Applied Computer Science, vol. 14, no. 1, pp. 73–83
doi:10.23743/acs-2018-07

Submitted: 2018-02-15
Revised: 2018-03-15
Accepted: 2018-03-25

industrial robots, robots programming, AS language, MATLAB

Anna CZARNECKA[*]*, Łukasz SOBASZEK*[*]*, Antoni ŚWIĆ*[*]

2D IMAGE-BASED INDUSTRIAL ROBOT END EFFECTOR TRAJECTORY CONTROL ALGORITHM

Abstract

This paper presents an algorithm for programming an industrial robot's end effector path based on 2D images. The first section gives a brief overview of modern solutions for industrial robot implementation. The next section describes the test set-up and the software used in tests. The work also presents the key elements of the controller algorithm and their operation: 2D image processing with MATLAB software, generating the code for robot control in AS language, and implementation of the produced codes to the Kawasaki RS003N robot.

1. INTRODUCTION

Automation in manufacturing and production processes would be virtually impossible without implementation of robots. Robots have become a common solution in a wide range of industrial processes, and have therefore contributed to boosting the efficiency and flexibility of production, increasing work safety, allowing to achieve high quality of production at higher reliability and lower cost (Hajduk & Koukolová, 2015).

Robots replace human employees in harmful working environments, where highly qualified personnel is scarce. Robots perform a wide range of jobs including: welding, painting, lacquering, adhesive joining or distribution of industrial agents, as well as in auxiliary processes, such as cleaning, polishing or grinding (Hajduk, Jenčik, Jezný, & Vargovčik, 2013).

[*] Lublin University of Technology, Faculty of Mechanical Engineering, Institute of Technological Systems of Information, Nadbystrzycka 36, 20-618 Lublin, tel.: +48 81 538 45 35, l.sobaszek@pollub.pl, a.swic@pollub.pl

In addition to typical applications, robots are implemented to perform other jobs, primarily oriented towards assistance in production. A prominent example of such auxiliary jobs is found in logistics, where the concept of autonomous robots building automated transportation systems has been the subject of much systematic investigation and development, particularly with regard to transporting tools, components and subassemblies. Automated robot are also applied in distribution centres to transport and sort goods ("W fabryce Audi", n.d.). Human-robot interaction at a single workstation is yet another scenario of robot implementation, which is a growing trend in industrial processes (Sobaszek, Gola & Varga, 2016; Sobaszek, Gola & Świć, 2017).

One of the most recent developments in robotics is application of industrial robots as machine tools, performing basic machining jobs. There is a marked trend towards implementing industrial robots (equipped with necessary instrumentation) in machining processes in which the regular CNC machine tools are inadequate, owing to intricate geometry or big size of workpieces (Riexinger Information, n.d.).

This section attempted to give a brief summary of robotic applications in the industry. It becomes apparent that the range of jobs that may be performed by automated robots is constantly increasing. Implementation of robots in industrial processes may require defining the trajectory of robot end effector paths in order to perform the jobs. This process is burdened with considerable time expense and demands various analytical works. Therefore, manufacturers offer different solutions aimed at facilitating and accelerating the process of defining trajectory paths, including employing: teaching through demonstration (Haage, Piperagkas, Papadopoulos, Mariolis, Malec, Bekiroglu, Hedelind & Tzovaras, 2017), neural networks (Jin, Li, Yu & He, 2018), 3D vision systems (Wan, Lu, Wu & Harada, 2017), or machine learning (Xu, Yang, Zhong, Wang & Zhao, 2018). The solution introduced in this paper is a 2D image-based algorithm for controlling an industrial robot, which constitutes an alternative solution to classic programming.

2. RESEARCH OBJECTIVES

We attempted to develop an algorithm that would execute the motion of an industrial robot's end effector along the trajectory defined by means of a 2D image. The proposed solution should improve the work of the robot operator and shorten the programming time. Furthermore, the proposed algorithm would be capable of generating the trajectory path of the robot from the technological documentation (generated with CAD software) and would find application in such processes as: welding, soldering, adhesive joining, trimming or more complex machining.

The algorithm was executed and implemented by means of MATLAB software and used commands derived from the AS programming language. Fig. 1 below presents an overview of the developed algorithm.

