Technologia i Automatyzacja Montażu

Volume 123, Issue 1/2024, Pages 37-45
https://doi.org/10.7862/tiam.2024.1.6

Original Research

DEVELOPMENT OF A BASIC CAM PROCESSOR FOR A COLLABORATIVE ROBOT FOR WORKSHOP AUTOMATION

OPRACOWANIE PODSTAWOWEGO PROCESORA CAM DLA ROBOTA WSPÓŁPRACUJĄCEGO NA POTRZEBY AUTOMATYZACJI PRAC WARSZTATOWYCH

Andrzej CHMIELOWIEC[1,*], Karol ŁYSIAK[1], Jindřich VILIŠ[2]

[1] Faculty of Mechanics and Technology, Rzeszow University of Technology, Kwiatkowskiego 4, Stalowa Wola, Poland
[2] Department of Mechanical Engineering, Faculty of Military Technology, University of Defence, Kounicova 65, Brno, Czech Republic

* Corresponding author: achmie@prz.edu.pl

Abstract

The article presents an algorithm for constructing a tool path for a collaborative robot. This topic is significant due to the increasing popularity and prevalence of collaborative robots, alongside the lack of software for rapid path generation based on CAD models. Since the dynamics and control of a collaborative robot significantly differ from those of industrial robots and CNC machines, it is necessary to apply an approach that considers the limitations of the cobot. Beyond the tool path construction algorithm itself, the article presents the results of experiments carried out on an actual robot using a programmed CAM processor.

Keywords: collaborative robot, CAM, path generation, workshop automation

Streszczenie

Artykuł prezentuje algorytm tworzenia ścieżki narzędzia dla robota współpracującego. Temat ten jest bardzo istotny ze względu na rosnące rozpowszechnienie tego rodzaju robotów przy jednoczesnym braku oprogramowania pozwalającego na efektywne tworzenie ścieżki ruchu na podstawie modeli CAD. Ponieważ dynamika oraz system sterowania robota współpracującego znacznie różnią się od tych stosowanych w robotach przemysłowych i maszynach CNC, to konieczne jest zastosowanie podejścia dedykowanego kobotom. Poza algorytmem tworzenia ścieżki narzędzia artykuł prezentuje wyniki eksperymentów przeprowadzonych na rzeczywistym robocie przy użyciu zaimplementowanego procesora CAM.

Słowa kluczowe: robot współpracujący, robot kolaboracyjny, CAM, generowanie ścieżki, automatyzacja warsztatu

1. Introduction

Nowadays, technological advances and constant innovations in industry are offering new perspectives in the field of automation and robotics. These innovations are becoming a major source of efficiency and precision in manufacturing processes. A fundamental transformation is the gradual transition from conventional three-axis CNC machines to collaborative robots, known as cobots (Weidemann et al. 2023; Matheson et al. 2019). In this context, the question is - how to efficiently convert G-code, which is originally designed for three-axis CNC machines, into instructions capable of being implemented safely and efficiently in a collaborative robot platform? This issue symbolizes the uniting of two different spheres of industrial production. On the one side are traditional CNC machines, while on the other side are modern

cobots that offer the dynamics, flexibility and the ability to safety interact with human operators to the industrial system (Matheson et al. 2019).

Numerically controlled machine tools, or CNC machines, operate based on coded program instructions, allowing materials to be processed according to strict specifications (Suh et al. 2008). They eliminate the necessary direct manual control of operations. They operate with precisely defined trajectories within fixed workspaces. Trajectory planning on CNC machines is usually controlled by a geometric model created in CAD software (Suh et al. 2008; Ali and Mohsin 2021). The program instructions are passed to the CNC machine in the form of a sequential program that controls the machine behavior, mainly through G-codes. G-codes contain detailed information about the exact positions, feed rates, cutting depth and other parameters that affect the actual machining process (Suh et al. 2008; Ali and Mohsin 2021; Gayathri et al. 2022; Abd Rahman et al. 2023). There are many user interfaces and control programs for CNC machines, but most of them work with vector data (Khan, Shukla, and Singh 2018). For the cutting operations such as milling or turning, M-codes are also used to control the mechanisms of the machine tool (Suh et al. 2008). These M-codes control cooling fluid pumps, signalizing the need for tool changes and providing information about other modular functions. These machines were designed for accurate, repetitive and effective machining of materials. Their main disadvantage is the strict separation of the work area from human operators (Suh et al. 2008; Ali and Mohsin 2021; Gayathri et al. 2022; Abd Rahman et al. 2023; Khan, Shukla, and Singh 2018; Kheirabadi et al. 2023).

One of the possibilities is the use of collaborative robots (Kheirabadi et al. 2023). They are conceptualized with the primary purpose of collaboration and their design reflects the ability to synchronize with human workers. Their main goal is to efficiently perform the tasks that demonstrate high physical and exhaustive character (Berx, Decré, and Pintelon 2024; Chutima 2023; Jocelyn et al. 2023). Specifically, these tasks require high precision in a restricted space, such as precise manipulation of small components or accurate assembly (Hu 2023). For trajectory conversion for cobots, special attention is focused on safety aspects. Trajectory adjustments are necessary to eliminate potential risk of injury, which requires careful trajectory correction with regard to the cobot's ability to operate in the space which is shared with human operators (Toledano-García et al. 2023). The preparation of trajectories emphasizes user-friendly programming that allows quick and intuitive task setup for rapid response to dynamically changing challenges (Toledano-García et al. 2023; Faulwasser et al. 2016; Khoramshahi and Billard 2019). In this context, there is a need for the development of a CAM processor specifically designed for cobots to enable intuitive programming and customization of work trajectories without deep programming knowledge.

Several authors and scientific researchers are increasingly focusing on this innovative area, trying to understand the complex aspects involved in integrating cobots into the work environment. In a review of the evaluation of intelligent collaborative robots, Da Silva et al. (Da Silva et al. 2023) discussed the potential of CAD models for simulating realistic scenarios. They emphasized that simulation using CAD models allowed realistic testing of cobot behavior and was effective for prototype validation. However, simulation can be limited in some situations, especially during validation, so the authors recommend using CAD models in combination with other testing methods for comprehensive performance evaluation. In the study (Parsa and Saadat 2021), authors Soran Parsa and Mozafar Saadat focused on the utilization of CAD models to optimize disassembly planning for end-of-life products. Their goal was to combine human flexibility with robot precision to improve the efficiency of disassembly operations. The work resulted in the disassembly planning method that uses CAD models to identify and prioritize components for reusability. Nagata et al. (Nagata et al. 2013) developed the robotic CAM system for the RV1A industrial robot. This system allows the move of the robot according to tool position and orientation data without using a robot language and it serves as an interface between a conventional CAD/CAM system and an industrial robot. Through the collaboration between the main processor of the CAD/CAM system, the robotic servo controller, and the robot kinematics, the post-processing and learning process for generating robot languages was streamlined. Experimental results demonstrated that the system was able to successfully control robot motion based on tool position and orientation data.

The main goal of this study is to develop a user-friendly tool that enables intuitive trajectory programming for the HCR5 collaborative robot. The result will be the method of converting contours in a three-dimensional CAD model into instructions suitable for the HCR5 collaborative robot. In this way, we expect that the development of the CAM processor will improve the flexibility of approaches to automation in manufacturing, which will increase the efficiency of cobot implementation in industrial operations, and this will ultimately lead to optimization of manufacturing processes and an increase the total productivity.

2. Materials and Methods

Based on the CAD model, it is possible to generate a mesh representing curves, surfaces, and volumes. Curves are one-dimensional objects whose definition does not require significant memory and computational resources. However, curves themselves do not contain information about normal vectors to the surface on which they are located. A different situation arises in the case of a surface mesh, which provides the possibility of accurately determining normal vectors. In this section, conditions for the correct reconstruction of normal vectors based on the curve mesh and methods for their reconstruction based on the surface mesh will be presented.

The selected components of the CAD model were transformed into computational meshes, including both one-dimensional and two-dimensional types, using the open-source Gmsh software (Geuzaine and Remacle 2009). In Fig. 1, an example of a CAD model (blue outline) covered with a mesh of varying density (black and gray lines) is presented. The meshio library, designed for the Python language, enables reading such a format and numerically processing the information contained in it. In the CAM processor implemented for this article, numerical data contained in the mesh were converted into a graphical representation. Following this, the toolpath and the surface's normal vectors within that path were efficiently calculated using algorithms derived from Eulerian graphs and Hamiltonian cycles (Zhang and Guo 1986; Stapleton et al. 2010).

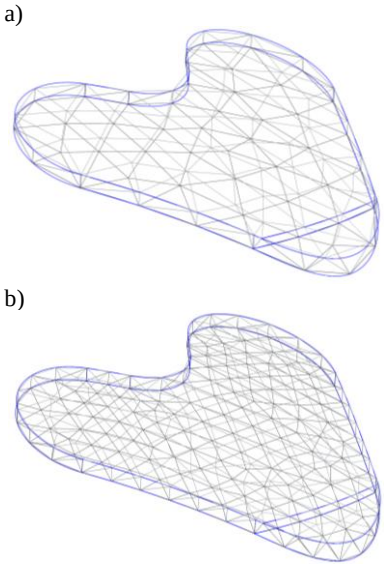

Fig. 1. CAD model with a mesh of (a) low density, (b) high density

Assume that $Q = (Q_1, Q_2, \dots, Q_m)$ represents the path defined by points Q_i between which the tool motion follows a linear trajectory. Therefore, this trajectory consists of a family of straight lines $\Gamma_i(t) = tQ_i + (1-t)Q_{i+1}$. The entire tool motion trajectory Γ can thus be considered as a sequence of individual component trajectories, expressed as $\Gamma = (\Gamma_1, \Gamma_2, \dots, \Gamma_{m-1})$. The first task is to reconstruct normal vectors to the surface on which the path Q is located. While it is not possible to achieve this exactly, we will assume that consecutive points on the path are close enough to each other so that, in the immediate vicinity, the surface can be approximated by a spherical tangent with a specified radius and center.

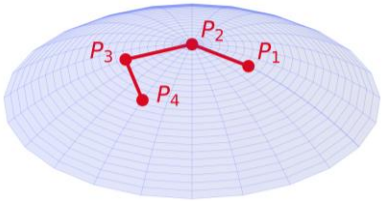

Fig. 2. Example path on a spherical dome

Consider four points $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, $P_3 = (x_3, y_3, z_3)$, and $P_4 = (x_4, y_4, z_4)$ which do not lie in the same plane (Fig. 2). The equation of a sphere with a center at $C = (x_C, y_C, z_C)$ and a radius r is given by equation (1).

$$(X - x_C)^2 + (Y - y_C)^2 + (Z - z_C)^2 = r^2 \quad (1)$$

If we assume that $d = x_C^2 + y_C^2 + z_C^2 - r^2$, then the above equation takes the form of equation (2).

$$X^2 + Y^2 + Z^2 - 2x_C X - 2y_C Y - 2z_C Z + d = 0 \quad (2)$$

The center and radius of the sphere, on which the four previously established points lie, can be explicitly determined. To achieve this, it is necessary to solve a system of linear equations (3).

$$\begin{pmatrix} 2x_1 & 2y_1 & 2z_1 & -1 \\ 2x_2 & 2y_2 & 2z_2 & -1 \\ 2x_3 & 2y_3 & 2z_3 & -1 \\ 2x_4 & 2y_4 & 2z_4 & -1 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ z_C \\ d \end{pmatrix} = \begin{pmatrix} x_1^2 + y_1^2 + z_1^2 \\ x_2^2 + y_2^2 + z_2^2 \\ x_3^2 + y_3^2 + z_3^2 \\ x_4^2 + y_4^2 + z_4^2 \end{pmatrix} \quad (3)$$

After finding the solution of linear system (3), the radius of the sphere is determined from the equation $r^2 = x_C^2 + y_C^2 + z_C^2 - d$.

In Fig. 3, a comparison of normal vectors obtained using the method described above (vectors in red) with vectors obtained from the analysis of the surface mesh (vectors in green) is presented. It was observed that when the surface curvature is constant or changes continuously, the reconstructed vectors coincide with

the vectors determined from the surface mesh. However, in the case of a sudden change in curvature, the reconstructed vectors deviate in their direction from the actual normal vectors. In Fig. 3, the discrepancy appears in the bottom right corner, where we encounter a radical change in curvature radius, and the second derivative of the surface function in the direction determined by the trajectory has a discontinuity point.

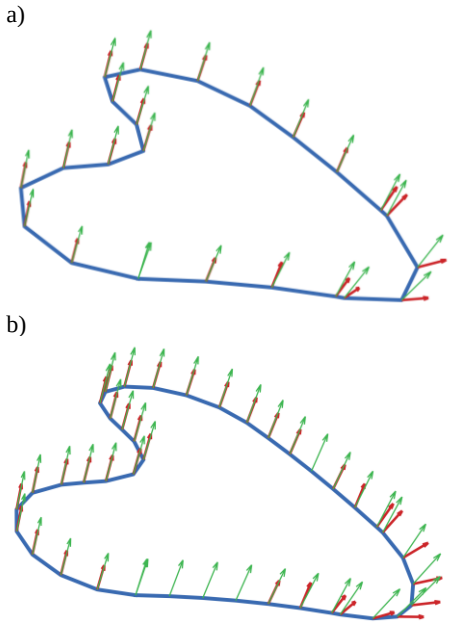

Fig. 3. Normal vectors to the surface obtained from the CAD model (green) and determined based on spherical approximation (red)

The CAD program allows to generate the output path Q with practically any precision. Distances between successive points on the path $|Q_iQ_{i+1}|$ can be so small that the physical movement of the robot between them is practically imperceptible. Considering the finite precision with which the drives of individual robot axes can be adjusted, this results in motion that is not smooth and proceeds much slower than it should. Therefore, the path Q must be adjusted to the physical capabilities of executing movements by a given robot. For each device, there is a minimum size of displacement that it can smoothly perform. Hence, it is necessary to develop a method for selecting from the specified path Q a subset $S \subset Q$ that ensures smooth movements while maintaining the trajectory within a specified tolerance ε.

Now suppose that we have a set of consecutive points on the path $T = (Q_k, Q_{k+1}, \dots, Q_{k+n})$. To determine the error produced by approximating the path T by the segment Q_kQ_{k+n}, it is necessary to determine the maximum value d_j, which is the distance from point Q_j to the segment Q_kQ_{k+n}.

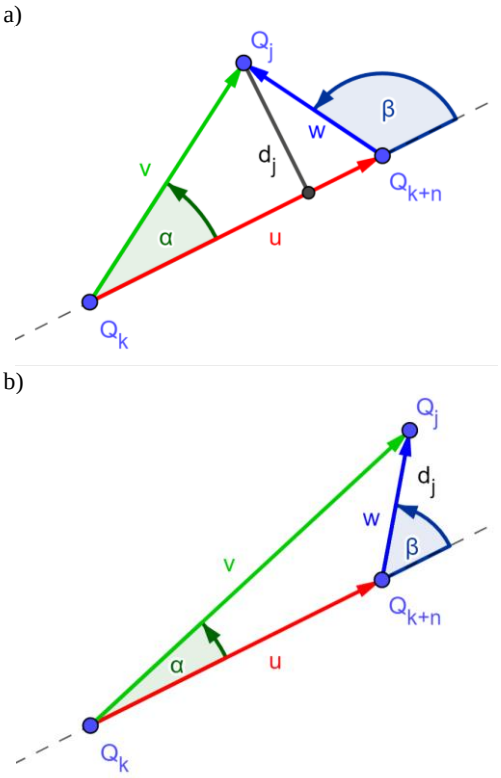

Fig. 4. Distance of point Q_j from segment Q_kQ_{k+n}

To determine the value of d_j, the properties of the cross and dot products can be utilized. In Fig. 4, two situations that may occur are presented. The first situation, illustrated in Fig. 4 (a), occurs when d_j is the distance from point Q_j to the segment Q_kQ_{k+n} – that is, d_j is the height of the triangle $Q_kQ_{k+n}Q_j$. If Q_j does not project onto the segment Q_kQ_{k+n}, as shown in Fig. 4 (b), then the distance d_j is equal to the shorter of the lengths of either side. The resolution of the both cases can be achieved by comparing the signs of the dot products. Case (a) occurs when the dot products $u \cdot v$ and $u \cdot w$ have opposite signs. In comparison, case (b) arises when the signs of both dot products are identical. Using the notations as in Fig. 4, the distance d_j can be expressed by the formula (4).

$$d_j = d\left(Q_j, Q_kQ_{k+n}\right) = \\ = \begin{cases} |u \times v|/|u|, & \text{if } (u \cdot v)(u \cdot w) \le 0, \\ \min\{|v|, |w|\}, & \text{if } (u \cdot v)(u \cdot w) > 0. \end{cases} \quad (4)$$

The introduced notations and formulas allow to define an algorithm whose task is to reduce the number of points on the robot's path. This algorithm takes the original path $Q = (Q_1, Q_2, \dots, Q_m)$ and a certain number ε as input arguments, where ε denotes the permissible deviation in determining the new path S for the robot's movement.

Algorithm for optimal path finding

1. read $Q = (Q_1, Q_2, \dots, Q_m)$ and ε;
2. $S \leftarrow (Q_1)$; remove first element from Q;
3. **while** $Q \neq ( )$ **do**
   a. $T \leftarrow ( )$;
   b. remove last element of S and append to T;
   c. remove first element of Q and append to T;
   d. **while** $Q \neq ( )$ **do**
      i. remove first element of Q and append to T;
      ii. assign first element of T as Q_k;
      iii. assign last element of T as Q_{k+n};
      iv. compute $d_{\max} = \max\{d_j\}$, where $d_j = d\left(Q_j, Q_k Q_{k+n}\right)$ for all Q_j different from Q_k and Q_{k+n};
      v. **if** $d_{\max} > \varepsilon$ **then** break loop 3.d;
   e. **if** $Q \neq ( )$ **then** remove last element of T and append as first to Q;
   f. remove first element of T and append to S;
   g. remove last element of T and append to S; go to point 3;
4. **return** S;

The effects of the above algorithm were presented in Fig. 5, which shows the path progression for the same target trajectory and different values of permissible deviation.

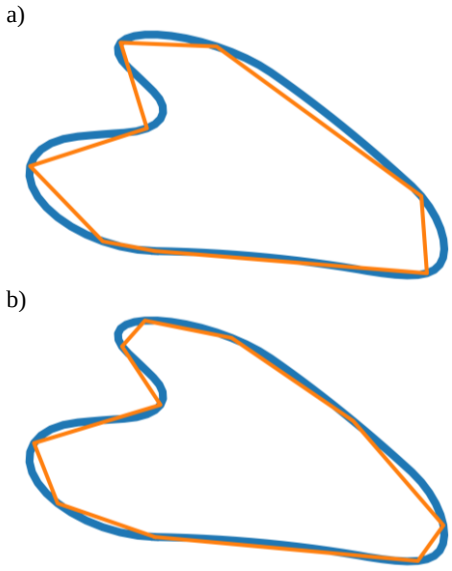

Fig. 5. Example paths (orange) for different values of deviation from the target trajectory (blue)

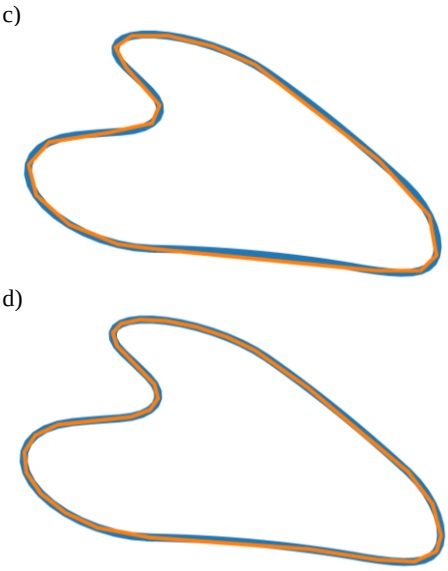

Fig. 5 (cont.). Example paths (orange) for different values of deviation from the target trajectory (blue)

3. Results

For the purposes of the article, CAM software was implemented to transform a MESH file (Gmsh msh version 2.0 ASCII format) into a set of commands that execute the movement of the robot along a given path. The Python language was used for the implementation along with the corresponding mathematical libraries (Chmielowiec 2021; Chmielowiec and Klich 2021). The main element of the created software is the path optimization algorithm presented in the previous section, which adjusts the tool's path of movement based on the size of the permissible deviation from the precise trajectory.

Table 1. Time and velocity of tool movement along a path approximating a circle with a radius of 50 mm using a regular n-sided polygon

		Vertical		Normal to surface	
n	v [mm/s]	t [s]	$\bar{v}$ [mm/s]	t [s]	$\bar{v}$ [mm/s]
6	5	45.2	6.63	45.2	6.64
6	10	22.8	13.18	22.7	13.20
6	20	11.5	25.99	11.6	25.97
6	40	6.0	49.64	6.1	49.45
6	80	3.6	84.01	3.6	84.10
12	5	54.7	5.68	54.9	5.66
12	10	27.6	11.27	27.8	11.17
12	20	14.4	21.64	14.1	22.02
12	40	7.6	40.72	7.6	40.71
12	80	4.9	63.97	5.1	60.61
24	5	59.3	5.29	59.5	5.26
24	10	30.0	10.43	30.0	10.43
24	20	15.7	19.90	15.5	20.19

Table 1 (cont.). Time and velocity of tool movement along a path approximating a circle with a radius of 50 mm using a regular n-sided polygon

		Vertical		Normal to surface	
n	v [mm/s]	t [s]	$\bar{v}$ [mm/s]	t [s]	$\bar{v}$ [mm/s]
24	40	9.1	34.35	9.2	34.16
24	80	6.6	47.49	6.6	47.51
48	5	62.1	5.05	62.1	5.05
48	10	31.7	9.92	31.9	9.84
48	20	16.8	18.64	16.9	18.52
48	40	10.8	29.03	10.9	28.89
48	80	9.9	31.65	10.2	30.72
96	5	63.7	4.93	64.1	4.90
96	10	33.4	9.42	33.5	9.36
96	20	18.6	16.92	18.8	16.68
96	40	13.9	22.67	14.2	22.11
96	80	15.7	19.97	16.0	19.63

The experiments were conducted on a HANWHA HCR-5 robot (Fig. 6), which was equipped with a special holder along with a pen. This made it possible to visually verify the accuracy of the algorithm generating the optimized path.

Fig. 6. Photo of test stand

In the first experiment, the velocity of robot movement was measured as it moved a tool along a path approximating a circle with a radius of 50 mm. The path was parameterized by the number n, which defines the number of vertices of a regular n-sided polygon approximating the circle. Tests were conducted for values of $n \in \{6,12,24,48,96\}$ and nominal velocity $v \in \{5,10,20,40,80\}$ expressed in mm/s. It should be emphasized that by the nominal velocity v, here we refer to the velocity set for the robot as one of the parameters of the motion command. Table 1 lists the times t of the tool's passage along the path and the average velocity of passage $\bar{v}$ when the tool is set vertically and perpendicular to the surface over which it moves. From the numerical data, it is clear that the orientation of the tool does not have a significant impact on the actual velocity of the tool passage along the path. The average relative difference in passage times between these two cases is indeed at the level of 1%. However, a significant change in the velocity of passage can be observed depending on the number of path points n.

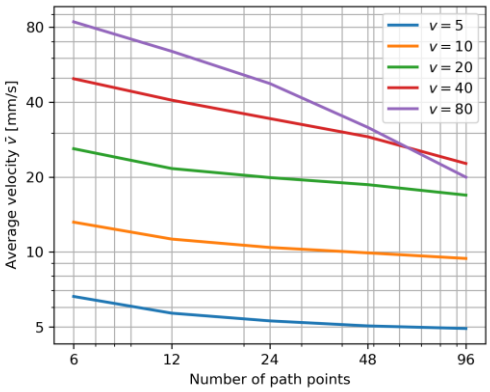

Fig. 7. Average velocity $\bar{v}$ as a function of the number of path points n for different nominal velocity v values

In Fig. 7, one can observe how the average velocity $\bar{v}$ of tool movement clearly decreases with the increasing number of path points. These decreases are significant enough that the use of a logarithmic scale was necessary for the clarity of the plot. Particular attention should be paid to the results obtained for the nominal velocity of $v = 80$ mm/s. The plot clearly shows that the average velocity in the case of a large number of path points drops even below the average velocity obtained for the nominal velocity of $v = 40$ mm/s (the purple curve is below the red for $n = 96$). This result is unexpected and means that in certain cases, it is possible to reduce the passage time of a given path by reducing the nominal velocity.

Table 2. Average time $\bar{t}$ and average valocity $\bar{v}$ of tool movement along shape presented in Fig. 1

			Vertical		Normal to surface	
v [mm/s]	ε [mm]	n	$\bar{t}$ [s]	$\bar{v}$ [mm/s]	$\bar{t}$ [s]	$\bar{v}$ [mm/s]
80	0.1	60	11.2	25.63	11.2	25.59
80	0.2	42	9.0	31.61	8.8	32.31
80	0.5	26	6.7	42.41	6.7	42.26
80	1.0	18	5.7	49.38	5.9	48.32
80	2.0	14	5.2	54.00	5.2	53.98

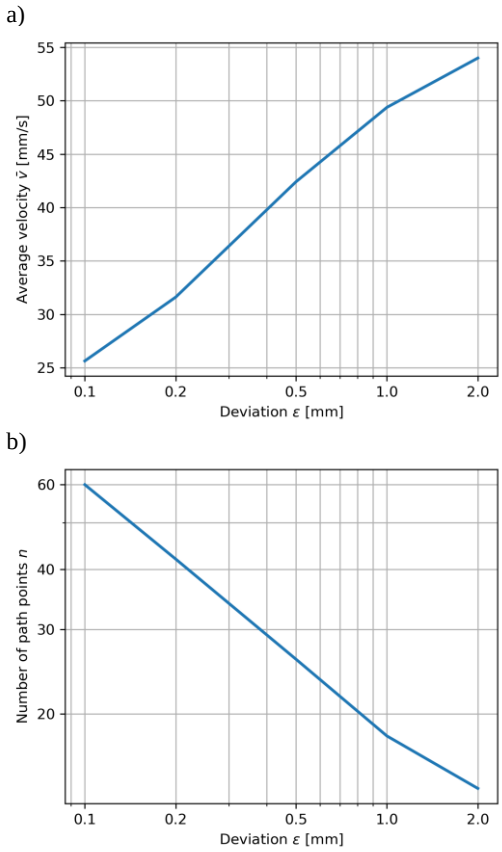

Fig. 8. Average valocity $\bar{v}$ as a function of: deviation ε (a), number of path points n generated by the algorithm as a function of deviation ε (b)

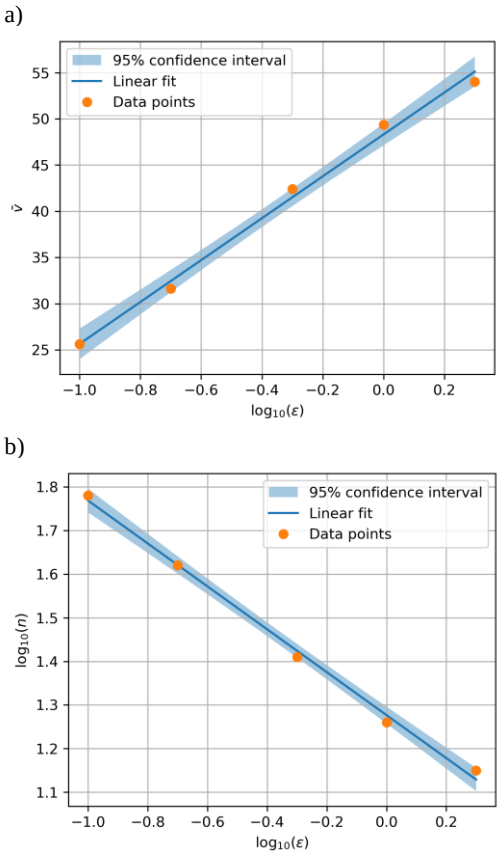

Fig. 9. Linear approximation of average velocity $\bar{v}$ as a function of: deviation logarithm $\log_{10}\varepsilon$ (a), linear approximation of path points number logarithm $\log_{10}n$ generated by the algorithm as a function of deviation logarithm $\log_{10}\varepsilon$ (b)

Subsequent measurements were conducted for the shape presented in Fig. 1. A path generation algorithm was used to generate motion instructions for the robot, whose tool would move along the specified trajectory with various values of deviation ε. For each selected value of ε, 5 measurements were conducted at a fixed nominal valocity of $v = 80$ mm/s. The average values of the obtained results have been presented in Table 2. In Fig. 8 the relationships between the adopted value of permissible deviation ε, and the average valocity of traversal along the path (Fig. 8 (a)) and the number of path points (Fig. 8 (b)) are presented.

The use of logarithmic scale for the ε and n axes shows linear relationships between the values obtained during the measurements. Using the method of least squares, the following relationships can be obtained for the studied intervals:

$$\bar{v} = 22.66 \cdot \log_{10}(\varepsilon) + 48.31 \quad (5)$$

$$\log_{10}(n) = -0.50 \cdot \log_{10}(\varepsilon) + 1.28 \quad (6)$$

Both relationships are graphically presented in Fig. 9 (a) and Fig. 9 (b), along with the marked set of measurement data and the confidence interval for the least squares method.

4. Summary

In this article, an algorithm is presented that allows to generate the robot's motion path based on data from a CAD model. The algorithm operates based on the precise trajectory of the tool and the deviation value ε, which defines the deviation of the generated path from the original trajectory. This algorithm is a greedy type of algorithm and has been implemented in a CAM processor prepared for the purposes of this article. The created tool was used to conduct experiments on an actual HANWHA HCR-5 collaborative robot. The tests showed that defining too high a nominal velocity in the robot's motion instructions can have the opposite effect to the intended one, and cause the traversal of a given path to slow down even compared to a lower nominal velocity. The obtained results clearly show that the best way to maintain an appropriate velocity of robot movement is to adjust the number of path points to the given nominal velocity. However, it should be remembered that too large number of path points significantly reduces the efficiency of robot movements and lowers the average velocity of passage. The relationship $\log_{10}(n) = -0.50 \cdot \log_{10}(\varepsilon) + 1.28$ derived from simulation

data can be simplified to the equation $\varepsilon = \frac{363}{n^2}$, which demonstrates the effectiveness of the presented algorithm. It implies that doubling the number of points results in a fourfold reduction in deviation from the desired trajectory. Despite this, it is worth analyzing other methods for generating and optimizing the robot's motion path in order to achieve the smallest possible number of points for a given motion precision.

The methods presented in the article can be applied in processes such as burning, milling, and applying adhesives and sealants.. The automation of these processes using collaborative robots is currently quite limited by the lack of suitable software tools for automatically generating robot instructions. The approach introduced allows for efficient and precise application of assembly adhesives and various types of sealants. It also enables cutting of various patterns on surfaces of practically any shape, which, at a later stage of production, allows for the assembly and attachment of other elements.

References

Abd Rahman, Z., Mohamed, S.B., Minhat, M., & Abd Rahman, Z. (2023). "Design and Development of 3-Axis Benchtop CNC Milling Machine for Educational Purpose." *International Journal of Integrated Engineering* 15 (1): 145–60. https://doi.org/10.30880/ijie.2023.15.01.013.

Ali, S. M., & Mohsin, H. (2021). "Design and Fabrication of 3-Axes Mini CNC Milling Machine." *IOP Conference Series: Materials Science and Engineering* 1094 (1): 012005. https://doi.org/10.1088/1757-899X/1094/1/012005.

Berx, N., Decré, W., & Pintelon, L. (2024). "A Tool to Evaluate Industrial Cobot Safety Readiness from a System-Wide Perspective: An Empirical Validation." *Safety Science* 170: 106380. https://doi.org/10.1016/j.ssci.2023.106380.

Chmielowiec, A. (2021). "Algorithm for error-free determination of the variance of all contiguous subsequences and fixed-length contiguous subsequences for a sequence of industrial measurement data." *Computational Statistics* 36 (4): 2813–40. https://doi.org/10.1007/s00180-021-01096-1.

Chmielowiec, A., & Klich, L. (2021). "Application of python libraries for variance, normal distribution and Weibull distribution analysis in diagnosing and operating production systems." *Diagnostyka* 22 (4): 89–105. https://doi.org/10.29354/diag/144479.

Chutima, P. (2023). "Assembly Line Balancing with Cobots: An Extensive Review and Critiques." *International Journal of Industrial Engineering Computations* 14 (4): 785–804. https://doi.org/10.5267/j.ijiec.2023.7.001.

Da Silva, M., Regnier, R., Makarov, M., Avrin, G., & Dumur, D. (2023). "Evaluation of Intelligent Collaborative Robots: A Review." In *2023 IEEE/SICE International Symposium on System Integration (SII)*, 1–7. IEEE. https://doi.org/10.1109/SII55687.2023.10039365.

Faulwasser, T., Weber, T., Zometa, P., & Findeisen, R. (2016). "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot." *IEEE Transactions on Control Systems Technology* 25 (4): 1505–11. https://doi.org/10.1109/TCST.2016.2601624.

Gayathri, N., Sundar, M., Sargurunathan, R., Sudharsan, R., & Sajith, A.. (2022). "Design of Voice Controlled Multifunctional Computer Numerical Control (CNC) Machine." In *2022 International Conference on Inventive Computation Technologies (ICICT)*, 657–63. https://doi.org/10.1109/ICICT54344.2022.9850659.

Geuzaine, C., & Remacle, J.-F. (2009). "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities." *International Journal for Numerical Methods in Engineering* 79 (11): 1309–31. https://doi.org/10.1002/nme.2579.

Hu, M. (2023). "Research on Safety Design and Optimization of Collaborative Robots." *International Journal of Intelligent Robotics and Applications* 7 (4): 795–809. https://doi.org/10.1007/s41315-023-00299-7.

Jocelyn, S., Ledoux, E., Marrero, I.A., Burlet-Vienney, D., Chinniah, Y., Bonev, I.A., Mosbah, A.B., & Berger, I. (2023). "Classification of Collaborative Applications and Key Variability Factors to Support the First Step of Risk Assessment When Integrating Cobots." *Safety Science* 166: 106219. https://doi.org/10.1016/j.ssci.2023.106219.

Khan, A., Shukla, A.K., & Singh, A. (2018). "Design and Fabrication of 3-Axis Computer Numerical Control (CNC) Milling Machine." *International Journal of Creative Research Thoughts* 6 (2): 1347–53.

Kheirabadi, M., Keivanpour, S., Chinniah, Y.A., & Frayret, Y.-M. (2023). "Human-Robot Collaboration in Assembly Line Balancing Problems: Review and Research Gaps." *Computers & Industrial Engineering* 186: 109737. https://doi.org/10.1016/j.cie.2023.109737.

Khoramshahi, M., & Billard, A. (2019). "A Dynamical System Approach to Task-Adaptation in Physical Human–Robot Interaction." *Autonomous Robots* 43: 927–46. https://doi.org/10.1007/s10514-018-9764-z.

Lin, R.-S. (2000). "Real-time surface interpolator for 3-D parametric surface machining on 3-axis machine tools." *International Journal of Machine Tools and Manufacture* 40 (10): 1513–26. https://doi.org/10.1016/S0890-6955(00)00002-X.

Matheson, E., Minto, R., Zampieri, E.G.G., Faccio, M., & Rosati, G. (2019). "Human–Robot Collaboration in Manufacturing Applications: A Review." *Robotics* 8 (4). https://doi.org/10.3390/robotics8040100.

Nagata, F., Yoshitake, S., Otsuka, A., Watanabe, K., & Habib, M.K. (2013). "Development of CAM System Based on Industrial Robotic Servo Controller Without Using Robot Language." *Robotics and Computer-Integrated Manufacturing* 29 (2): 454–62. https://doi.org/10.1016/j.rcim.2012.09.015.

Parsa, S., & Saadat, M. (2021). "Human-Robot Collaboration Disassembly Planning for End-of-Life Product Disassembly Process." *Robotics and Computer-Integrated Manufacturing* 71: 102170. https://doi.org/10.1016/j.rcim.2021.102170.

Stapleton, G., Rodgers, P., Howse, J., & Zhang, L. (2010). "Inductively generating Euler diagrams." *IEEE Tran-*

sactions on Visualization and Computer Graphics 17 (1): 88–100. https://doi.org/10.1109/TVCG.2010.28.

Suh, S.-H., Kang, S.K., Chung, D.-H., & Stroud, I. (2008). *Theory and Design of CNC Systems*. Springer Science & Business Media. https://doi.org/10.1007/978-1-84800-336-1.

Toledano-García, A.A., Pérez-Cabrera, H.R., Ortega-Cabrera, D., Navarro-Durán, D., & Pérez-Hernández, E.M. (2023). "Trajectory Generator System for a UR5 Collaborative Robot in 2D and 3D Surfaces." *Machines* 11 (9). https://doi.org/10.3390/machines11090916.

Weidemann, C., Mandischer, N., van Kerkom, F., Corves, B., Hüsing, M., Kraus, T., & Garus, C. (2023). "Literature Review on Recent Trends and Perspectives of Collaborative Robotics in Work 4.0." *Robotics* 12 (3). https://doi.org/10.3390/robotics12030084.

Zhang, F.-J., & Guo, X.-F. (1986). "Hamilton cycles in Euler tour graph." *Journal of Combinatorial Theory, Series B* 40 (1): 1–8. https://doi.org/10.1016/0095-8956(86)90060-2.

DOCUMENT CREATED WITH

PDF COMBINER

PDF Combiner is a free application that you can use to combine multiple PDF documents into one.

Three simple steps are needed to merge several PDF documents. First, we must add files to the program. This can be done using the Add files button or by dragging files to the list via the Drag and Drop mechanism. Then you need to adjust the order of files if list order is not suitable. The last step is joining files. To do this, click button Combine PDFs.

Main features:

secure PDF merging - everything is done on your computer and documents are not sent anywhere

simplicity - you need to follow three steps to merge documents

possibility to rearrange document - change the order of merged documents and page selection

reliability - application is not modifying a content of merged documents.

Visit the homepage to download the application:

www.jankowskimichal.pl/pdf-combiner

To remove this page from your document, please donate a project.