Adaptive Toolpath Correction for Robotic Finishing Based on Workpiece Shape Deviation

Luka Drobilo, Mihovil Legin, Tomislav Staroveški, Danko Brezak

Abstract— **Thin-walled parts, especially those with complex surface geometries, are prone to geometric inaccuracies and structural distortions, which can lead to issues during finishing operations like sanding. Using a toolpath based on the reference part to machine deformed workpieces may compromise surface quality or even damage the machined surface. In robotic applications, force control is often used to adapt the toolpath, but its ability to compensate for tool orientation remains limited. Digitizing the part allows for toolpath and tool orientation corrections before machining, preventing surface defects caused by workpiece deviations. This paper proposes an algorithm for toolpath and tool orientation correction, with the developed software solution simulated on a virtual twin of the robot control unit.**

I. Introduction

Over the past few decades, manufacturing industries have increasingly adopted robotic solutions as alternatives to conventional machine tools due to their flexibility, cost-effectiveness, and ability to process complex workpiece geometries [1]. Although many robot-related challenges have emerged, extensive research has been conducted, making robots pivotal for advancements in modern manufacturing [2]. However, due to geometric intricacies and deviations from the reference model, achieving high geometric accuracy and machined surface quality when processing components with complex geometries remains a challenge [3].

Besides potentially affecting the functionality of the product, these deviations can also prevent the parts from fitting properly into clamping devices. An additional factor exacerbating the issue is the use of excessive clamping forces or applying force in inappropriate positions, such as positions that are not properly supported [4]. This issue is especially pronounced in thin-walled structures, where forces exerted by both clamping and cutting can cause elastic deformation, which can affect the final dimensions and surface quality of the part [5].

When performing finishing operations on components with significant geometric deviations, programming the toolpath according to the reference model can lead to inappropriate engagement of the tool, reducing surface quality or even damaging the surface if the toolpath is not adjusted. Dynamic correction of the Tool Centre Point (TCP) position can be achieved through force control, either by using collaborative robots (cobots) or a standard industrial robot equipped with a force-torque sensor on the end effector (EE) [6]. This approach is limited in its ability to detect tool orientation deviations, force sensing accuracy in case of cobots, and response time in both cases. An alternative is using a so-called Force Control Unit (FCU), a specially designed external device for position compensation mounted on the robot's EE. Such devices generally provide dynamic characteristics superior to those of robots and can maintain a constant axial force. A limitation of commercially available FCU devices is that they can only provide axial compensation without accounting for deviations in orientation.

Multiple studies have been undertaken to digitize the workpiece in order to ascertain the form and magnitude of deviations present on the part surface, enabling toolpath generation and correction. Workpiece digitization was used for toolpath generation in [7], [8], [9], and [10] in cases where either the reference model was not available, or the real workpiece was compared with existing CAD data. The Scan-N-plan suit of tools enables robot trajectory planning from 3D scan data though ROS-Industrial [11]. Kurc et al. [12] and Li et al. [13] determined machining allowances on turbine blades using a 3D scanner to define appropriate machining forces and material removal rates for each part of the blade surface. Cheng et al. [14] used a laser vision sensor for digitization and a proprietary active compliance mechanism to enhance the propeller blade sharpening process, while Alt et al. [15] developed *RoboGrind*, an AI-powered sanding robot with perception, program generation, planning, and control capabilities. A communication method for a robotic cell integrating a 3D scanner for toolpath corrections in a deburring process was presented by Kurc et al. [16].

Considerable research efforts have focused on generating toolpaths from CAD models or scanned data. However, generative approaches are still not robust enough for widespread industrial implementation. When accounting for the geometric complexities of machined parts, as well as unique characteristics of specific materials and processes, technological expertise remains essential for defining appropriate toolpaths, strategies, and machining parameters. By leveraging this expert knowledge and incorporating non-contact workpiece digitization in the preparatory phase, errors in tool position and orientation can be detected and corrected before the machining phase. While the majority of research focuses on toolpath generation and correcting tool position, deviations in tool orientation and their impact on the machined surface are often neglected. In operations with large tool contact surfaces, such as sanding, tool orientation can have a significant impact on surface quality. Therefore, further research into the effect of these deviations is required.

The paper presents algorithms and software solutions required for an adaptive machining process that uses a 3D scanner to digitize the workpiece and perform corrections to

*Reserach was supported by the European Regional Development Fund (ERDF) through the grant KK.01.1.1.07.0016 (ARCOPS).

