2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

A CAD-Free Vision-Guided Framework for Robotic Deburring of Flexible Shoe Soles

Alessandra Tafuro[1], Angelo Mineo[2], Marco Guarini[2], Andrea Maria Zanchettin[2] and Paolo Rocco[2]

***Abstract*— Robotic sole deburring is a key, yet underexplored, challenge in footwear automation, where the deformable nature of rubber, variability of burrs, and diversity of sole geometries make automation difficult. Existing deburring approaches typically rely on CAD models or large training datasets, and often lack the ability to adapt online during execution. This paper presents a CAD-free, vision-guided framework for robotic deburring of shoe soles that integrates: (i) defect detection using the Segment Anything Model 2 without sole-specific training; (ii) motion planning for burr removal; and (iii) motion execution combining Forward Dynamics Compliance Control with online vision-based path tracking. The framework was validated on a UR5e robot equipped with a custom vacuum gripper. Results demonstrate a 95% success rate across soles of varying sizes, colors, and shapes. By eliminating CAD dependence, ensuring robust online correction, and maintaining compatibility with existing industrial deburring machines, this work provides a scalable step toward robotic finishing solutions in footwear manufacturing.**

I. INTRODUCTION

Recent advances in industrial automation have been driven by robotic systems capable of performing structured tasks with high precision and repeatability, with the global stock of industrial robots reaching 4.2 million units [1]. While manufacturing is moving from mass production to adaptive and intelligent systems that support small batches and product variants, complex finishing tasks like deburring (the removal of excess material from manufactured components) still require human dexterity. In shoe sole production, burrs frequently arise during outsole injection molding, with their removal typically performed by skilled operators relying on visual inspection and manual expertise (Fig. 1*a.*). Footwear factories must offer the flexibility to adapt to various designs, sizes, and volumes while keeping production fast. Although manual deburring adapts well to variations in burrs, it is time-consuming, inefficient in its use of human resources, repetitive, and offers low added value. The emergence of more intelligent robots allows to bridge the gap between human flexibility skills and high-volume production. In this, the integration of new-generation AI plays a pivotal role [2]. Future smart factories will heavily rely on automation, where robots replace manual tasks and information technologies ensure systematic, efficient, and responsive operations.

This paper elaborates on this transition, aiming to address the key challenges related to an automated robotic pipeline for sole deburring. Rubber's elasticity and deformation under pressure make automating deburring of soles challenging. The wide range of sole designs, sizes, and varying burr shapes further complicates standardization, requiring precise yet flexible systems that can adapt to different configurations. This paper addresses these challenges by presenting a CAD-free, vision-guided framework for robotic deburring of shoe soles. Unlike previous works that rely on CAD templates [3], extensive sole-specific training datasets [4], or purely offline planning [5], our system integrates defect detection, trajectory generation, and online compliant execution into a single pipeline. The main contributions of this paper are:
- *CAD-free defect detection*: A segmentation pipeline, eliminating the need for CAD templates or training data.
- *Hybrid control with online correction*: Integration of position/admittance control with vision-based algorithms for online trajectory correction on deformable workpieces.
- *Industrial relevance and validation*: Validation on a UR5e robot with a custom vacuum gripper, achieving a 95% success rate across multiple sole types, compatible with existing industrial deburring machines.

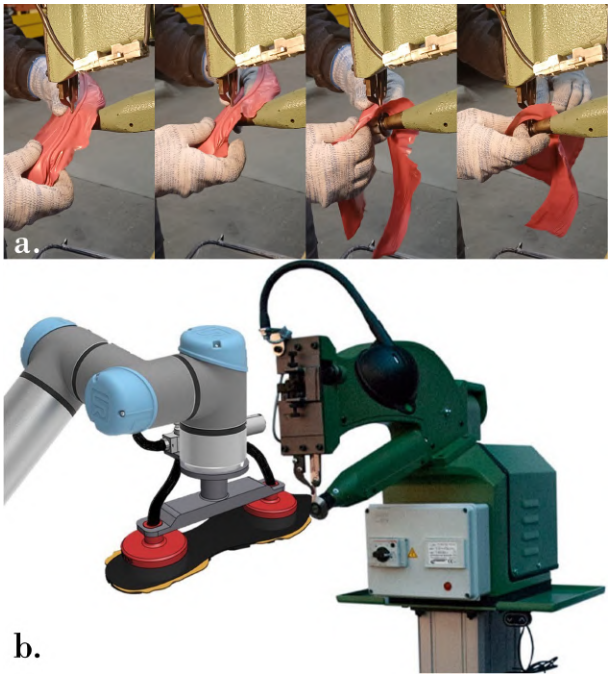

Fig. 1: *a.* Manual deburring of a shoe sole.
b. Proposed robotic solution for shoe sole deburring.

[1] Alessandra Tafuro is with Politecnico di Milano, 20133 Milano, Italy and Istituto Italiano di Tecnologia, 16163 Genova, Italy (alessandra.tafuro@polimi.it)

[2] Angelo Mineo, Marco Guarini, Andrea Maria Zanchettin and Paolo Rocco are with Politecnico di Milano, 20133 Milano, Italy (andreamaria.zanchettin@polimi.it, paolo.rocco@polimi.it)

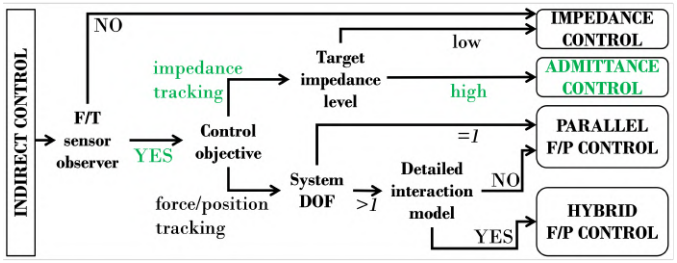

Fig. 2: Selection scheme for indirect force control approaches, re-elaborated from [10].