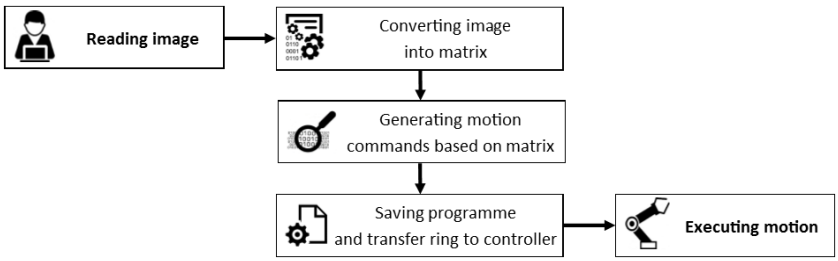

Fig. 1. General overview of the algorithm

3. DESCRIPTION OF THE WORKSTATION

The workstation conditions were simulated by the laboratory test set-up with the Kawasaki RS003N industrial robot. The robot was fitted with an appropriate end effector, a gripper, which enabled verification of the developed solution (Fig. 2).

Fig. 2. Laboratory test-set up with Kawasaki RS003N robot

Kawasaki RS003N robot is a 6-axis industrial robot, programmable with the Teach Pendant, dedicated K-ROSET robot simulation software and AS programming language. The Teach Pendant is an LCD controller that allows to programme and operate the robot, either through execution of AS language commands or by point-to-point teaching. K-ROSET is an advanced programming tool allowing to program robots and perform simulations of robot arm trajectories, analyse cycle-time, check for collision, and verify robot layout. AS is a code developed

by Kawasaki and implemented in the K-ROSET software as a set of commands for programming robot motion, signal handling, gripper operation, etc. (Grobelny, Jarosiński, Piłat, Pieniak & Sobaszek, 2016). The robot motion control is performed by means of the controller, which is additionally responsible for communicating with the robot through special communication protocols. The robot and the controller specifications are shown in Table 1.

Tab. 1. Robot and controller specifications (Kawasaki Heavy Industries, 2010)

Robot specifications	Controller specifications
Model: RS003N-A Type: articulated robot Degree of freedom: 6 Repeatability: ±0,05 mm Maximum load: 3 kg Maximum speed: 6000 mm/s Driving motor: brushless AC servo motor	Model: E70/E71 Structure: enclosed Number of controlled axes: 6 Programming: TEACH/REPEAT Teaching: AS language Power requirements: 200–240 V

4. PROGRAMMING TOOLS

The proposed algorithm was developed and verified with the application of the following programming tools:

- MATLAB software,
- AS programming code.

MATLAB software has a wide range of applications, and may be employed to perform numerical computations, simulations, data visualisation and analysis or image processing. MATLAB is a programming language applied for higher-order programming, and may be applied in works on matrixes, vectors and structures. This software package is applicable in mathematical computations, numerical algorithms, modelling and simulation, data analysis, visualisation of results, engineering graphics and creating functional applications (Fig. 3). In the programming works performed in the study, the following elements of MATLAB package were employed:

- MATLAB scripting language – executed subsequent steps of the algorithm,
- working environment – where the m-files with commands were created and managed,
- selected libraries – performing complex operations on images and matrixes.

