to set up a pipeline which is entirely bin-independent, assuming that the CAD model of each target bin is available. The same procedure may also be employed in the generation of the best path to clean the surface by only replacing the cone of vision of the camera with the frustum of action of a cleaning nozzle.

As far as the classifier is concerned, preliminary results are very promising, reaching an accuracy of 98% on conditioned data.

Two lines of further development can be pursued starting from the results discussed in this article. On the one hand, there are a few improvements to viewpoints generation, filtering and sorting that can be carried out, such as:

- the point cloud can be sub-sampled to concentrate more seed points in potentially critical areas of the bin, such as corners and junctions, where dirt tends to accumulate;
- exploiting the presented pipeline and working backwards, it is possible to optimize the position of the robot with respect to the bin to maximize the visible area by the sensor/washing nozzle;
- the Greedy Area method can be upgraded by including in its objective function not only the amount of visible area but also some indices related to its morphology[6]; in this way, it would be possible to prioritize those zones where dirt is likely to accumulate;

As far as the image classifier is concerned, instead, possible improvements are:

- the use of the validation set to tune the initial value of the learning rate, keeping the weight decay to 0;
- the use of the validation set to tune the value of the weight decay;
- the inclusion of dropouts to increase robustness;
- a different transformation train[7] for data augmentation;
- the use a different back-bone network than ResNet18;
- the use of ensemble techniques to create models of the same bin under different light conditions;
- the use of few-shot and unsupervised learning on unlabeled bin surface images, to mitigate the problem of limited labeled data;
- setting a threshold in the predictions to reduce the number of false positives (e.g. surfaces that are considered clean but are actually dirty) at the expense of increasing false negatives (e.g. surfaces that are considered dirty but are actually clean); in this way, a human-in-the-loop could be in charge of double-checking the detected dirty area and tell the machine whether the prediction is right or wrong; this human feedback may also be included in further training to increase robustness.

As one can see, the room for improvement is large, but with suitable changes, this work has the merit of becoming an effective and versatile industrial product.

ACKNOWLEDGMENT

The authors would like to thank the "Research and Innovation" and "Active" divisions of IMA S.p.A., for hosting the project, providing the necessary equipment and supporting its development.

[6]In the context of computer graphics and Computer-Aided Design (CAD), "mesh morphology" involves the examination and analysis of the geometric characteristics and structural properties of a mesh, such as its topology, geometry, connectivity and surface characteristics.

[7]A "transformation train" refers to a sequence of operations or transformations applied to original data to create variations or augmentations to improve the robustness and generalization capabilities of a machine-learning model.

REFERENCES

[1] G. Darragjati, "Studio di fattibilità di macchina per il lavaggio di contenitori farmaceutici (feasibility study of a cleaning machine for pharmaceutical bins)," M.S. thesis, Dept. Ind. Eng., Univ. Bologna, Bologna, Italy, 2011.

[2] S. Comari, R. Di Leva, M. Carricato, S. Badini, A. Carapia, G. Collepalumbo, A. Gentili, C. Mazzotti, K. Staglianò, and D. Rea, "Mobile cobots for autonomous raw-material feeding of automatic packaging machines," *J. Manuf. Syst.*, vol. 64, pp. 211–224, Jul. 2022, doi: 10.1016/j.jmsy.2022.06.007.

[3] A. Baldassarri, M. Bertelli, and M. Carricato, "Design of a reconfigurable mobile collaborative manipulator for industrial applications," *J. Comput. Nonlinear Dyn.*, vol. 18, no. 9, pp. 1–8, Sep. 2023, doi: 10.1115/1.4062595.

[4] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *Int. J. Robot. Res.*, vol. 37, no. 10, pp. 1205–1232, Sep. 2018, doi: 10.1177/0278364918765952.

[5] J. K. Behrens, K. Stepanova, and R. Babuska, "Simultaneous task allocation and motion scheduling for complex tasks executed by multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 11443–11449, doi: 10.1109/ICRA40945.2020.9197103.

[6] A. Pupa, S. Comari, M. Arrfou, G. Andreoni, A. Carapia, M. Carricato, and C. Secchi, "Enhancing performance in human–robot collaboration: A modular architecture for task scheduling and safe trajectory planning," *IEEE Trans. Automat. Sci. Eng.*, submitted for publication.

[7] L. Li, R. Wang, and X. Zhang, "A tutorial review on point cloud registrations: Principle, classification, comparison, and technology challenges," *Math. Problems Eng.*, vol. 2021, Jul. 2021, Art. no. 9953910, doi: 10.1155/2021/9953910.

[8] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Found. Trends Robot.*, vol. 4, no. 1, pp. 1–104, 2015, doi: 10.1561/2300000035.

[9] G. K. L. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 7, pp. 1199–1217, Jul. 2013, doi: 10.1109/TVCG.2012.310.