II. BACKGROUND KNOWLEDGE

A. Automation in footwear industry

The footwear sector has been slow to adopt automation, with most processes still performed manually. Recent work explored robotic roughing [6], polishing [7], and robotic grasping of soles [8]. Robotic deburring of shoe soles remained largely unexplored, with [4] and [9] being the only dedicated studies. [4] addressed defect detection and path planning using vision-based burr identification combined with Learning from Demonstrations. That work was however limited to offline path planning and did not incorporate online robot control. In this work, we introduce a complete framework that spans the entire process, from defect detection to path planning and robot control considering integration with industrial trimming machines. Two strategies are common in manual sole deburring: *a.* moving the tool against a fixed workpiece, or *b.* moving the workpiece against a fixed tool [11]. The choice between these setups depends on factors such as the size and geometry of the workpiece, the available deburring tools, the material properties of the workpiece, and the availability of a gripper capable of handling various objects. We developed a solution for the fixed-deburring tool configuration (*b.*), which involves workers using stationary machines (Fig. 1*b.*). By replacing manual handling with robotic manipulation of the soles, our system allows for the use of existing industrial deburring machines.

B. Defect detection and motion planning

Robotic deburring involves three key stages: *burr detection*, *motion planning* and *motion execution* [12].

Burr detection methods depend on the application, workpiece, and burr properties, and are generally classified as *contact* (using touch sensors for hard-to-see burrs) or *non-contact* (used when burrs are visible) [13]. Contact-based detection is impractical for rubber soles, while laser or thermal sensing [14] is costly, not suitable for rubber, and suffers from interference and diffraction. In contrast, vision systems provide an effective approach. CAD-based reconstruction [3] and contour matching [4] have been explored, but both require either CAD models or sole type-specific training. Our approach eliminates these dependencies, ensuring robustness across varied sole designs.

Regarding *path planning* for robotic deburring, there are three primary approaches: (a) CAD/CAM-based methods [12], (b) sensor or vision-based methods [15, 16] and (c) Learning from Demonstration approaches [4]. Our method plans the deburring path using vision-based defect detection results, eliminating the need for CAD models or human demonstrations. However, during execution, possible deviations are adaptively corrected by online path adjustment and admittance control.

C. Motion execution

Two key components of robotic deburring execution are force control and online path tracking and correction [12]. Force control can be passive or active. Passive control uses inherently compliant tools, but it offers limited flexibility and precision [17], while active control employs sensors and software to adjust a system's dynamic behavior and is typically classified as either indirect force control (admittance/impedance) or direct force control (hybrid or parallel force/position). Selecting a control strategy depends on factors such as force/torque sensing availability, impedance requirements, and environment modeling (Fig. 2). In shoe sole deburring, precise modeling is infeasible, making hybrid force/position control unsuitable. Impedance control can model Cartesian behavior, but its application is limited because most industrial robots are position-controlled and lack direct torque interfaces [18]. Admittance control is therefore preferred, exploiting internal position loops and available force feedback.

Forward Dynamics Compliance Control (FDCC) [18, 19] extends this approach by simulating robot dynamics; however, the limited stiffness of manipulators reduces machining precision, requiring online path tracking and correction [12]. To address this limitation, and in contrast to prior laser- or CAD-based approaches [5, 20], we developed an RGB-D vision module that monitors task state and adapts trajectories during task execution.

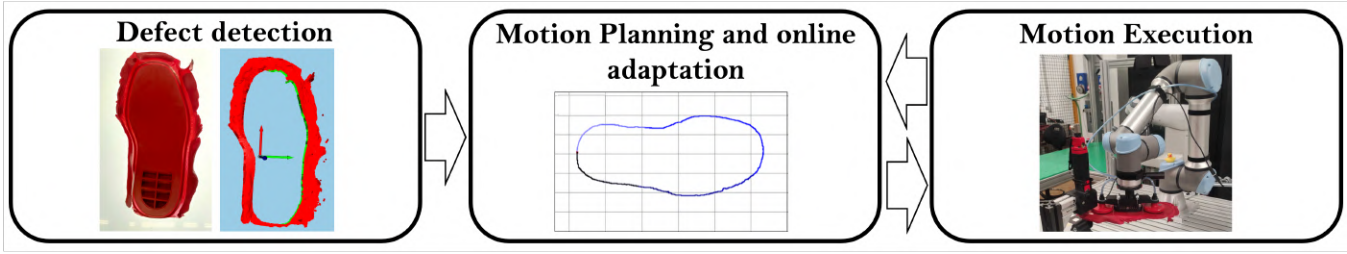

Fig. 3: Complete pipeline of our system: Defect detection, motion planning and motion execution phases.

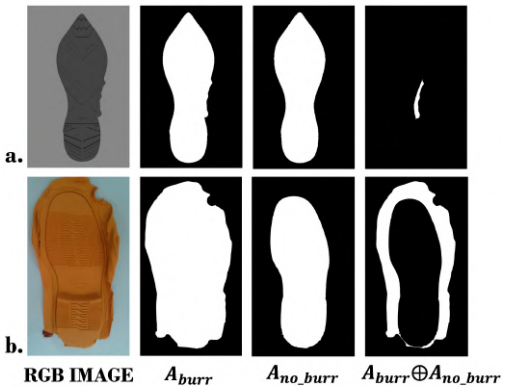

Fig. 4: SAM2-based defect detection in simulation (*a.*) and reality (*b.*).

III. PROPOSED APPROACH

As anticipated, our system (Fig. 3) solves three challenges:
1. *Defect Detection*: with image processing and deep learning, the burrs are identified on the soles.
2. *Path Planning*: based on the detected burrs and the geometry of the sole, the robot Cartesian trajectory is determined for the removal of the burrs.
3. *Motion Execution*: robotic deburring is performed with admittance control and vision-based path correction.

A. Defect Detection

This stage must determine the burrs' positions regardless of the sole type, color and size, as well as under different lighting conditions or sole placements within the workspace. Assuming that, in an industrial setting, soles with burrs arrive in a known pose relative to the robot's base frame, the workflow begins at time instant t_1, when the robot moves its end effector to a predefined image-capture position above the sole. The proposed approach removes the need for type-specific training while retaining high segmentation accuracy across varied sole configurations. We employed the Segment Anything Model 2 (SAM2) [21], developed by Meta AI, to segment both the nominal sole contour and the burr-affected contour. The process starts with converting the RGB image captured by the robot's end-effector camera (hereafter referred to as $camera_1$) into the CIELAB color space, followed by contrast enhancement using Contrast-Limited Adaptive Histogram Equalization (CLAHE). Then a dual segmentation step is applied to extract both the external contour of the sole (including the burrs) and the internal contour representing the nominal sole profile (Fig. 4). Since SAM2 can generate multiple masks for the same object, and the scene contains only the sole placed on the supporting plane, the model predicts both the burr-inclusive sole mask (largest non-background region) and the burr-free mask. The latter is generally well defined under suitable lighting, since in injection molding burrs arise mainly along the lateral edges of the mold, leaving the nominal profile visible. To guarantee correct contour selection, we compute the Dice similarity coefficient [22] between the burr-inclusive sole and each candidate mask, thus excluding unrelated regions. Once both contours have been finalized and stored as binary masks (obtaining A_{burr} and A_{no_burr}), the burr regions (A_b) are extracted through a pixel-wise XOR operation (1):

$$A_b = A_{burr} \oplus A_{no\_burr} \tag{1}$$

B. Path Planning

Once burrs are detected, the path-planning phase begins (Fig. 5). The cutting trajectory in pixel space is derived by intersecting the burr mask A_b with the nominal sole mask A_{no_burr}. This operation produces N_b binary masks A_j ($j = 1...N_b$) each representing the cutting trajectory to remove an individual burr in pixel space. If $N_b > 1$, the burrs are treated sequentially. For a given burr ($j = \hat{j}$), its binary contour $A_{\hat{j}}$ is transformed into an ordered set of trajectory points using the Coordinates Alignment for Mapping Image Lines (CAMIL) method [9]. This yields N_p deburring points $O'_{px} = \{(u_i, v_i)\}_{i=1}^{N_p}$ which define the cutting path in pixel coordinates. These points are then used as control points to interpolate a quadratic Non-Uniform Rational B-Spline (NURBS) curve with unit weights. The path points are then projected into the $camera_1$ reference frame obtaining

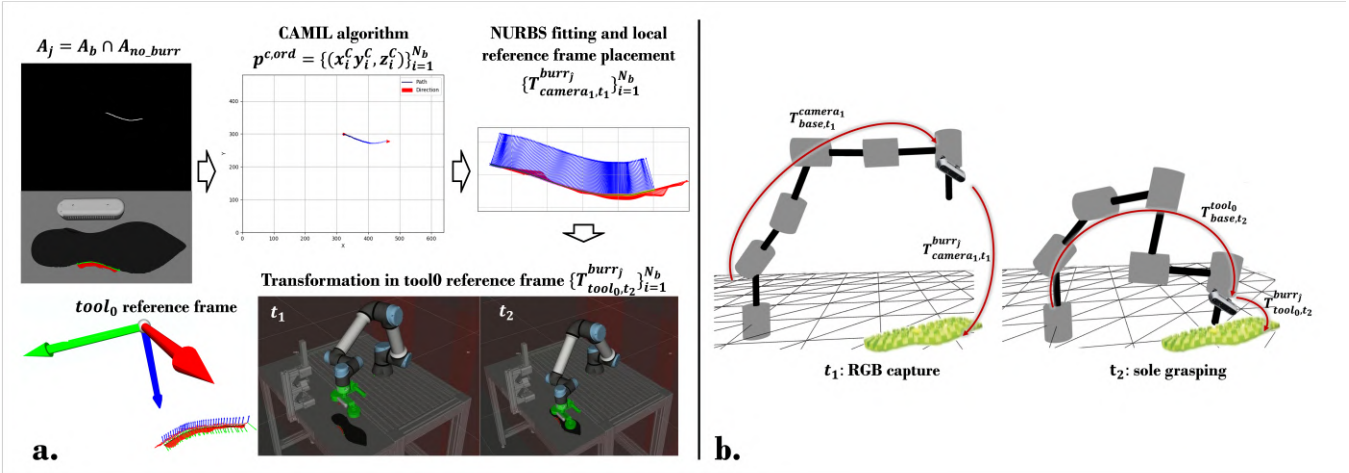

Fig. 5: *a.* Path planning: CAMIL algorithm converts pixels of A_j into an ordered set $p^{c,ord}$. A NURBS is fitted and local burr frames are defined. *b.* Robot and $camera_1$ configuration at $t = t_1$ (RBG image capture) and $t = t_2$ (sole grasping).

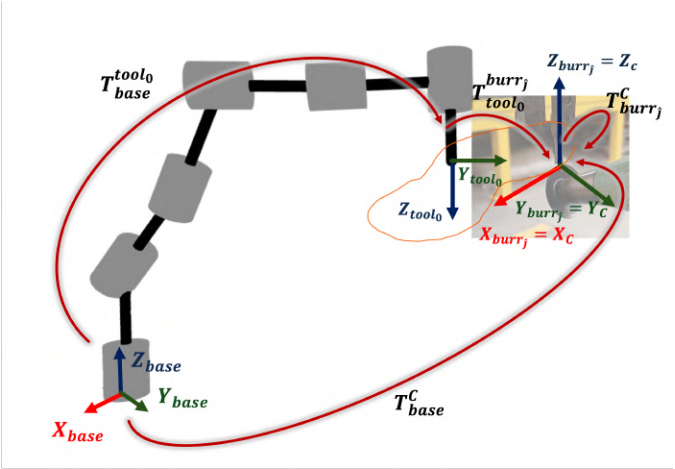

Fig. 6: Constraint imposition to compute the $tool_0$ motions for burr removal.

$p^c = \{(x_i^c, y_i^c, z_i^c)\}_{i=1}^{N_p}$ through the use of camera intrinsics and the known vertical offset z_c between the sole and the camera. This step assumes burrs are approximately planar on the sole's surface. This approximation, though not universal, holds in most soles manufacturing cases. Finally, the points are reordered into $p^{c,ord}$ to enforce a clockwise cutting direction when viewed from above the sole. For each point in $p^{c,ord}$, a local reference frame $[t,n,z]$ is defined, where t is the tangent to the NURBS curve, n is the normal unit vector pointing away from the sole's center, and z is constrained to be orthogonal to the sole's support plane (Fig. 5*a.*). The N_p frames associated with the $\hat{j}^{th}$ burr are represented as transformations from the camera frame. At $t = t_2$, the robot grasps the sole (Fig. 5*b.*). The burr frames are now expressed relative to the robot flange $tool_0$, $\{T^{burr_{\hat{j}}}_{tool_0,t_2}\}_{i=1...N_p}$, which allows computing the robot motions required to remove the burr. Indeed, the resulting end-effector poses must satisfy the application constraint represented in Fig. 6:

$$\{T^{tool_0}_{base} = T^{C}_{base} \cdot (T^{C}_{burr_{\hat{j}}})^{-1} \cdot (T^{burr_{\hat{j}}}_{tool_0,t_2})^{-1}\}_{i=1...N_p} \quad (2)$$

At this stage, $\{T^{tool_0}_{base}\}_{i=1...N_p}$ represents the offline-computed reference trajectory of the robot flange, denoted x_d^{offline}.