L. Drobilo, M. Legin, T. Staroveški, D. Brezak are with the University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, Zagreb, Croatia {luka.drobilo@fsb.unizg.hr, miha.mihac5@gmail.com, tomislav.staroveski@fsb.unizg.hr, danko.brezak@fsb.unizg.hr}

both the toolpath and tool orientation. This process was then simulated on a virtual twin of the robot control unit, validating the proposed solutions for use in real-world scenarios. By automating the digitization and correction processes, the added time is minimal, making it applicable for both research and manufacturing purposes.

Chapter II presents the materials and methods, including the equipment, software, and operations used, followed by a detailed explanation of the toolpath and orientation correction algorithm. The correction process is then simulated and verified on a virtual robot control unit in Chapter III. Finally, conclusions and future research directions are presented in Chapter IV.

II. Materials and Methods

A. Simulation setup

The simulation setup is based on the ARCOPS robotic cell, located in the Laboratory for Machine Tools at the Faculty of Mechanical Engineering and Naval Architecture in Zagreb, as shown in Figure 1. This specialized cell, designed for research on robotic finishing operations, consists of three industrial robots, a rotary table, and a swivel rotary table.

The part of the cell included in the experimental setup, which is simulated in this paper and currently in its finalization stage, utilizes two of the three robots: the ABB IRBP 6660-205/1.9 (R1) and the ABB IRB 4600-40/2.55 (R2), along with the ABB IRBP A500 swivel rotary table (SRT). The R1 robot, configured for material removal applications, is equipped with a servo-electric FCU mounted on the robot's EE, with a Mirka AIROS 650CV Ø150 mm orbital sander (OS) incorporated as the primary tool for sanding operations. A specialized fixture mounted on the SRT secures the sample workpiece, while the R2 robot facilitates in-process monitoring using an ATOS 5X 3D scanner (ZEISS, Germany) mounted on its EE. The structured light 3D scanner emits monochromatic blue light at wavelengths between 440 nm and 470 nm—a spectrum rarely found in natural environments, ensuring robust measurement.

B. Test Sample Definition

For simulation purposes, neither the R2 nor the scanner was used. However, all algorithms and software were prepared as if the scanner was used to ensure proper verification. Instead of scanning the part, a geometric deviation was simulated using a 3D model with a deformation of known geometry and dimensions. The reference part is represented by a plate measuring 500×600×5 mm, while the deformed part is a plate of the same general dimensions but featuring a curved deformation protruding 15 mm above the non-deformed surface.

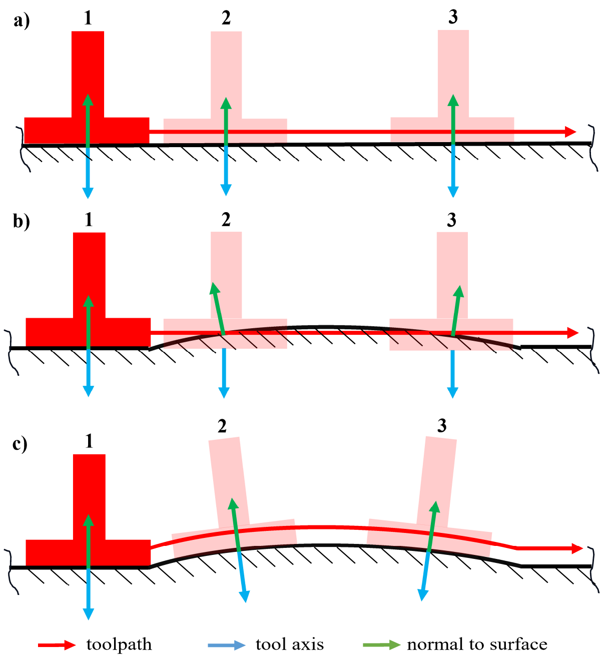


Figure 2. a) Reference part and reference toolpath,b) Deformed part and reference toolpath,c) Deformed part and corrected toolpath

Both the reference and deformed parts are shown in Figure 2, along with the influence of the introduced deviation on the tool. In the first case (a), the reference part and the

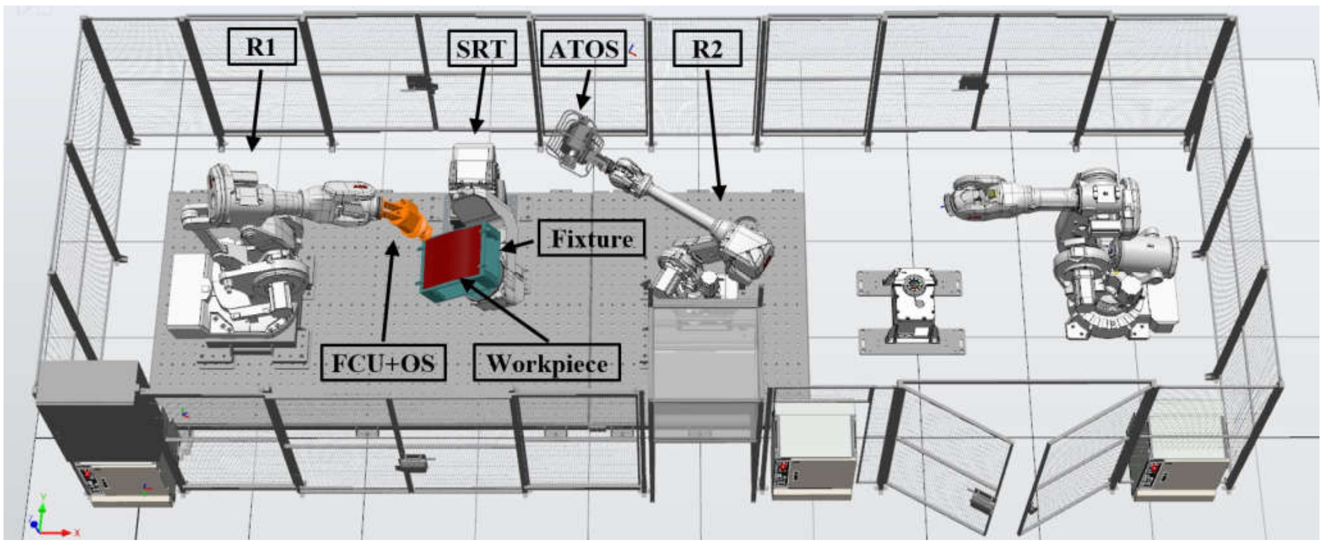


Figure 1. Robotic finishing cell ARCOPS

corresponding toolpath are presented. The second case (b) illustrates the deformed part with the tool following the original toolpath, while the third case (c) retains the deformed part but introduces a corrected toolpath and orientation based on the part deviation. When analyzing the second case, two adverse effects can be observed. First, there is an increase in axial force exerted by the tool onto the surface (and vice versa). Second, the specific pressure applied by the tool to the part surface increases. These factors combined can have a significant negative impact on both the machined surface and the tool itself. By modifying the toolpath and orientation, as shown in the third example, the unwanted increase in axial force is avoided, and the specific pressure exerted by the tool remains uniform.

The process was simulated in *RobotStudio*, a specialized software platform used for commissioning, programming and simulating ABB robots. Its main advantage lies in its ability to create virtual control units identical to those used to control the robots in the ARCOPS cell, enabling reliable simulation of the robot functionalities. Due to *RobotStudio's* limitations in Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) functionalities, two additional software packages—CATIA V5 (Catia) and *RoboDK*—were also used.

C. RAPID Code Preparation

Catia was first used to create the 3D CAD models of both the reference and deformed parts and to incorporate them into the fixture mounted on the SRT. In the second step, the machining process was defined using *Catia's* CAM module. For the purposes of the experiment, an arbitrary curve—defined with respect to the dimensions of the part and the diameter of the sanding paper—was used as the toolpath. The toolpath curve is positioned on the top surface of the reference part shown in Figure 3 above the deformed part for better visibility. The orientation of the tool axis along the entire path follows the direction of the vector normal to the surface of the reference part, i.e. the Z-axis direction. The Numerical Control (NC) code, which defines the tool's motions during the orbital sanding process, was generated and exported from *Catia* in APT format.

Given *RobotStudio's* limitations regarding the direct conversion of alternative code formats (such as APT or G-code), the *RoboDK* software was used as an intermediate platform. This software enables motion simulation, robot programming, and program generation independent of the robot manufacturer. In the context of this implementation, *RoboDK* was used to define the R1 and SRT motions using the NC code generated in *Catia* and to post-process the program into RAPID code compatible with ABB robots.

Considering the spatial arrangement of R1 and SRT, and the kinematic configuration of R1 limiting its motion range, the tilting functionality of the SRT was necessary for achieving toolpath execution. A rotation of 14° on the primary SRT axis enabled full motion execution; however, this angle was later increased to 30° to accommodate toolpath adjustments due to part deformation. The toolpath and tool orientations contained in this code were considered the reference toolpath and orientations suitable for machining operations on the reference part.

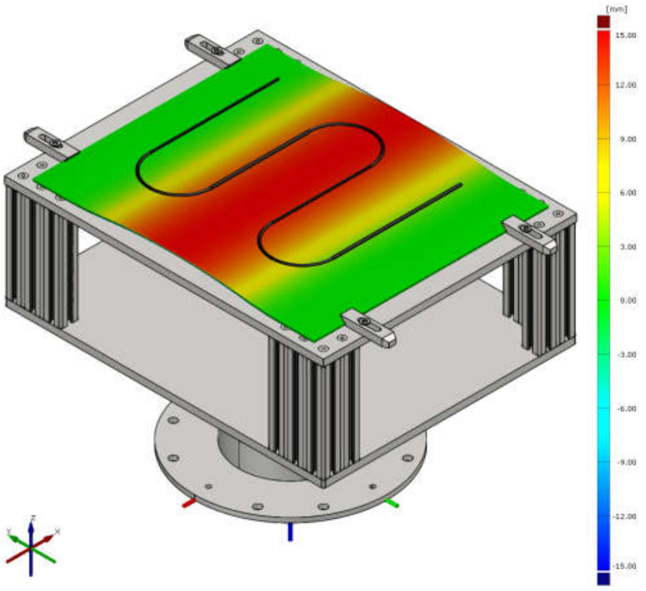


Figure 3. Deformed part mounted on the fixture with color-coded quantification of the part deformation and toolpath curve shown above the part in black

D. Toolpath Correction

The activities described in the previous section constitute the preparatory phase of the machining process. While specific details may vary depending on part characteristics, machining operations, software packages, and other factors, the general sequence—*modelling → programming → post-processing*—is typical for most machining operations. Given the deformation of the part, an additional phase is introduced between the preparatory and machining phases. This phase involves digitization of the clamped part to enable toolpath and orientation corrections.

The process, described in Figure 4, is managed through a Python script and can be broadly divided into four main steps:

1. Parsing data from the RAPID code
2. Surface deformation analysis
3. Toolpath and orientation correction
4. Writing the corrections into the RAPID code

The ZEISS INSPECT (ZI) software package is used for both scanning operations and the analysis of digitized images performed by the scanner. Consequently, this software was also used for part deviation analysis, which was necessary for implementing the required toolpath and orientation corrections into the RAPID code. ZI supports recording user command sequences (macros) and the Python Application Programming Interface (API), both of which were used to automate the analysis. Functions integrated into ZI (marked in blue in the diagram) were called, and information from objects was read using the API.

The initial step in the toolpath modification procedure involved importing both the reference and deformed models into the ZI environment, followed by precise alignment. This alignment provided a clear visualization of geometrically corresponding sections while revealing the morphology and propagation patterns of deformations throughout the non-conforming regions of the part (Figure 3).

Independent of the alignment process, all motion-related commands (mainly *MoveL*) were parsed from the RAPID program. *MoveL* represents a fundamental RAPID code instruction defining linear EE movement from the current pose to a specified EE pose and robot configuration. In addition to spatial coordinates and orientation parameters, *MoveL* commands encapsulate movement speed parameters, target robot configuration, positions of external robot axes, as well as active *WorkObject* coordinate system and tool parameters. Although the toolpath contains circular trajectories, such movements are conventionally approximated by sequences of linear movements in NC code. Each linear segment of the toolpath corresponds to a single move command, while the circular sections are divided into 29 segments—resulting in 29 *MoveL* commands per curve.

Data on TCP position and orientation was systematically extracted from each individual *MoveL* command. ABB robots use Cartesian coordinates (x, y, z) in millimeters to define the TCP position, while orientation is represented using quaternions. Quaternions [17] are hypercomplex numerical entities consisting of one real and three imaginary components which define orientation in three-dimensional space and are written as shown in Equation 1:

$$\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} = q_1 + q_2\mathbf{i} + q_3\mathbf{j} + q_4\mathbf{k}\ . \tag{1}$$

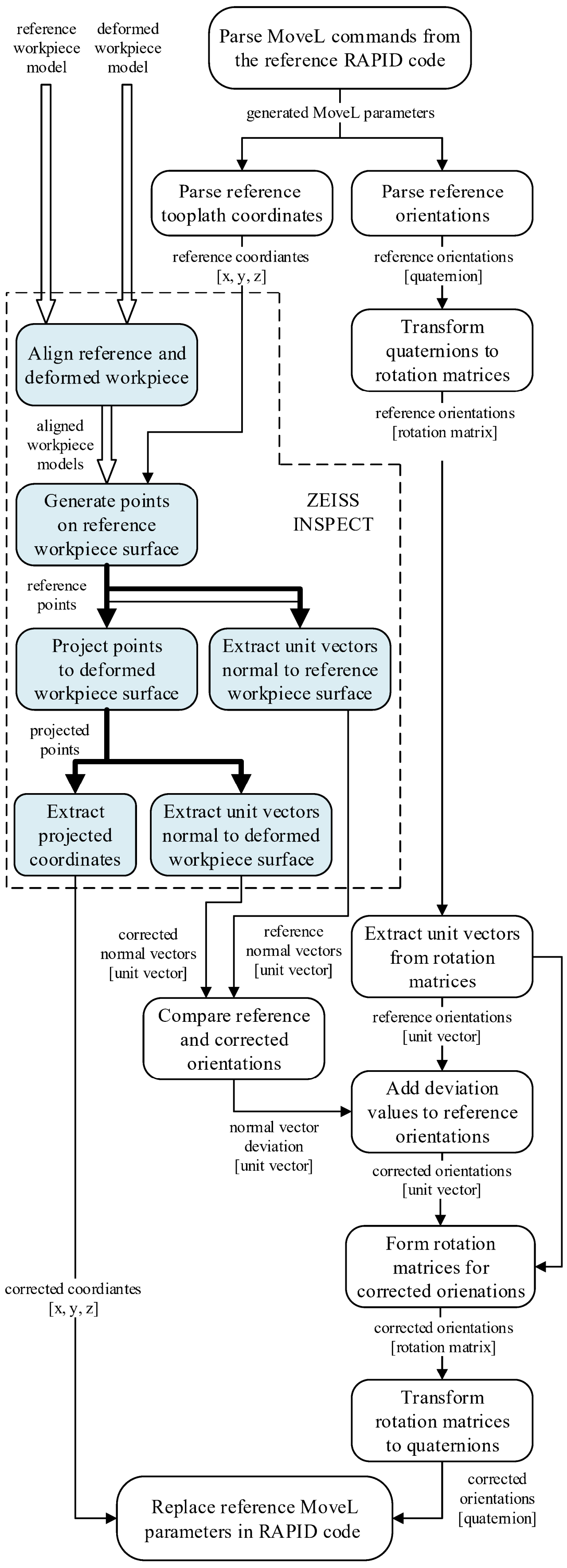
Although Euler angles are more commonly used in robotic applications due to their intuitive interpretation, they require additional information to define the angle sequence and are prone to singularities. In contrast, quaternions provide an unambiguous representation using only the four-digit notation, eliminating the possibility of singularities, making them particularly advantageous for this type of application. All orientation data obtained from the parsed quaternions was converted into rotation matrices using Equation 2 [18]:

$$\mathbf{R}_{Ref} = \begin{bmatrix} 1-2q_3^2-2q_4^2 & 2q_2q_3-2q_1q_4 & 2q_2q_4+2q_1q_3 \\ 2q_2q_3+2q_1q_4 & 1-2q_2^2-2q_4^2 & 2q_3q_4-2q_1q_2 \\ 2q_2q_4-2q_1q_3 & 2q_3q_4+2q_1q_2 & 1-2q_2^2-2q_3^2 \end{bmatrix}. \tag{2}$$

Since unit vector notation is used in ZI and the tool axis is positioned along the Z-axis direction, Z-axis unit vectors ($\hat{\mathbf{u}}_{zRef}$) were obtained by extracting the values of the third column of the rotation matrix for each point:

$$\mathbf{R}_{Ref} = \begin{bmatrix} \hat{\mathbf{u}}_{xRef} & \hat{\mathbf{u}}_{yRef} & \hat{\mathbf{u}}_{zRef} \end{bmatrix}. \tag{3}$$

The reference toolpath coordinates were used to create a series of individual point objects on the reference part surface. Using the 'Intersection with Mesh' function, these *Nominal* point objects were then projected onto the deformed part surface. By selecting a point object, it is projected along the chosen axis—in this case, the negative direction of the Z-axis—until the vector intersects with the first *Actual* surface of the ZI model, which, in this case, is the top surface of the deformed part. This process generated a new series of point objects that defined the corrected toolpath on the top surface of the deformed part (Figure 5). The three-dimensional Cartesian coordinates of all projected points were extracted



reference workpiece model
deformed workpiece model
Parse MoveL commands from the reference RAPID code
generated MoveL parameters
Parse reference tooplath coordinates
Parse reference orientations
reference coordiantes [x, y, z]
reference orientations [quaternion]
Transform quaternions to rotation matrices
reference orientations [rotation matrix]
Align reference and deformed workpiece
aligned workpiece models
Generate points on reference workpiece surface
ZEISS INSPECT
reference points
Project points to deformed workpiece surface
Extract unit vectors normal to reference workpiece surface
projected points
Extract projected coordinates
Extract unit vectors normal to deformed workpiece surface
corrected normal vectors [unit vector]
reference normal vectors [unit vector]
Extract unit vectors from rotation matrices
Compare reference and corrected orientations
reference orientations [unit vector]
Add deviation values to reference orientations
normal vector deviation [unit vector]
corrected orientations [unit vector]
Form rotation matrices for corrected orienations
corrected coordiantes [x, y, z]
corrected orientations [rotation matrix]
Transform rotation matrices to quaternions
corrected orientations [quaternion]
Replace reference MoveL parameters in RAPID code

Figure 4. Toolpath and tool orientation correction process

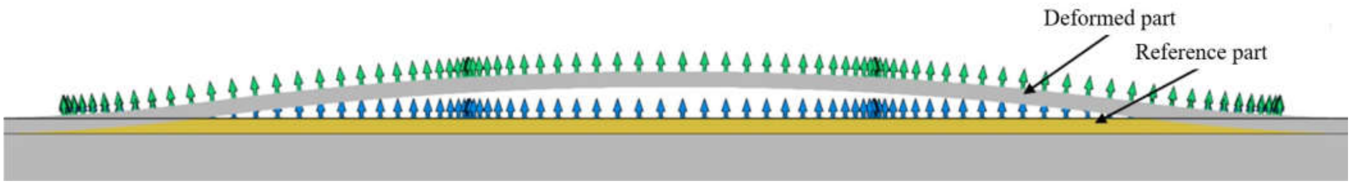


Figure 5. Point and vector objects in reference to their respective models in ZEISS INSPECT: a) reference toolpath (blue), b) corrected toolpath (green)

from these objects via the Python API and integrated into the appropriate *MoveL* commands.

For both sets of points objects—those on the reference part surface and those on the deformed part surface—unit vector objects normal to the respective surfaces were created by ZI at each point (Figure 5). Each of these vectors ($\hat{\mathbf{v}}$) was defined by projecting a vector normal to the surface in that point outward from the surface in the positive direction of the mesh element. Data defining each unit vector was extracted from the vector objects via the Python API and recorded in the appropriate vector set, either for the reference or deformed part. For each toolpath point, the deviation of the deformed normal vector from the reference normal vector was calculated (Equation 4). These deviations were then added to the reference orientations, originally parsed from the RAPID code, forming the corrected orientation vector (Equation 5).

$$\triangle\mathbf{u}_z = \hat{\mathbf{v}}_{Def} - \hat{\mathbf{v}}_{Ref} \quad (4)$$

$$\mathbf{u}_{zCor} = \hat{\mathbf{u}}_{zRef} + \triangle\mathbf{u}_z \quad (5)$$

After adding the deviations and correcting the reference orientations, the corrected orientations were transformed back into rotation matrix notation. The first column vector of the rotation matrix was calculated using Equation 6, the second using Equation 7 and for the third column the corrected orientation vector was used. Before completing the rotation matrix, as shown in Equation 8, all three vectors had to be normalized.

$$\mathbf{u}_{yCor} = \mathbf{u}_{zCor} \times \hat{\mathbf{u}}_{xRef} \quad (6)$$

$$\mathbf{u}_{xCor} = \mathbf{u}_{yCor} \times \mathbf{u}_{zCor} \quad (7)$$

$$\mathbf{R}_{Cor} = \begin{bmatrix} \hat{\mathbf{u}}_{xCor} & \hat{\mathbf{u}}_{yCor} & \hat{\mathbf{u}}_{zCor} \end{bmatrix} \quad (8)$$

The rotation matrices were then converted back to quaternions using the code available in [19], replacing the reference orientation values in the corresponding *MoveL* commands.

Finally, the modified *MoveL* commands were written into the original RAPID code, replacing the reference toolpath coordinates and orientation values to align the toolpath and tool orientation with the actual part.

III. Results and Discussion

After the modified RAPID code was loaded into *RobotStudio* and the machining simulation was conducted, a clear deviation in the programmed movement of the sanding tool relative to the deformed part became evident. Figure 6 presents three sets of images comparing positions from both the reference program (top three images) and the modified program (bottom three images). The X, Y, and Z axes of the robot tool coordinate system are depicted in blue, green, and red, respectively. Although both toolpaths follow the top surface of the reference part (i.e., the deformed part), they are offset below the part for clarity of presentation.

In the top three images, which visualize the reference toolpath, all toolpath points lie within a plane parallel to the fixture's clamping plate, positioned on the surface of the reference part. Along the length of the toolpath, the Z-axis vector orientation remains constant, with variations occurring only in rotation around the Z-axis, causing corresponding changes in the X-axis and Y-axis vectors. A visual inspection of the tool reveals that it protrudes into the machined surface.

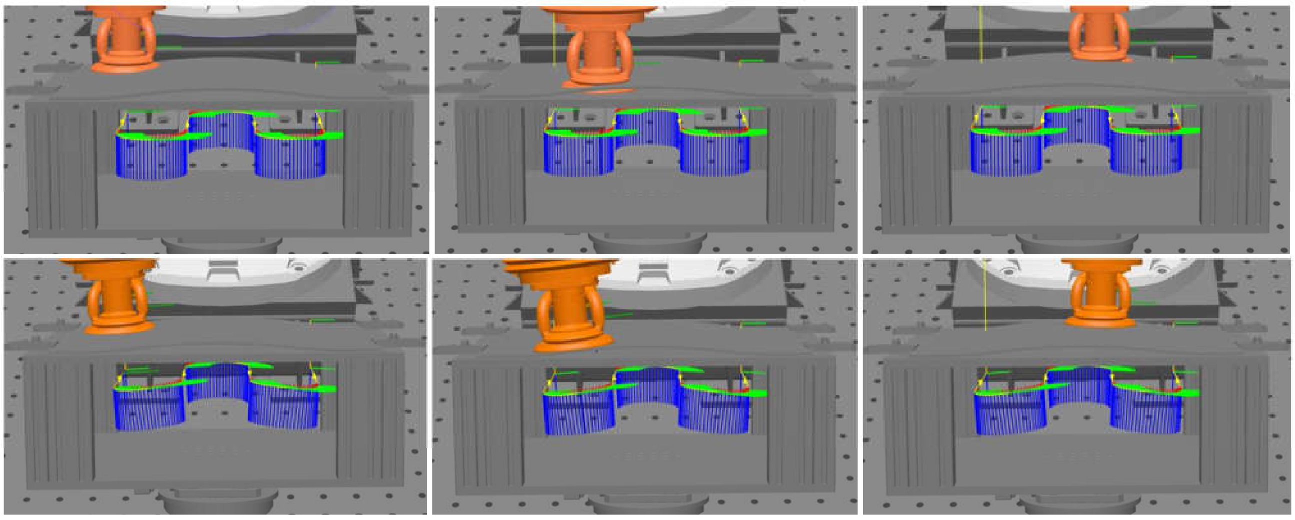


Figure 6. Machining simulation: reference toolpath (up), corrected toolpath (down)

Conversely, the bottom three images (Figure 6) illustrate the modified toolpath, which successfully conforms to the curvature of the deformed part. In this modified configuration, the Z-axis direction follows the normal vectors of the part surface at each position. Furthermore, the tool no longer intrudes into the part but instead traces the toolpath while maintaining a consistent relative position and orientation with respect to the part surface.

A detailed examination of the point objects and unit vectors at each position along the toolpath, as depicted in Figure 5, confirms that all points align accurately with the deformed surface. Additionally, the tool orientation vectors properly conform to the surface slope at each position. These observations verify that the software implementation functions as intended according to the proposed methodology.

Although the simulation supports the functional principle of the system, only practical experiments can confirm its real-life applicability, as additional limiting or obstructing factors may arise. To develop a robust automated system, further insight into factors that could affect its practical applicability is required so that appropriate solutions can be proposed to address these issues.

IV. Conclusion and Future Research

In this paper, an algorithm for toolpath correction in response to deformations present on the digitized part was proposed. Simulation on a virtual version of the robot control unit demonstrated that the proposed approach functioned as intended and met all expectations.

Once the robotic cell becomes operational, the tests simulated in this study will be experimentally implemented and further expanded. This will include analyzing the effects of varying machining parameters—combined with both the reference and corrected toolpaths and orientations—on the quality of the machined surface. Additionally, separate tests will be conducted to evaluate toolpath correction and orientation corrections individually, as well as their combined influence with and without the use of an FCU. Factors such as scan quality, surface complexity, and other elements that might affect the system's applicability or limit the level of automation will also be examined.

The scanner used in the ARCOPS cell is a highly expensive piece of equipment. Future research will also explore potential usage of more cost-effective alternatives and assess their suitability for integration with the proposed toolpath correction system.

References

[1] H. Kim, H. Jin, C. Moon, S. Kim, T. Kim, and T. Seo, "The Abrasion Robotic Solutions: A review," *INT. J. PRECIS. ENG. MAN-GT.*, vol. 12, no. 1, pp. 381–407, Jan. 2025, doi: 10.1007/s40684-024-00657-1.

[2] A. F. V. Pedroso *et al.*, "An overview on the recent advances in robot-assisted compensation methods used in machining lightweight materials," *Robotics and Computer-Integrated Manufacturing*, vol. 91, p. 102844, Feb. 2025, doi: 10.1016/j.rcim.2024.102844.

[3] D. Zhu *et al.*, "Robotic grinding of complex components: A step towards efficient and intelligent machining – challenges, solutions, and applications," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101908, Oct. 2020, doi: 10.1016/j.rcim.2019.101908.

[4] H. Liu, C. Wang, T. Li, Q. Bo, K. Liu, and Y. Wang, "Fixturing technology and system for thin-walled parts machining: a review," *Front. Mech. Eng.*, vol. 17, no. 4, p. 55, Dec. 2022, doi: 10.1007/s11465-022-0711-5.

[5] I. Del Sol, A. Rivero, L. N. López De Lacalle, and A. J. Gamez, "Thin-Wall Machining of Light Alloys: A Review of Models and Industrial Approaches," *Materials*, vol. 12, no. 12, p. 2012, Jun. 2019, doi: 10.3390/ma12122012.

[6] S. Gadringer, H. Gattringer, and A. Mueller, "Assessment of force control for surface finishing – an experimental comparison between Universal Robots UR10e and FerRobotics active contact flange," *Mech. Sci.*, vol. 13, no. 1, pp. 361–370, Apr. 2022, doi: 10.5194/ms-13-361-2022.

[7] G. Wang *et al.*, "Trajectory Planning and Optimization for Robotic Machining Based On Measured Point Cloud," *IEEE Trans. Robot.*, vol. 38, no. 3, pp. 1621–1637, Jun. 2022, doi: 10.1109/TRO.2021.3108506.

[8] J. Oyekan, M. Farnsworth, W. Hutabarat, D. Miller, and A. Tiwari, "Applying a 6 DoF Robotic Arm and Digital Twin to Automate Fan-Blade Reconditioning for Aerospace Maintenance, Repair, and Overhaul," *Sensors*, vol. 20, no. 16, p. 4637, Aug. 2020, doi: 10.3390/s20164637.

[9] S. Diao, X. Chen, and J. Luo, "Development and Experimental Evaluation of a 3D Vision System for Grinding Robot," *Sensors*, vol. 18, no. 9, p. 3078, Sep. 2018, doi: 10.3390/s18093078.

[10] X. Zhen, J. C. Y. Seng, and N. Somani, "Adaptive Automatic Robot Tool Path Generation Based on Point Cloud Projection Algorithm," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain: IEEE, Sep. 2019, pp. 341–347. doi: 10.1109/ETFA.2019.8869301.

[11] "Scan-N-Plan," ROS-Industrial. Accessed: Apr. 07, 2025. [Online]. Available: https://rosindustrial.org/scan-n-plan

[12] K. Kurc *et al.*, "Application of a 3D Scanner in Robotic Measurement of Aviation Components," *Electronics*, vol. 11, no. 19, p. 3216, Oct. 2022, doi: 10.3390/electronics11193216.

[13] W. Li, H. Xie, G. Zhang, S. Yan, and Z. Yin, "3-D Shape Matching of a Blade Surface in Robotic Grinding Applications," *IEEE/ASME Trans. Mechatron.*, vol. 21, no. 5, pp. 2294–2306, Oct. 2016, doi: 10.1109/TMECH.2016.2574813.

[14] Y.-S. Cheng, S. H. Shah, S.-H. Yen, A. R. Ahmad, and C.-Y. Lin, "Enhancing Robotic-Based Propeller Blade Sharpening Efficiency with a Laser-Vision Sensor and a Force Compliance Mechanism," *Sensors*, vol. 23, no. 11, p. 5320, Jun. 2023, doi: 10.3390/s23115320.

[15] B. Alt *et al.*, "RoboGrind: Intuitive and Interactive Surface Treatment with Industrial Robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan: IEEE, May 2024, pp. 2140–2146. doi: 10.1109/ICRA57147.2024.10611143.

[16] K. Kurc *et al.*, "Robotic machining in correlation with a 3D scanner," *Mechanics and Mechanical Engineering*, vol. 24, no. 1, pp. 36–41, Sep. 2020, doi: 10.2478/mme-2020-0003.

[17] R. Goldman, "Understanding quaternions," *Graphical Models*, vol. 73, no. 2, pp. 21–49, Mar. 2011, doi: 10.1016/j.gmod.2010.10.004.

[18] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*, 5. print., 1. paperback ed., [Nachdr.]. Princeton, NJ: Princeton University Press, 2007.

[19] B. Martin, "Maths - Conversion Matrix to Quaternion - Martin Baker." Accessed: Apr. 03, 2025. [Online]. Available: https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/ethan.htm