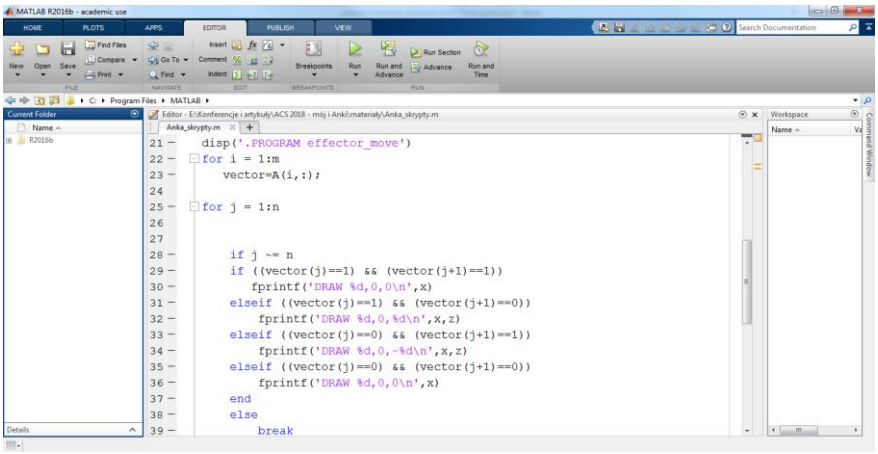

Fig. 3. MATLAB software used in development of algorithm

Programming robots with the AS language resembles classical high-level programming. By means of a series of commands (processed by the controller) the programmer determines the sequence of robot motions. Similarly to other programming languages, the data takes the form of: constant, local variable or global variable. The AS language also makes use of standard data types: numeric expression, combination of alphanumeric characters strings in ASCII, and logical values. The instructions in AS language include commands for programme execution, defining robot locations, motions, velocities, accuracy of robot motion and tool (Kawasaki Heavy Industries, 2010). Fig. 4 shows an example of instructions in AS.

```
.PROGRAM point1()
JMOVE #start
JAPPRO #p001,50
JMOVE #p001
CLAMP 1
.END
```

Fig. 4. A part of code written in AS

The tools used in the research works have been properly integrated, which consisted in generating codes controlling the robot in AS with the application of MATLAB software. The conducted works required employing available image conversion tools, matrix operation instructions, and elements of structural programming. A detailed description of the algorithm will be presented in section 5.

5. DESCRIPTION AND EXECUTION OF ALGORITHM

Each main part of the algorithm, presented in section 2, is composed of a finite number of steps. Each step has its role and functionality. For the purpose of clarity, the developed algorithm is presented in block diagram, whose subsequent steps were executed by means of programming tools (Fig. 7).

First, the algorithm reads the image to be drawn by the robot end effector. Next the image is converted to binary image by means of *im2bw* function. The output bi-level image takes the form of a matrix of elements 0 and 1, which will provide the base for generating the commands of robot motion (Fig. 5).

```
gfx=imread('face.jpg');
A=im2bw(gfx);
[m,n] = size(A);
imshow(A);
```

Fig. 5. Reading and conversion of image

The next stage consists in preparing the environment – the main functionality used here is *diary* function, which allows to save commands generated in AS language as *.pg files. This is a standard file recognised by Kawasaki robots. The subsequent lines of the programme define the values of robot motions (the distance between pixels). This declaration of motion steps enables convenient scaling of the image created by the robot's arm. The distances are specified for each axis (X, Y and Z) (Fig. 6).

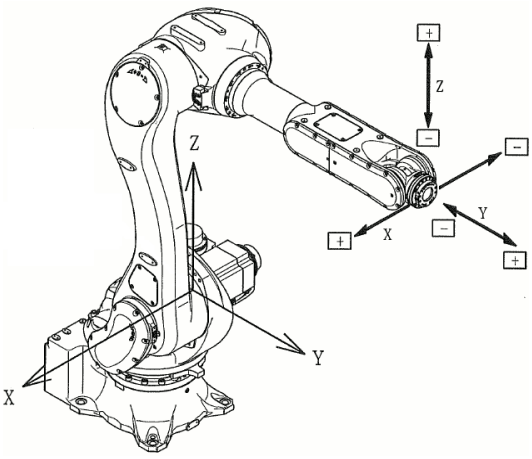

Fig. 6. The robot effector motion options in execution of algorithm (Kawasaki Heavy Industries, 2010)