During execution, the reference poses are continuously validated and corrected online (Section III-C) prior to serving as the robot's motion commands.

C. Motion Execution

For this task, motion execution relies on the integration of force signals from dedicated sensors with online vision feedback. This multimodal strategy parallels the way humans perform deburring, combining visual perception for precise trajectory alignment with compliant interaction against the fixed deburring tool. By fusing these modalities, the system enables adaptive robot behavior. Our controller builds upon FDCC [18, 19]. FDCC imposes Cartesian compliance on position-controlled manipulators without requiring joint torque interfaces, by simulating the dynamic response of a virtual manipulator model to generate joint commands from desired motion and external force measurements. To extend this framework, we integrated an online vision module using an RGB-D camera (referred to as $camera_2$), complementing the flange-mounted $camera_1$. $Camera_2$ is fixed in the environment and oriented toward the cutting point, with a field of view covering both the deburring tool contact region and the approach trajectory of the sole (Fig. 7*b.*). This configuration enables online adaptation of the motion reference through continuous vision-driven updates of the robot's target pose, compensating for defect-detection or path-planning inaccuracies and adapting to workpiece deformations during execution. The precomputed offline reference trajectory x_d^{offline} is continuously validated during motion, and the updated reference, x_d^{online}, is streamed to the FDCC controller (Fig. 7*a.*).

The proposed framework comprises two interacting modules: the *Path Corrector* and the *Motion Executor* (Fig. 8). This modular architecture decouples vision-based perception and path adjustment (*Path Corrector*) from time-critical robot control (*Motion Executor*):

- The *Path Corrector* processes live RGB-D data to output a stream of validated or corrected reference path segments.
- The *Motion Executor* stores these segments and commands the low-level robot controller. It monitors the end-effector pose to verify task execution.

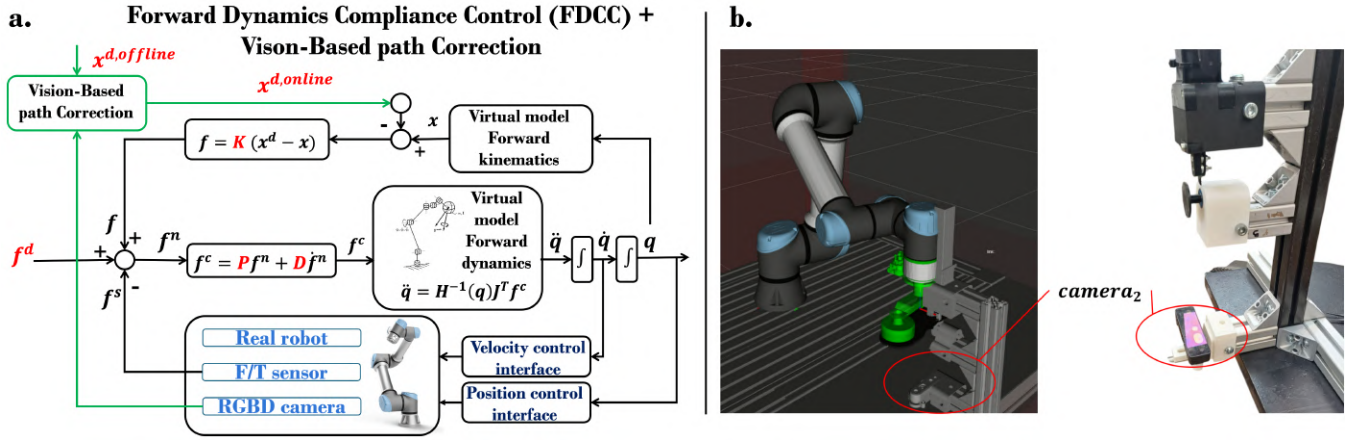

Fig. 7: *a.* Forward Dynamics Compliance Control + vision-based path correction. *b.* $Camera_2$ in simulation and reality.

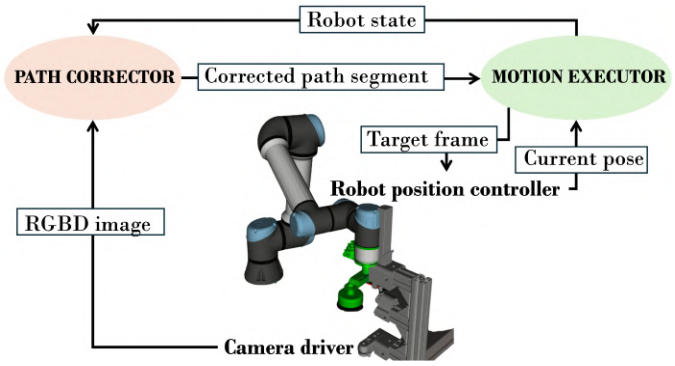

Fig. 8: The two core units of the framework (*Path Corrector* and *Motion Executor*) and the exchanged information.

Path correction is carried out in a *zone-by-zone* manner, dividing the overall trajectory into smaller segments for localized validation and correction. Before correction begins, the *Motion Executor* moves the robot to a predefined *approach pose*, a safe non-contact configuration that positions the workpiece within *camera*$_2$'s field of view. Once the robot is at the approach pose, the system partitions the camera's field of view $\mathscr{P} \subseteq \mathbb{R}^2$ into four vertical zones $\mathscr{Z} = \{\mathscr{Z}_{safe}, \mathscr{Z}_1, \mathscr{Z}_2, \mathscr{Z}_3\}$ (Fig. 9). They are defined as:

$$\mathscr{Z}_k = \begin{cases} \{(x,y) \in \mathscr{P} \mid 0 \le x < b_s\}, & k = safe \\ \{(x,y) \in \mathscr{P} \mid b_s \le x < b_1\}, & k = 1 \\ \{(x,y) \in \mathscr{P} \mid b_1 \le x < b_2\}, & k = 2 \\ \{(x,y) \in \mathscr{P} \mid b_2 \le x < w\}, & k = 3 \end{cases} \tag{3}$$

where b_s, b_1, b_2 are vertical boundaries and w is the image width. Each zone has a distinct role: $\mathscr{Z}_{safe}$ contains only pre-validated trajectory segments that are being passed to the robot controller; $\mathscr{Z}_1$ triggers the vision-based path correction mechanism when the offline-planned trajectory first enters; $\mathscr{Z}_2$ applies corrections as new portions of the offline trajectory enter and $\mathscr{Z}_3$ passively monitors the upcoming trajectory segment to be corrected. Path correction is triggered when the robot reaches the approach pose and the first offline-computed trajectory point ($i = 0$) enters $\mathscr{Z}_1$ (Fig. 9). At this stage, the initial corrected segment $\mathbf{S}_0$ is generated. Subsequently, each new segment $\mathbf{S}_{i+1}$ is produced by the *Path Corrector* (if necessary) once the end point the previous segment crosses the boundary b_1. Within $\mathscr{Z}_2$, path correction proceeds in three stages: *Sobel-based contour detection*, *segment validation*, and *fallback planning*. The outcome of these stages is a corrected path conforming to the burr profile as captured by *camera*$_2$, initially represented in pixel space. This path is mapped into the *tool*$_0$ reference frame via the projection $\Pi : \mathbb{R}^2_{\text{pix}} \to \mathbb{R}^3_{tool_0}$. Substituting the resulting transformation into (2) yields the required robot motions to follow the corrected burr contour. In parallel, the *Motion Executor* maintains a buffer of sequential poses from the corrected trajectory segments sent by the *Path Corrector* and coordinates with the robot's position controller to ensure each commanded pose is reached before advancing, guaranteeing smooth execution.

The three constituent elements of the path-correction mechanism operating within $\mathscr{Z}_2$ are delineated and discussed separately below.

1) Sobel-based contour detection: For each segment of the offline-computed deburring path lying within $\mathscr{Z}_2$, the algorithm first fits a smooth spline through the projected trajectory pixels (cyan curve in Fig. 10*a.*). This spline is then divided into localized Regions of Interest (ROIs). Within each ROI, the image is converted to black and white (Fig. 10 *b.*), contrast is enhanced, and Sobel edge detection is applied (Fig. 10 *c.*). The contour closest to the sole's center is then selected (green in Fig. 10*d.*). For each image column the pixel nearest to the sole body belonging to the green contour is extracted, and these pixels collectively define the corrected trajectory, yielding the refined deburring path in image space.

2) Segment Validation: Each newly generated segment (with n points) is then validated in 2D with a *Multi-Criteria Geometric Validation Filter* (Fig. 11), which applies a length and a directional check. The effective segment length, defined as the sum of the Euclidean distances between consecutive points along the segment, according to 4,

$$L_{\text{eff}} = \sum_{i=1}^{n-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2 \tag{4}$$

must satisfy the inequality $L_{\text{direct}} < L_{\text{eff}} \le \lambda_T \, D_{\text{man}}$.
L_{direct} denotes the Euclidean distance between the first and last points of the segment ($L_{\text{direct}} = \|\mathbf{p}_n - \mathbf{p}_1\|_2$) and D_{man} denotes the Manhattan distance between the same points ($D_{\text{man}} = |x_{pn} - y_{p1}| + |y_{pn} - y_{p1}|$). This inequality filters out segments that are either too short or fragmented, and it also detects paths that are excessively tortuous or contain loops. Smoothness and directional regularity of the path is instead assessed using the Mean Resultant Length (5):

$$R = \tfrac{1}{n-1} \sqrt{\Big(\sum_{i=1}^{n-1} \cos\delta_i\Big)^2 + \Big(\sum_{i=1}^{n-1} \sin\delta_i\Big)^2} \tag{5}$$

where δ_i is the angular deviation of the tangents $\mathbf{e}_i$ in every trajectory point from the global direction expressed as $v_{\text{global}} = \mathbf{p}_n - \mathbf{p}_1$. A segment is accepted only if both

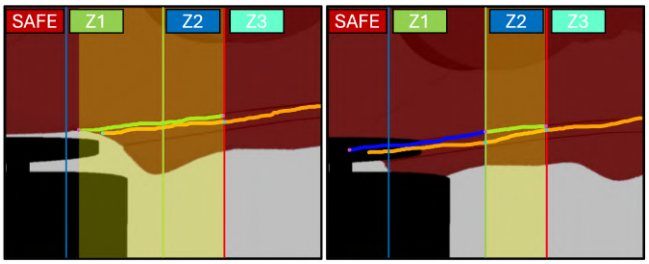

Fig. 9: RGB image from *camera*$_2$ (the sole in red, the background in gray, and the deburring tool in black). *Left*: Entry of the first point of the trajectory (orange) inside $\mathscr{Z}_1$ triggers the generation of the first corrected segment (green). *Right*: When the current segment endpoint reaches the left boundary of $\mathscr{Z}_2$, the next corrected segment is generated (green).

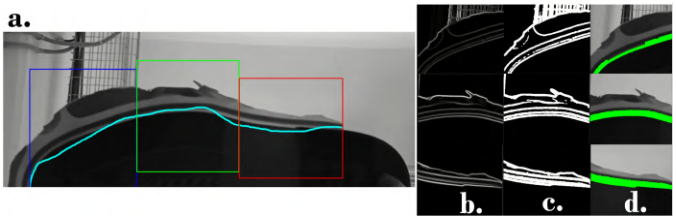

Fig. 10: Sobel-based contour detection.

conditions hold: $L_{\text{direct}} < L_{\text{eff}} \leq \lambda_T D_{\text{man}}$ and $R > R_{\text{min}}$. R_{min} and λ_T are two hyperparameters to be tuned. With this filter, when trajectory segments exhibit geometric irregularities such as jagged edges, discontinuities, or sharp turns, they are rejected and recomputed using the fallback planner.

3) Fallback Planning: The Adaptive A* Fallback Planner (Fig. 12) is activated when a trajectory segment generated is rejected by the Multi-Criteria Validation Filter previously described. Its role is to compute an optimal replacement path between the start and end points of the invalid segment within a discrete search space, subject to multiple weighted constraints. The planner extends the classical A* algorithm [23] by incorporating domain-specific penalties into the cost function, thereby ensuring consistency with the workpiece geometry. The planner operates on a 2D discrete grid corresponding to the image plane and takes as input two sets of pixels: the Ideal Path Set ($\mathscr{P}_{ideal} = \{n_i\}_{i=0\ldots m}$), representing the original but rejected trajectory segment, and the Contour Set ($\mathscr{P}_{contour}$), representing the detected workpiece boundary as extracted by Sobel edge detection. As a preprocessing step, both sets are dilated, introducing a tolerance margin that reduces sensitivity to pixel-level discontinuities. The planner then seeks an optimal path (in pixel space) from a start node $n_0 \in \mathscr{P}_{ideal}$ to a goal node $n_{\text{m}} \in \mathscr{P}_{ideal}$, minimizing a task-specific cost function. For each new node n_i in the path from n_0 to n_{m}, the cost is defined as per 6,

$$f'(n_i) = g(n_i) + h(n_i) + P_{\text{contour}} + P_{\text{ideal}} + P_{\text{smooth}} \quad (6)$$

where $g(n_j)$ is the cumulative path cost, and $h(n_i) = \|n_{\text{m}} - n_i\|_2$ is the Euclidean heuristic. Additionally, three penalty terms enforce task-specific constraints (7).

$$\begin{aligned} P_{\text{contour}} &= w_{\text{contour}} \cdot \mathbb{I}(n_i \notin \mathscr{P}_{\text{contour}}) \\ P_{\text{ideal}} &= w_{\text{ideal}} \cdot \mathbb{I}(n_i \notin \mathscr{P}_{\text{ideal}}) \\ P_{\text{smooth}} &= K_{\text{curvature}} \cdot |\theta_2 - \theta_1|^{\alpha_{\text{curvature}}} \end{aligned} \quad (7)$$

P_{contour} restricts deviations from the previously identified contour, P_{ideal} discourages excessive divergence from the original discarded segment, and P_{smooth} penalizes sharp tuns to promote smooth trajectories. w_{contour}, w_{ideal} and $K_{\text{curvature}}$

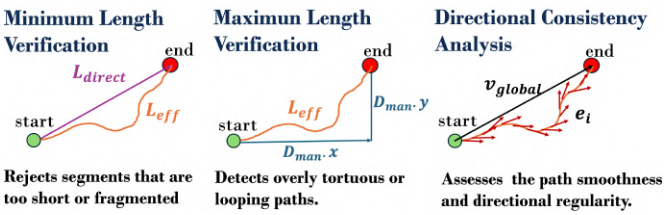

Fig. 11: Multi-Criteria Geometric Validation Filter.

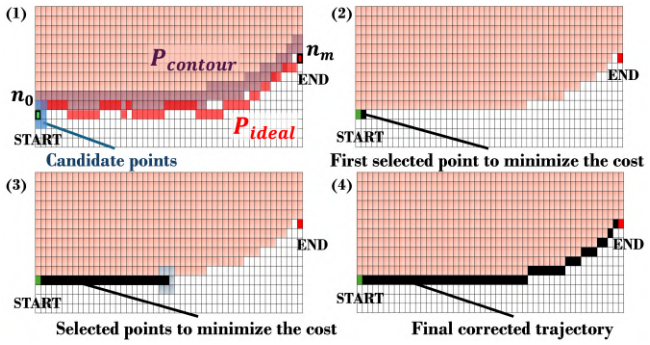

Fig. 12: Adaptive A* Fallback Planner.

are tunable weighting factors while $\alpha_{\text{curvature}} \geq 1$ is an exponent to more heavily penalize high-curvature sections. The angles θ_1 and θ_2 are defined at the two most recent nodes, each quantifying the directional change between consecutive segments determined by the last three nodes.

IV. EXPERIMENTAL VALIDATION

The proposed framework was validated first with the Gazebo simulator and then with a real robotic platform, using ROS2 for implementation. The experimental setup (Fig. 13*a*) consisted of a Universal Robots UR5e equipped with a custom dual-suction vacuum gripper and two cameras: an Intel RealSense D435i ($camera_1$, end-effector mounted) for defect detection and an OAK-D Pro ($camera_2$, fixed, capturing the cutting point) for online correction. Shoe soles of varying sizes, shapes, colors, and rubber hardness (measured in Shore A) were tested to evaluate robustness. A custom tool was designed to replicate the geometry and vibrations of industrial trimming machines. For safety regulations in

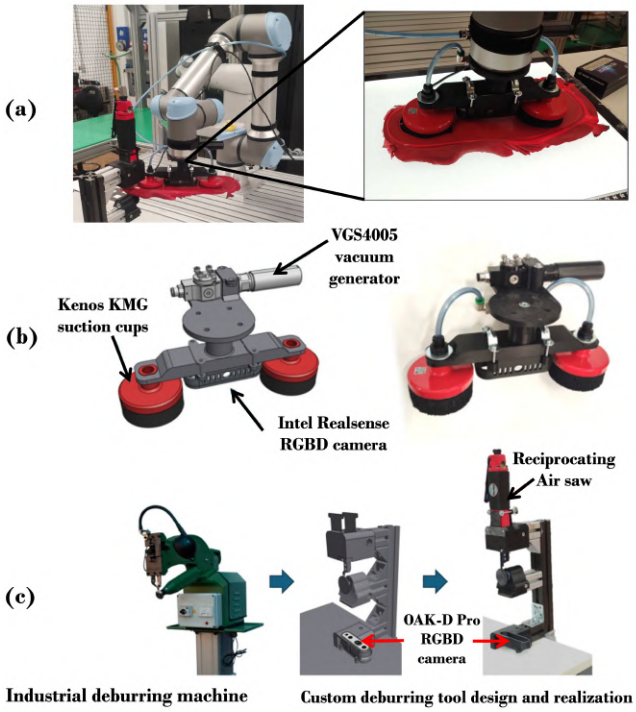

Fig. 13: Experimental setup.

the laboratory, no actual cutting was performed; instead, the path following accuracy was assessed defining a contact point on the deburring tool as the cutting point. Since the cutting blade was not mounted on the tool, execution was carried out without the use of force signals. Instead, the robot operated under Cartesian position control, sufficient to validate that corrected paths remained aligned with the burr geometry. The vision-based path correction controller, however, is designed for direct integration into an admittance control loop, enabling the combined use of force and visual feedback in real cutting.

