Research Article

Krzysztof Kurc*, Andrzej Burghardt, Dariusz Szybicki, Piotri Gierlak, Wojciech Łabuński, Magdalena Muszyńska, and Józef Giergiel

Robotic machining in correlation with a 3D scanner

https://doi.org/10.2478/mme-2020-0003
Received Nov 28, 2019; accepted May 13, 2020

Abstract: The article presents an original method of communication and data exchange in a robotic machining station consisting of two robots, a positioner and a 3D optic scanner. The task of one of the robots, equipped with a 3D optic scanner, was to receive point cloud of a detail (mould) attached to the positioner table. After detail digitalisation, the received point cloud was adjusted to (compared with) a model detail in the form of a CAD file in the Atos Professional software. In the software, casting material excesses were received in places selected on the detail. Values of the excesses and their coordinates were saved in the script and sent to the robot controller using TCP/IP protocol. The other of robots, equipped with the force control addition and the option of obtaining various processing tools, received sent excess and its coordinates. The other robot adjusted the processing parameters to random excesses, the value of which was received from measurements of the optic scanner of the first robot.

Keywords: Robot, scanner 3D, robotic machining, communication, automatic processing

***Corresponding Author: Krzysztof Kurc:** Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: kkurc@prz.edu.pl
Andrzej Burghardt: Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: andrzejb@prz.edu.pl
Dariusz Szybicki: Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: dszybicki@prz.edu.pl
Piotri Gierlak: Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: pgierlak@prz.edu.pl
Wojciech Łabuński: Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: w.labunski@prz.edu.pl
Magdalena Muszyńska: Rzeszow University of Technology, Faculty of Mechanical Engineering and Aeronautics, Rzeszow, Poland; Email: magdaw@prz.edu.pl

1 Introduction

Communication and data exchange between devices is currently one of the crucial stages of robotic stations design and construction, according to the idea of industry 4.0, as an umbrella term meaning integration of smart devices and systems and implementing changes into production processes in order to increase the production capacity and introduce flexible options. Cutting and grinding are examples of processes which require shared communication of numerous station elements in a way, so that information exchange is quick and uninterrupted. Exemplary reviews of robotic cutting may be found in Łygas and Danilczuk [1] and Song *et al.* [2]. Algorithms controlling force and their utilisation in processes are found in Refs [3–6]. Technological parameters' super vision and communication between many robots have been described in Refs [7–9]. Numerous authors use various communication protocols in their works, Profibus [10–12] and TCP/IP [13–17]. To use the necessary network, various programming languages may be used. Increasingly popular is Python [18–20], that is utilised, for example, in robotic applications [21]. Grinding process improvement may also be obtained using CAD/CAM models [22] or laser scans [23–25]. Laser systems equipped with triangulation for quick measuring of blades are presented in the Ref. [26]. 3D laser systems measuring blade geometry and comparing it to the model detail in order to manage robotic machining are included in the Ref. [27], while utilising GOM brand 3D scanner to measure blade geometry during control, servicing and repairs is presented in the Refs [28, 29].

None of the aforementioned papers contain information on obtaining and sending information between Atos Core 135 optic scanner and robot controller in an automatic mode for the purpose of path adaptation and adjustment.

Józef Giergiel: Institute of Technology State University of Applied Sciences in Nowy Sącz, Nowy Sącz, Poland; Email: jozef.giergiel@gmail.com

2 Robotic station

Figure 1 presents a design of a virtual robotic station with an implemented option of an automatic data exchange. As a consequence of using hardware and software additions, there is an option of adapting the robot trajectory to the detailed shape.

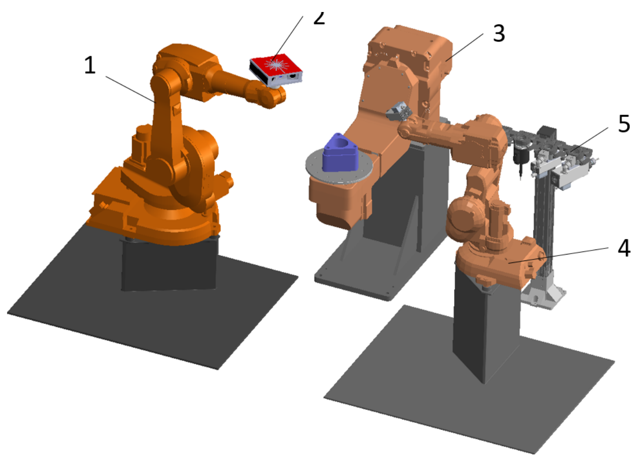


Figure 1: Robotic station design

The station consists of robot IRB140 (4 in Figure 1), fulfilling robotic ironworking actions utilising trajectory adapted automatically to the changing shape of the detail. The next element of the station is robot IRB1600 (1 in Figure 1) equipped with 3D optical scanner Atos Core 135 (2 in Figure 1). The scanner uses blue, structured light for scanning, which allows accurate measurement at different illumination. The series of photos is then converted to point cloud that can be used in later operations and measurements. Machined part is placed on a two-axis positioner, IRBPA250, and additional tools are stored on tool change system. It can store up to four different tools that can be mounted on a robot using pneumatic tool changer.

Conceptual design of the robotic station (Figure 2) shows connections, dependencies and data exchange between elements of the station. The measuring system cooperates with ATOS Professional software installed on PC and communicates by transferring measurement data to IRC5 robot controller using TCP/IP protocol. Other elements of the station include: two-axis positioner, IRBPA250 manipulating processed metal, spindles, tool changer, valve island, frequency converter, optic barriers and other security elements. RobotWare robot controller software ensures control of two robots and positioner and is additionally equipped with force control option. Grinding tool workspeed control is ensured by a variable frequency drive connected to the robot controller using DeviceNet protocol.

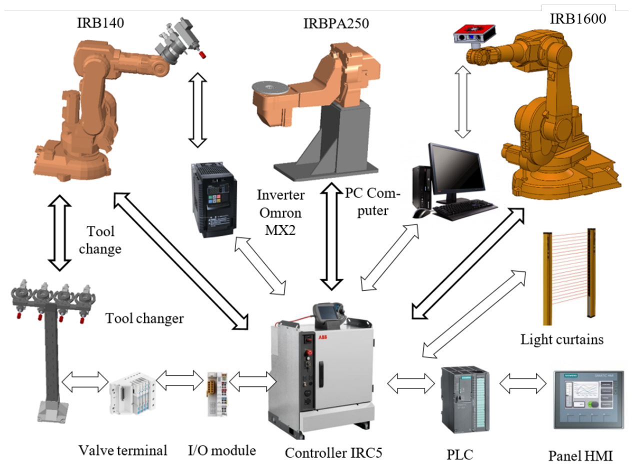


Figure 2: Conceptual design of communication between devices of the robotic centre

Figure 3 shows the flow of information in robotic machining in correlation with a 3D scanner.

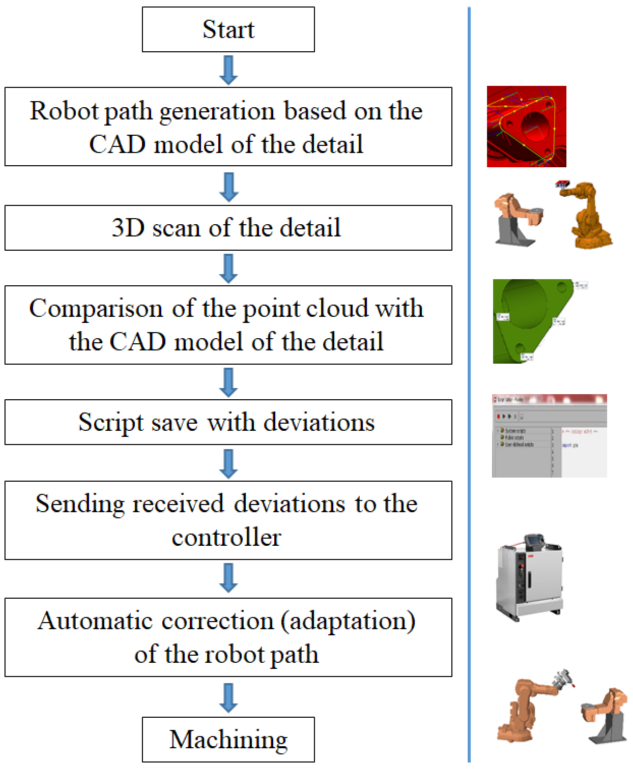
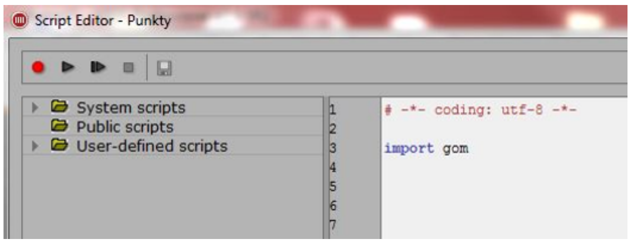

Start
Robot path generation based on the CAD model of the detail
3D scan of the detail
Comparison of the point cloud with the CAD model of the detail
Script save with deviations
Sending received deviations to the controller
Automatic correction (adaptation) of the robot path
Machining

Figure 3: Block diagram

3 GOM scripting

Having scanned the details before its processing in the Atos Professional software enables scripts creation based on commands recorder, which can record all performed actions in the software and save the record as a script in Python language (Figure 4). Thus, a recorded operation sequence may be performed multiple times. The recorded script may be adjusted to new tasks or generalised through editing. Script programming is based on modifying or combining records. Atos Professional environment software does not allow writing programs in the internal interface without using a record which is a basis of the whole script.



```
# -*- coding: utf-8 -*-

import gom
```

Figure 4: Creating scripts in Atos Professional

The first line (Figure 4) of the script editor contains recorder buttons for recording, starting and stopping of the recorded scripts. The left side of the window contains a catalogue of scripts which were pre-installed or defined by the user in the form of a tree. The context menu of this part consists of elements used to create new scripts, change its name and delete its elements, as in the case of standard documents. Scripts and script archives may also be exported and imported. Script catalogue is divided into three parts. The first part consists of pre-installed system scripts, which cannot be edited. The second part is a public folder, which is intended for sharing access to scripts by several uses. The last part contains private scripts of the logged in user. The largest part of the window is the editor, which is used for modifying the currently selected script. It also displays the script block structure. Besides standard operations (copy, paste, find and replace), the editor context menu contains also equivalents of recorder buttons. Menu elements enable more detailed modification of the script, for example, changing the position of registered operations in the script. The last part of the script editor is output range at the bottom of the window, where all results received during script performance appear.

Recorded script consists of prescribed functions automatically modified by the Atos Professional program. All parameters are defined and selected accordingly to the CAD model and point cloud got from scanning.

In the first line (Figure 5), the function is selected from the gom library. The gom library is divided into several layers. Thus, achieving create_multisection_by_parallel_planes function requires several additional parameters. An opened parenthesis at the end of the line informs of the beginning of invoking the function. It is followed by the parameters (one per line). Closing parenthesis marks the end of invoking the function. The result of a multi-sectional operation is bound to the MCAD_ELEMENT variable.

```
import gom

MCAD_ELEMENT=gom.script.section.create_multisection_by_pa
rallel_planes (
        direction_mode='symmetric',
        name='Section 1',
        position=0.00000000e+00,
        properties=gom.Binary ('eAHtV11o...'),
        reference_plane=gom.app.system['system_plane_x'],
        section_distance=5.00000000e+01,
        separated_elements=False)
```

Figure 5: An exemplary result of the recording process

The following data types are used as parameters values:

- character strings (direction_mode, name),
- whole numbers (num_sections),
- floating point numbers (position and section_distance) and
- Boolean values (separated_elements).

There is an option of changing all parameters, thanks to which a programmer can adjust the script to their needs.

4 Scripts recording and communication

In order to realise the grinding process, it is necessary to define material excess, which should be removed. Point cloud received as a result of the GOM scanning process was compared to a CAD model sample. Using geometrical actions on the point cloud, the largest excesses (distances) created due to comparison with the CAD model sample were determined (Figure 6).

A few characteristic points on point cloud were chosen (Figures 6 and 7). Shape of the perimeter of the point cloud

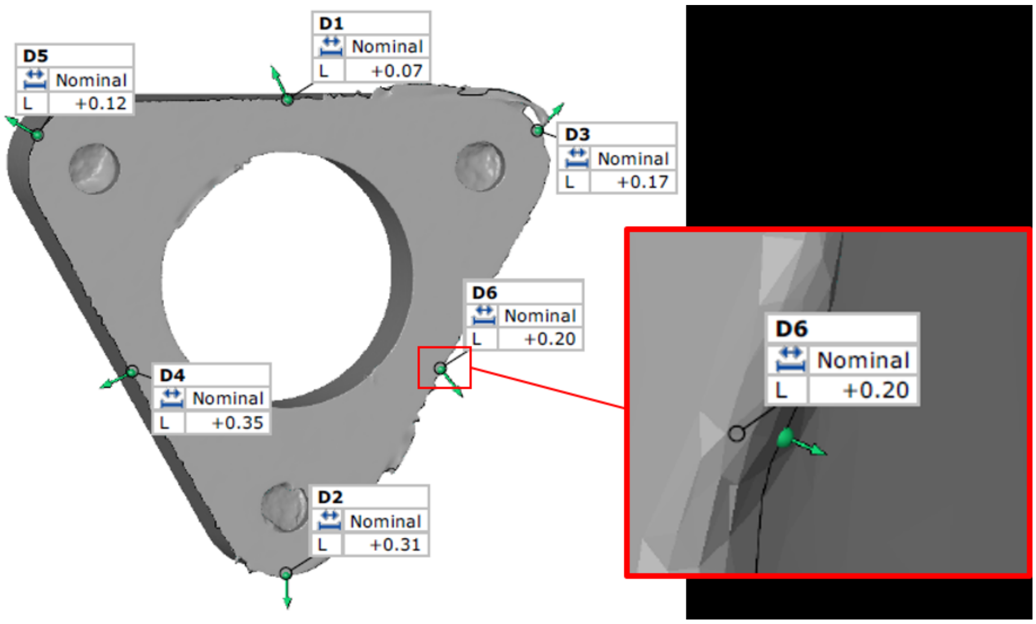


Figure 6: Characteristic points of a scanned part

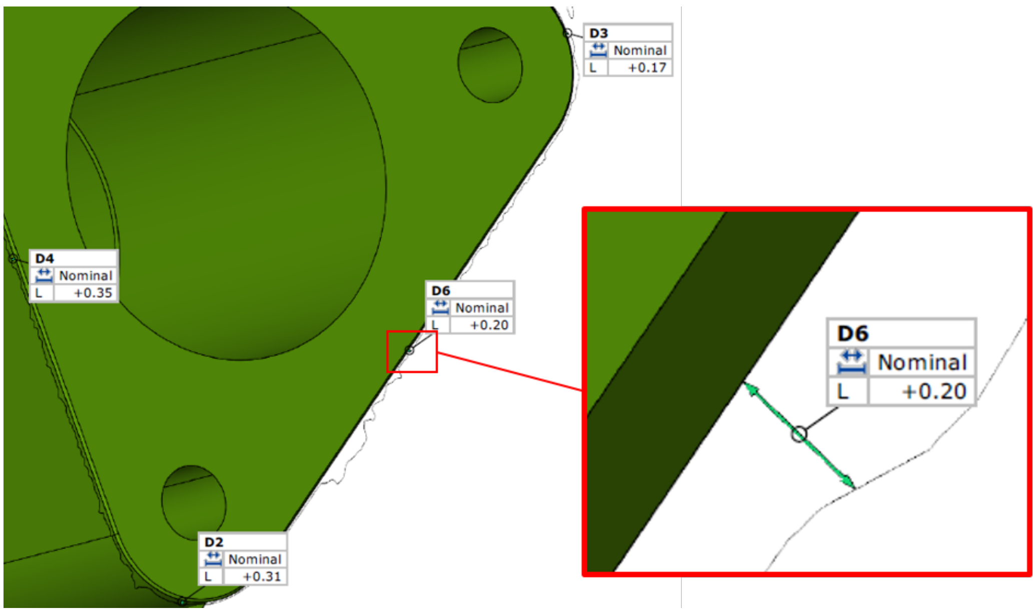
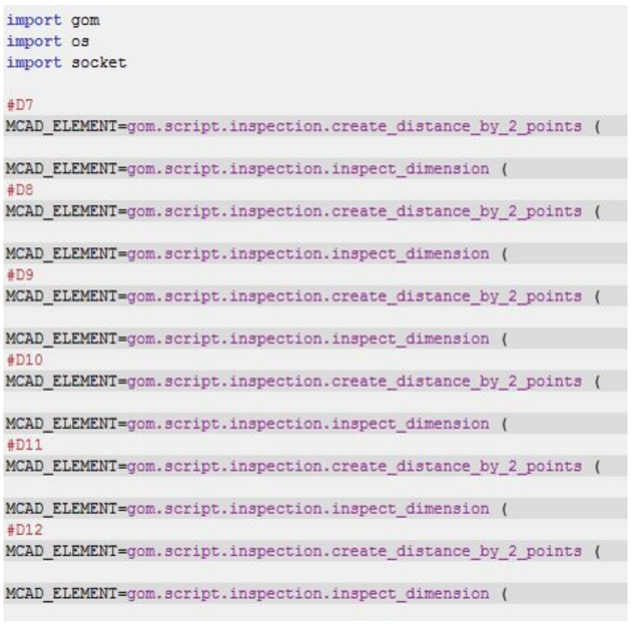


Figure 7: Measured excesses

was projected on a surface. In this way, the curve describing the scanned part and consisting of biggest deviations was defined. Then the CAD model and the received curve were compared and values of deviations were calculated.

Processing procedures were prepared this way for six points (Figures 6 and 7): D1, D2, D3, D4, D5, D6. They were recorded in the script using the recording options (Figure 8).

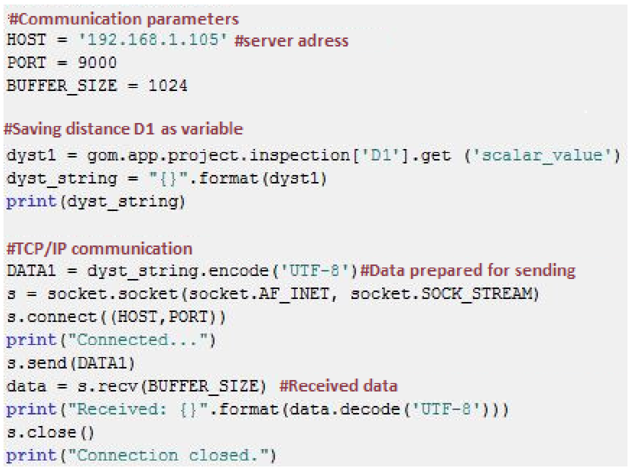
The first three lines of the script are responsible for adding Python language libraries: gom, os, socket. The gom library is built into and integrated with the Atos Professional environment. The user has no direct access to it. It may only be used by using recording options. It contains the functions used during recording and related to the elements of Atos Professional environment. The os library contains functions and options of the currently used operating system, and the socket library is responsible for the creation and usage of TCP/IP protocol. Further, script lines are the results of the recording action and perform actions on the point cloud in the software.



```
import gom
import os
import socket

#D7
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
#D8
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
#D9
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
#D10
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
#D11
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
#D12
MCAD_ELEMENT=gom.script.inspection.create_distance_by_2_points (

MCAD_ELEMENT=gom.script.inspection.inspect_dimension (
```

Figure 8: Script with recorded excesses

The next stage is sending received distances over to the robot controller (Figure 9). Communication parameters need to be defined: server address, port and buffer needed for the received data. Then, using the gom module function, the previously measured distances are downloaded and saved in a variable. It has to be converted into the byte format since the TCP/IP protocol does not allow sending other kinds of data. Then, communication with the robot controller is initiated and the prepared data is sent over. The last step is receiving confirmation from the robot controller that data was sent correctly. The data need to be converted from byte format back to the type usable for later use. After the transmission is finished, the connection is closed. Transferred data is processed, and then the robot's path is automatically adapted to the processed detail.



```
#Communication parameters
HOST = '192.168.1.105' #server adress
PORT = 9000
BUFFER_SIZE = 1024

#Saving distance D1 as variable
dyst1 = gom.app.project.inspection['D1'].get ('scalar_value')
dyst_string = "{}".format(dyst1)
print(dyst_string)

#TCP/IP communication
DATA1 = dyst_string.encode('UTF-8')#Data prepared for sending
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))
print("Connected...")
s.send(DATA1)
data = s.recv(BUFFER_SIZE) #Received data
print("Received: {}".format(data.decode('UTF-8')))
s.close()
print("Connection closed.")
```

Figure 9: Communication parameters definition

5 Conclusions

The article presents an original method of communication and data exchange in a robotic station. In the initial stage, the process was created in the virtual reality, that is, Robot Studio and Atos Professional software, according to the concept of Industry 4.0 which assumes that before the creation of the physical project, it will be possible to create its model in the virtual reality. It allowed to optimise the process and considerably decreased its costs. The virtual process was verified on a real station by conducting automatic scanning of a selected detail, then excesses' detection, script recording and sending it to the robot controller.

References

[1] Łygas K., Danilczuk W.: Pick'n'place application with the use of vision system and own communication interface (in Polish). *Autobusy: technika, eksploatacja, systemy transportowe*, 18, **2017**.

[2] Song Y., Liang W., Yang Y.: A method for grinding removal control of a robot belt grinding system. *Journal of Intelligent Manufacturing*, **23**(5), 1903-1913, **2012**.

[3] Burghardt A., Szybicki D., Kurc K., Muszyńska M., Mucha J.: Experimental Study of Inconel 718 Surface Treatment by Edge Robotic Deburring with Force Control. *Strength of Materials*, **49**(4), 594-604, **2017**.

[4] Jiang Y.F.: The Application of the TCP/IP on the Robot System. *Applied Mechanics and Materials* **159**, 351-354, **2012**.

[5] Kinder K.: Event-driven programming with Twisted and Python. *Linux journal*, **2005**(131): 6, **2005**.

[6] Tutak J. S.: Virtual reality and exercises for paretic upper limb of stroke survivors. *Tehnički vjesnik*, **24**(Supplement 2), 451-458, **2017**.

[7] Hui-ping L., Dai-min C., Miao Y.: Communication of Multi-robot System on the TCP/IP. In *Mechatronic Science, Electric Engineering and Computer (MEC)*, 2011 International Conference on, 1432-1435, **2011**.

[8] Latos H., Mikolajczyk T.: Surface shaping with industrial robot. *OPTIROB-2006*, Predeal, Romania, University "POLITEHNICA" of Bucharest, Faculty IMST, Department MSP, 265-269, **2006**.

[9] Gierlak P., Burghardt A., Szybicki D., Szuster M., Muszyńska M.: On-line manipulator tool condition monitoring based on vibration analysis. *Mechanical Systems and Signal Processing*, **89**, 14-26, **2017**.

[10] Sempere V. M., Silvestre J.: Multimedia applications in industrial networks: Integration of image processing in profibus. In *IEEE Transactions on Industrial Electronics*, **50**(3), 440-448, **2003**.

[11] Valera A., Salt J., Casanova V., Ferrus S.: Control of industrial robot with a fieldbus. In *Emerging Technologies and Factory*

Automation, 1999. Proceedings. ETFA'99. 1999 7th IEEE International Conference on, Vol. 2, 1235-1241, **1999**.

[12] Zolkiewski S., Pioskowik D.: Robot control and online programming by human gestures using a kinect motion sensor. In *New Perspectives in Information Systems and Technologies*, Volume 1, 593-604, Springer, Cham **2014**.

[13] Burghardt A., Kurc K., Szybicki D., Muszyńska M., Nawrocki J.: Software for the robot-operated inspection station for engine guide vanes taking into consideration the geometric variability of parts. *Technical Gazette*, **24**(2), 349-353, **2017**.

[14] Burghardt A., Kurc K., Szybicki D., Muszyńska M., Nawrocki J.: Robot-operated quality control station based on the UTT method. *Open Engineering*, **7**(1), 37-42, **2017**.

[15] Burghardt A., Kurc K., Szybicki D., Muszyńska M., Szczęch T.: Monitoring the parameters of the robot-operated quality control process. *Advances in Science and Technology - Research Journal*, **11**(1), 232-236, **2017**.

[16] Li W. L., Xie H., Zhang G., Yan S. J., Yin Z. P.: 3-D shape matching of a blade surface in robotic grinding applications. *IEEE/ASME Transactions on Mechatronics*, **21**(5), 2294-2306, **2016**.

[17] Piotrowski A.: An Analysis of the use of the Python Language in Robot Applications. *Applied Computer Science*, **12**(2), **2016**.

[18] Lutz M.: *Programming python*. O'Reilly, **1996**.

[19] Python 3.4.3 Documentation

[20] Rhodes B., Goerzen J., Beaulne A., Membrey P.: *Foundations of Python network programming*. Apress, **2014**.

[21] Python Scripting for GOM Applications, documentation

[22] Mizugaki Y., Sakamoto M., Kamijo K., Taniguchi N.: Development of metal-mold polishing robot system with contact pressure control using CAD/CAM data. *CIRP Annals-Manufacturing Technology*, **39**(1), 523-526, **1990**.

[23] Mikołajczyk T.: Robot application to surface finish machining. *J Polish CIMAC*, **5**(3), **2010**.

[24] Tam H. Y., Lui O. C. H., Mok A. C.: Robotic polishing of free-form surfaces using scanning paths. *Journal of Materials Processing Technology*, **95**(1-3), 191-200, **1999**.

[25] Zhao Y., Zhao J., Zhang L., Qi L.: Development of a robotic 3D scanning system for reverse engineering of freeform part. In *Advanced Computer Theory and Engineering*, 2008. ICACTE'08. International Conference on, 246-250, **2008**.

[26] Sun B., Li B.: Laser displacement sensor in the application of aero-engine blade measurement. *IEEE Sensors Journal*, **16**(5), 1377-1384, **2016**.

[27] Qi L., Gan Z., Yun C., Tang Q.: A novel method for Aero engine blade removed-material measurement based on the robotic 3D scanning system. In *Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, 2010 International Conference on, Vol. 4, 72-75, **2010**.

[28] Yilmaz O., Gindy N., Gao J.: A repair and overhaul methodology for aeroengine components. *Robotics and Computer-Integrated Manufacturing*, **26**(2), 190-201, **2010**.

[29] Burghardt A., Kurc K., Szybicki D., Muszyńska M., Szczęch T.: Robot-operated inspection of aircraft engine turbine rotor guide vane segment geometry. *Technical Gazette*, **24**(2), 345-348, **2017**.