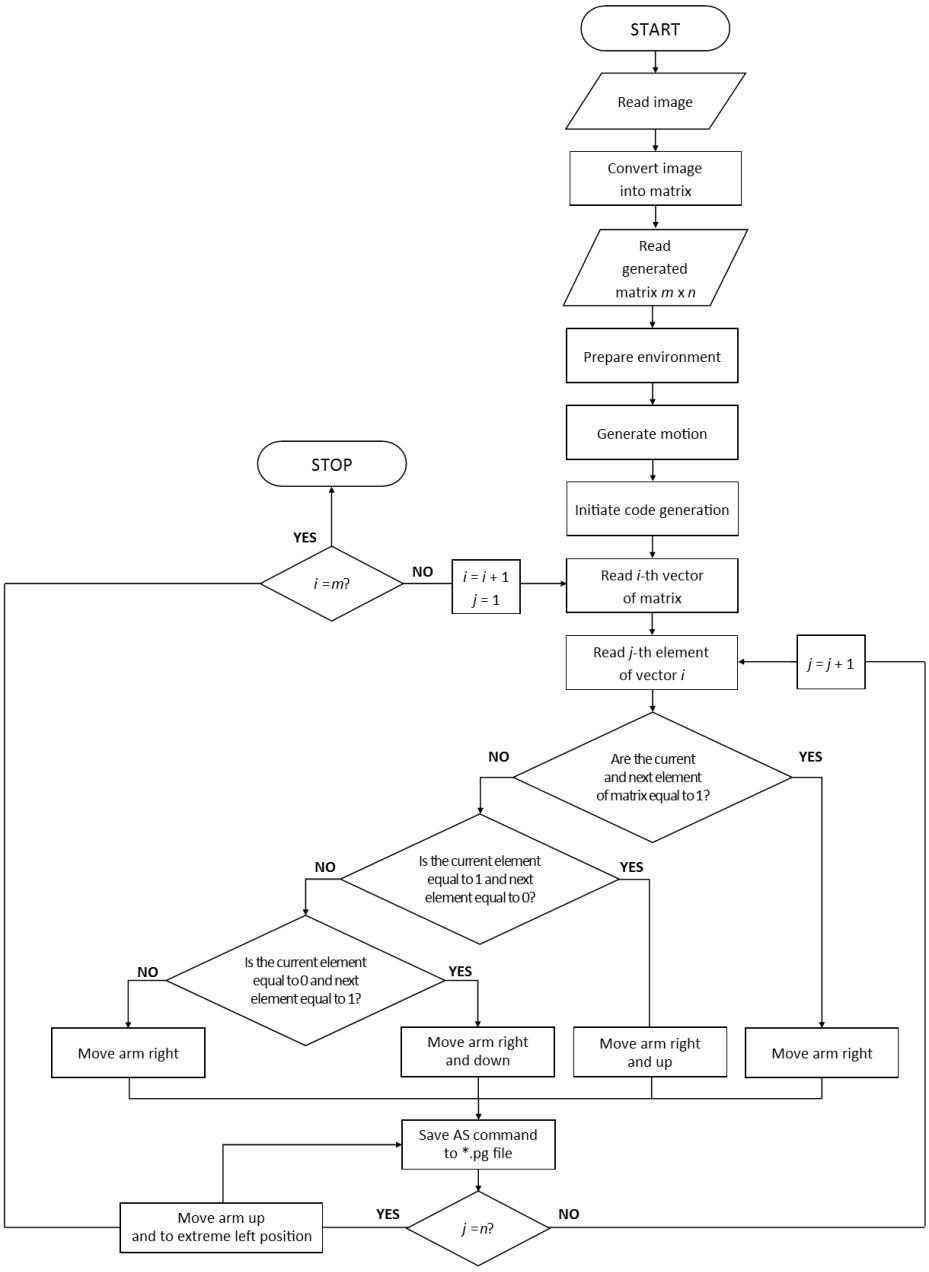

Fig. 7. Block diagram of the Kawasaki industrial robot control algorithm

Once the image is converted and the environment prepared, the fundamental part of the algorithm may commence, *i.e.* generation of robot motion commands. The code is generated iteratively, and therefore includes two *for loops* of specified values. The values depend on the size of the generated matrix (the number of columns and rows). The first loop was responsible for selecting a single matrix row (working vector), and the other one for analysing the values of elements in the working line (Fig. 8).

```
for j = 1:n
if j ~= n
    if ((vector(j)==1) && (vector(j+1)==1))
       fprintf('DRAW %d,0,0\n',X)
    elseif ((vector(j)==1) &&
(vector(j+1)==0))
        fprintf('DRAW %d,0,%d\n',X,Z)
    elseif ((vector(j)==0) &&
(vector(j+1)==1))
        fprintf('DRAW %d,0,-%d\n',X,Z)
    elseif ((vector(j)==0) &&
(vector(j+1)==0))
        fprintf('DRAW %d,0,0\n',X)
    end
else
    break
end
end
```

Fig. 8. Code generating AS instructions – loop analysing elements of the working vector

The analysis of matrix elements consists in checking the current and next element of the matrix. The conditional instruction *if* executes rules that generate instructions in AS language. If the first condition is met, the algorithm generates AS code command, „DRAW X,0,0", which moves the robot right by a previously defined value X. If the condition is not met, subsequent conditions are analysed. The condition which is fulfilled moves the end effector right and up or right and down. The procedure is identical for analysing all elements of the matrix.

The key element of the proposed algorithm is analysing the last element of the working line (Fig. 9). This step is aimed to execute correct pass of the end effector to the first element of the following line. The value of robot end effector motion is determined, and the logical value of the last element of the working line is checked. If the last element is 0, the programme generates instruction „DRAW –X,–Y,0", which causes the robot's arm to move left by *x* and down by *y*, otherwise the generated instruction takes the form of „DRAW 0,0,Z", lifting the arm of the robot by Z, and the instruction „DRAW –X,–Y,0", moving the arm of the robot to the next line.

```
if i ~= m
   Xr=n*X-X;

    if A(i,n)==0
        fprintf('DRAW -%d,-%d,0\n',Xr,Y)
    elseif A(i,n)==1
        fprintf('DRAW 0,0,%d\n',Z)
        fprintf('DRAW -%d,-%d,0\n',Xr,Y)
    end
    if (A(i+1,1)==1)
        fprintf('DRAW 0,0,-%d\n',Z)
    end
    X=x;
else
    break
end
```

Fig. 9. Code determining the pass of the end effector to the subsequent line (vector)

The execution of the algorithm is exported to a *.pg file. After generating, the whole code is sent to the robot's memory by means of KCwinTCP terminal, for remote communication with the robot terminal. The terminal enables executing the code generated by the developed script. Upon completion, the robot end effector motion error verification procedure is possible.

6. VERIFICATION OF ALGORITHM

The developed algorithm was tested in order to verify its efficiency and accuracy. The verification process was twofold:

1. Stage I – draw an image based on a 5x7 test matrix.
2. Stage II – generate end effector trajectory path from a 2D image.

At the first stage the robot was commanded to reflect the paths according to the randomly-generated matrix. In addition, the verification included such areas as the error of motion in X and Y axis and the height Z to which the robot arm was elevated if the matrix values were changed. The results of verification are shown in Fig. 10. Having successfully conducted stage I, stage II of verification was aimed to determine the execution of robot end effector motion based on a 2D image.

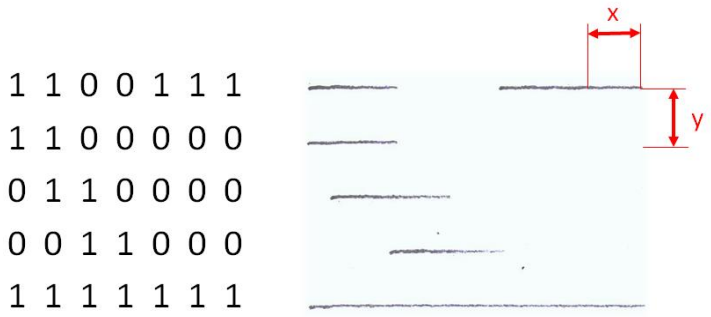

Fig. 10. Model matrix and its execution by the robot

The second stage of verification consisted in analysing the entire algorithm and its execution with the generated code. The selected image was converted into a matrix and drawn by means of the Kawasaki RS003N robot. The results of algorithm execution are shown in Fig. 11.

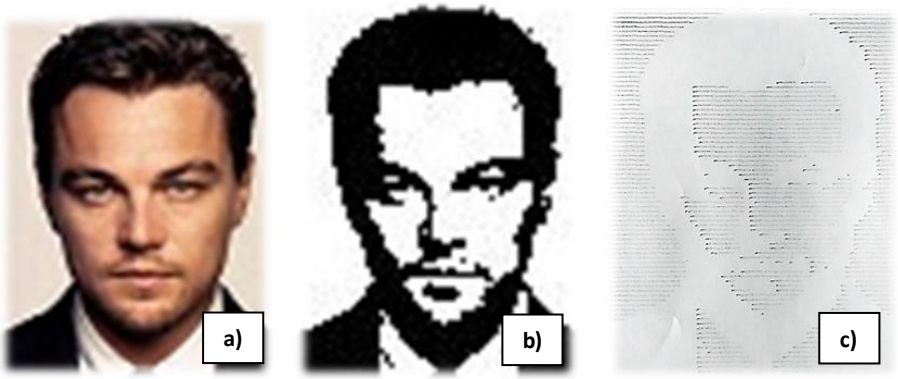

Fig. 11. Verification of image-processing algorithm: a) real image, b) binary image, c) image drawn by the robot – image source: (Sample image, 2014)

Clearly, the proposed algorithm has proven to be functional. The only element of the code that requires further development is the code generation mode, due to the reason that the image is processed into negative. It is, therefore, necessary to implement a special function executing appropriate conversion of the matrix generated from the 2D image.

7. SUMMARY AND FUTURE WORKS

The presented works are in keeping with the current trends towards designing applications for controlling industrial robots and thus implementing robots in applications which require defining end effector trajectory path, which may be of a high degree of complexity. Although the algorithm is operational, future works should concentrate on improving the script-generating code in order to enhance image processing, and, secondly, to generate commands controlling the motion in the vertical plane and along curves. Further works should be carried out in order to enable implementing images made in popular CAD tools. That would open the gate to applying the algorithm to such processes as trimming, soldering or adhesive joining.

REFERENCES

Grobelny, M., Jarosiński, S., Piłat, M., Pieniak, D., & Sobaszek, Ł. (2016). Projekt aplikacji komputerowej umożliwiającej sterowanie robotem przemysłowym kawasaki RS003N. *Zeszyty Naukowe Wydziału Elektroniki i Informatyki*, 10, 163–176.

Haage, M., Piperagkas, G., Papadopoulos C., Mariolis I., Malec J., Bekiroglu Y., Hedelind M., & Tzovaras D. (2017). Teaching Assembly by Demonstration Using Advanced Human Robot Interaction and a Knowledge Integration Framework. *Procedia Manufacturing*, 11, 164–173. doi:10.1016/j.promfg.2017.07.221

Hajduk M., & Koukolová L. (2015). Trends in Industrial and Service Robot Application. *Applied Mechanics and Materials*, 791, 161–165. doi:10.4028/www.scientific.net/AMM.791.161

Hajduk, M., Jenčik, P., Jezný, J., & Vargovčik, L. (2013). Trends in industrial robotics development. *Applied Mechanics and Materials*, 282, 1–6. doi:10.4028/www.scientific.net/AMM.282.1

Jin L., Li, S., Yu, J. & He, J. (2018), Robot manipulator control using neural networks: A survey. Neurocomputing, 285, 23–34. doi:10.1016/j.neucom.2018.01.002

Kawasaki Heavy Industries. (2010). *AS Language Programming*.

Kawasaki Heavy Industries. (2010). *Kawasaki Robots User Manual*.

Riexinger Information. (n.d.). Retrieved May 20, 2017, from https://riex.de/automatisierung/roboterfraesanlage

Sample image [online image]. (2014). Retrieved May 20, 2017, from http://www.boomsbeat.com/articles/1875/20140327

Sobaszek, Ł., Gola, A., & Świć A. (2017). Kierunki rozwoju robotyki w aspekcie projektowania współczesnych systemów produkcyjnych. In R. Knosala (Eds.), *Innowacje w zarządzaniu i inżynierii produkcji* (pp. 460–471). Opole: Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją.

Sobaszek, Ł., Gola, A., & Varga, J. (2016). Virtual designing of robotic workstations. *Applied Mechanics and Materials*, 844, 31–37. doi:10.4028/www.scientific.net/AMM.844.31

W fabryce Audi roboty transportują samochody. (n.d.). Retrieved March 15, 2018, from Audi Autorund Website, http://autorud.pl/audiblog/w-fabryce-audi-roboty-transportuja-samochody

Wan, W., Lu, F., Wu, Z., & Harada, K. (2017). Teaching robots to do object assembly using multi-modal 3D vision. *Neurocomputing*, 259, 85–93. doi:/10.1016/j.neucom.2017.01.077.

Xu, Y., Yang, Ch., Zhong, J., Wang, N., & Zhao, L. (2018). Robot teaching by teleoperation based on visual interaction and extreme learning machine. *Neurocomputing*, 275, 2093–2103. doi:10.1016/j.neucom.2017.10.034