A. Experimental Setup

1) Custom vacuum gripper design: A custom vacuum gripper was developed to ensure reliable manipulation of rubber soles, where suction cups were selected for their ability to adapt to irregular and compliant surfaces. The required gripping forces were derived following [24, 25], using the dynamic specifications of the UR5e robot. With a maximum angular velocity of $180/s$ and a reach of 0.85 m, the end-effector trajectory was modeled as a half-circular path of length L = 0.85π = 2.670 m. Assuming uniform circular motion, the end-effector acceleration was computed:

$$a = \frac{v_f^2 - v_i^2}{2L} = 1.335 m/s^2 \tag{8}$$

for $v_f = 2.670\ m/s$ and $v_i = 0\ m/s$. The theoretical horizontal and vertical gripping forces were then obtained as:

$$F_{th,v} = m(g+a)s = 10.03N, \quad F_{th,h} = \frac{F_{th,v}}{\mu} = 16.72N \tag{9}$$

where m is the mass of the sole, g the gravitational constant, $\mu = 0.6$ the friction coefficient, and $s = 3$ is the safety factor. Following the force analysis, two Piab Kenos® KMG suction cups (70 mm diameter, foam interface) were selected, providing a suction force above the required threshold. A custom end effector was then designed and 3D-printed to integrate the cups with a Piab VGS4005 vacuum generator. The final prototype (Fig. 13*b.*) features two aligned suction cups at a fixed spacing, enabling robust manipulation of soles with varying size and geometry.

2) Custom deburring tool design: A custom deburring tool was developed to reproduce the industrial trimming process in a laboratory setting, emulating the manual machines in [26]. Industrial trimmers typically comprise an adjustable support, a supporting wheel, and a reciprocating blade operating at $\sim 5{,}000$ cycles/min. The prototype (Fig.13*c.*) replicated the support and wheel via 3D printing, while the cutting mechanism was simulated with a reciprocating air saw (Clarke Air CAT32B 3110426) operating up to $10{,}000$ strokes/min, thus matching industrial performance. In compliance with laboratory safety regulations, the blade was excluded, and only the saw was employed to replicate the characteristic vibrations.

B. Testing results

We compared two experimental conditions to highlight the contribution of the proposed online vision-based correction:

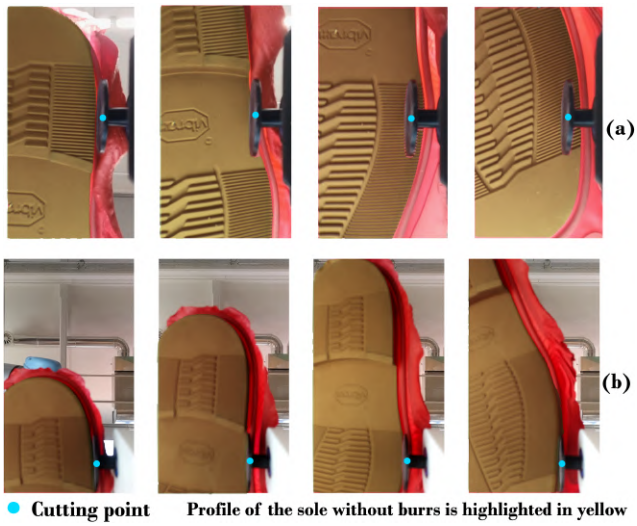

Fig. 14: Snapshots with (*b*) and without (*a*) online correction, deburring from the heel (left) to the toe (right); without correction, the cutting point enters the sole body.

a. Offline Baseline: The robot executed the trajectory computed offline after the defect detection and path planning stage (Sec. III-B). No online correction was applied, meaning deviations from $x^{d,offline}$ were not compensated;
b. Online Correction (Proposed framework): The robot executed a continuously validated and corrected path $x^{d,online}$, activating the vision-based correction module (Sec III-C). This allowed the system to adapt online to inaccuracies in segmentation, path planning, or deformations of the sole.

Performance was assessed according to three criteria:
1. Path accuracy: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) between executed robot trajectories and manually annotated ground-truth contours on high-resolution images of the grasped soles.
2. Task success rate: The percentage of trials in which the tool path remained aligned with the burr contour without entering the sole body.
3. Computational feasibility: Average execution time of the main modules to ensure real-time operation.

Each scenario was tested over $N = 10$ trials with 4 different soles. Results are shown in Table I. Offline-only execution suffered from large errors (MAE 22.6 px, RMSE 27.8 px) and low success (25%), as deviations quickly caused tool penetration into the sole. By contrast, the Online Correction approach reduced both error metrics by more than 80% (MAE = 4.0 px, RMSE = 4.6 px). Importantly, the success rate rose to 95%, demonstrating that the online correction mechanism effectively prevented tool intrusion into the sole and maintained alignment with the burr contour. These improvements are visually illustrated in Fig. 14, where the offline-only trajectory drifts into the sole body, while the online-corrected trajectory consistently follows the burr edge. Robustness was confirmed across different lighting conditions and soles types (in terms of pattern, hardness and color). The SAM2-based segmentation module extracted contours reliably without retraining, confirming the CAD-

TABLE I: Quantitative Comparison of Offline baseline and Online Correction Approaches (N = 20 trials).

Method	MAE [pixels]	RMSE [pixels]	Success Rate
Offline Baseline	22.6 ± 3.0	27.8 ± 2.9	25%
Online Correction	**4.0 ± 1.0**	**4.6 ± 1.2**	**95%**

TABLE II: Average Execution Time of Framework Components.

Stage	Component	Avg Time [ms]
Offline	SAM-2-based defect detection and motion planning.	43400 ± 6800
Online (Per Segment)	Multi-Criteria Validation Filter	12.6 ± 2.9
	Adaptive A* Fallback Planner (when invoked)	193.2 ± 26.7

free and dataset-independent nature of the framework. From a computational standpoint (Table II), the online correction modules operated well within real-time constraints. The Multi-Criteria Validation Filter required on average only 12 ms per trajectory segment, while the Adaptive A* fallback planner, when invoked, averaged 193 ms—still faster than the physical robot motion. Since the OAK-D Pro operated at 60 Hz, while the path-correction modules processed trajectory segments at an average rate of approximately 80 Hz (12 ms per segment) with the Adaptive A fallback planner running at about 5 Hz when invoked. The corrected trajectory was then streamed to the UR5e through its external position-control interface running at 125 Hz, with the robot's internal joint-servo loop operating at approximately 500 Hz. This frequency coordination ensured that perception, correction, and execution remained synchronized and within real-time constraints. Defect detection and path planning timings are reported for completeness; while reasonable, they should be assessed against human cycle times within the orchestration of tasks in a real industrial setting. These experiments demonstrate that the proposed system offers a solution for robotic sole deburring, a task currently carried out by human operators. The substantial improvements in accuracy and success rate confirm that online correction is foundamental to handle the uncertainties inherent in flexible sole processing.

V. Conclusion

This work introduced a comprehensive CAD-free framework for robotic deburring of flexible shoe soles, integrating defect detection, path planning, and compliant motion execution with online vision feedback. Through experiments on multiple sole variations, the system achieved a 95% success rate, validating its robustness against shape and burr variability. Unlike prior approaches, the framework avoids reliance on CAD templates or human demonstrations and can be directly integrated with existing industrial trimming machines, lowering deployment barriers. Future work will extend validation to real cutting operations with integrated force signals, enabling full admittance control. Additionally, we plan to benchmark cycle times against skilled human operators and evaluate scalability in real factory workflows. These efforts aim to establish robotic deburring as a practical alternative to manual finishing.

References

[1] International Federation of Robotics. `https://ifr.org/ifr-press-releases/news/record-of-4-million-robots-working-in-factories-worldwide`. 2024.

[2] Alatabani L. E. et al. "Machine Learning and Deep Learning Approaches for Robotics Applications". In: *Artificial Intelligence for Robotics and Autonomous Systems Applications*. Springer, 2023.

[3] Pillai A. et al. "Burr Registration Using Image Processing". In: *Advances in Industrial Machines and Mechanisms, Lecture Notes in Mechanical Engineering*. Ed. by Springer. 2021.

[4] Tafuro A. et al. "Towards intelligent robotic sole deburring: from burrs identification to path planning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7189–7195.

[5] Peng P. et al. "Tool path correction for robotic deburring using local non-rigid 3D registration". In: *Robotics and Computer-Integrated Manufacturing* 89 (2024), p. 102745.

[6] Ventuzelos V. et al. "Automating Lateral Shoe Roughing through a Robotic Manipulator Programmed by Demonstration". In: *2024 IEEE International Conference on Autonomous Robot Systems and Competitions*.

[7] Forlini M. et al. "Implementation and Testing of a Shoe Polishing Process with a Collaborative Robotic System". In: *Lecture Notes in Mechanical Engineering*. 2023.

[8] Oliver G. et al. "Towards footwear manufacturing 4.0: shoe sole robotic grasping in assembling operations". In: *International Journal of Advanced Manufacturing Technology* 114 (2021), pp. 811–827.

[9] Tafuro A. et al. "Self-supervised Vision-driven Trajectory Planning for Intelligent Robotic Deburring". In: *Machine Intelligence Research* 22.4 (2025).

[10] Schumacher M. et al. "An introductory review of active compliant control". In: *Robotics and Autonomous Systems* 119 (2019).

[11] Matour M. E. et al. "Force Controlled Deburring using a Collaborative Robot". In: *2022 International Conference on Methods and Models in Automation and Robotics*.

[12] Onstein I. F. et al. "Deburring using robot manipulators: A review". In: *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems*.

[13] Bahce E. et al. "Burr measurement method based on burr surface area". In: *IJPEM-Green Technology* 8 (2021), pp. 1287–1296.

[14] Lee Y. D. et al. "Robotic deburring strategy using burr shape recognition". In: *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3, pp. 1513–1518.

[15] Lai Z. et al. "Integration of visual information and robot offline programming system for improving automatic deburring process". In: *2018 IEEE International Conference on Robotics and Biomimetics*.

[16] Solvang B. et al. "Vision based robot programming". In: *2008 IEEE International Conference on Networking, Sensing and Control*.

[17] "Force Control". In: *Advanced Textbooks in Control and Signal Processing*. Ed. by Springer. 2010, pp. 363–405.

[18] Scherzinger S. et al. "Forward dynamics compliance control (FDCC): A new approach to cartesian compliance for robotic manipulators". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017, pp. 4568–4575.

[19] Makhdoomi M. R. et al. *Evaluation of Position and Velocity Based Forward Dynamics Compliance Control for Robotic Interactions in Position Controlled Robots*. arXiv preprint:2210.13421. 2022.

[20] Shah N. H. et al. "Real-Time Path Correction of an Industrial Robot for Adhesive Application on Composite Structures". In: *SAE Technical Paper Series*. 2018.

[21] Ravi N. et al. *SAM 2: Segment Anything in Images and Videos*. arXiv preprint:2304.02643. 2024.

[22] Dice L.R. "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26 (1945), pp. 297–302.

[23] Hart P. et al. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (), pp. 100–107.

[24] Calderón B. et al. "An adaptable vacuum gripper design for egg packaging attached to the KUKA KR 60-3 industrial robot". In: *LACCEI International Multiconference for Engineering, Education and Technology*. 2023, pp. 1–8.

[25] Jamaludin A. S. et al. "Design of spline surface vacuum gripper for pick and place robotic arms". In: *Journal of Modern Manufacturing Systems and Technology* (2020), pp. 48–55.

[26] *Colli FGB Website*. `https://collifgb.it/rifilatrice-gp1/`. Accessed on August 2025.