[10] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proc. Mach. Learn. Res., 2nd Conf. Robot Learn. (CoRL)*, vol. 87, Zurich, Switzerland, 2018, pp. 306–316.

[11] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science vol. 11210, Munich, Germany, Sep. 2018, pp. 712–729, doi: 10.1007/978-3-030-01231-1_43.

[12] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proc. 14th Robot., Sci. Syst. (RSS)*. Pittsburgh, PA, USA: Cornell Univ. Library, 2018, doi: 10.15607/RSS.2018.XIV.019.

[13] L. R. de Castro, "6 DoF tool path generator from CAD model for visual inspection of part surfaces," M.S. thesis, Eng. Fac., Univ. Porto, Porto, Portugal, 2022.

[14] G. H. Tarbox and S. N. Gottschlich, "Planning for complete sensor coverage in inspection," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 84–111, Jan. 1995, doi: 10.1006/cviu.1995.1007.

[15] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[16] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. Cambridge, MA, USA: MIT Press, Mar. 2019.

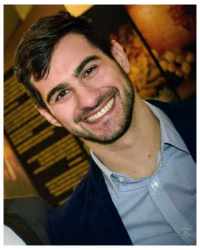
[17] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*.

[18] Q. Zhou. (2018). *PyMesh—Geometry Processing Library for Python*. [Online]. Available: https://pymesh.readthedocs.io/en/latest/

[19] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-level 3D object reconstruction via render-and-compare," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 3559–3568, doi: 10.1109/CVPR.2018.00375.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, doi: 10.1109/MRA.2012.2205651.

[21] S. Karaman and E. Frazzoli, ''Sampling-based algorithms for optimal motion planning,'' *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.
[22] W. K. Hastings, ''Monte Carlo sampling methods using Markov chains and their applications,'' *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970, doi: 10.1093/biomet/57.1.97.
[23] D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study* (Princeton Series in Applied Mathematics). Princeton, NJ, USA: Princeton Univ. Press, 2006. [Online]. Available: http://www.jstor.org/stable/j.ctt7s8xg
[24] C. Dahiya and S. Sangwan, ''Literature review on travelling salesman problem,'' *Int. J. Res.*, vol. 5, no. 16, pp. 1152–1155, 2018.
[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[26] M. D. Zeiler and R. Fergus, ''Visualizing and understanding convolutional networks,'' in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 8692, no. Part I, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Zurich, Switzerland: Springer, 2014, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.
[27] U. Michelucci, *Fundamentals of Convolutional Neural Networks*. Berkeley, CA, USA: Apress, 2019, ch. 3, pp. 79–123, 10.1007/978-1-4842-4976-5_3.
[28] S. Albawi, T. A. Mohammed, and S. Al-Zawi, ''Understanding of a convolutional neural network,'' in *Proc. Int. Conf. Eng. Technol. (ICET)*, Antalya, Turkey, Aug. 2017, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.
[29] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, ''A survey of convolutional neural networks: Analysis, applications, and prospects,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.
[30] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[31] A. Paszke et al., ''PyTorch: An imperative style, high-performance deep learning library,'' in *Proc. 33rd Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, in Advances in Neural Information Processing Systems, vol. 32. Vancouver, BC, Canada: Curran Associates, 2019, pp. 1–12. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html



SIMONE COMARI received the B.Sc. degree (Hons.) in automation engineering from the University of Bologna, the M.Sc. degree in robotics, systems, and control from the Institute of Technology, ETH Zürich, and the Ph.D. degree in mechanics and advanced engineering science from the University of Bologna. After a couple of years working abroad as a Software Developer with GoPro Switzerland AG, he returned to the academy focusing his research as a Ph.D. student on computer vision and AI applied to collaborative robotics. After a three-year-long affiliation with IMA S.p.A. Research and Innovation Department. He is currently a Consultant with DataSensing S.r.l. Research and Development Office.

MARCO CARRICATO (Senior Member, IEEE) received the M.Sc. degree (Hons.) in mechanical engineering and the Ph.D. degree in mechanics of machines from the University of Bologna, Bologna, Italy, in 1998 and 2002, respectively. He was a Visiting Researcher with the University of Florida, Gainesville, FL, USA; Laval University, Québec City, QC, Canada; the University of Guanajuato, Guanajuato, Mexico; Inria Sophia Antipolis, France; École Centrale de Nantes, France; and The Hong Kong University of Science and Technology, Hong Kong. He has been with the University of Bologna, since 2004, where he is currently a Full Professor and the Head of the Industrial Robotics, Mechatronics and Automation Lab @ Bologna (IRMA L@B). His research interests include robotic systems, servo-actuated automatic machinery, and the theory of mechanisms. He was awarded the AIMETA Junior Prize by Italian Association of Theoretical and Applied Mechanics for outstanding research results in the field of mechanics of machines.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement