Mechanical Engineering
Машиноведение

DOI: 10.17516/1999-494X-0445

УДК 681.5.01

Adaptive Control System for Automated Honeycomb Filler Cutter

Iakov Yu. Pikalov,
Nikolai V. Shtabel and Mikhail V. Saramud*
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Federation

Received 27.09.2022, received in revised form 05.10.2022, accepted 21.11.2022

Abstract. This article presents the results of control system development of an automated complex for cutting honeycomb or cellular filler, which consists of several modules performed on various software and hardware platforms and combined into a single system. On the control computer, the following are implemented: an interface module, a planning and control module written in the C ++, and a machine vision module written in the Python. The KUKA controller has a module for communication with a computer and a movement module written using the Kuka Robot Language. The interface module is required to interact with the operator, the planning and control module, and the robot information exchange module. The planning and control module performs the functions of generating sequence of elementary operations and moving points, compiling control programs and executing them. The machine vision module, using a machine vision camera, takes a series of images of the workpiece, by which the boundaries of the workpiece are identified, the edges of the cellular filler are recognized, and the coordinates of the optimal cut points are calculated.

Keywords: honeycomb filler, control system, process cell, cutting.

Citation: Pikalov, Ia. Yu., Shtabel, N.V., Saramud, M. V. Adaptive control system for automated honeycomb filler cutter. J. Sib. Fed. Univ. Eng. & Technol., 2022, 15(8), 1009–1019. DOI: 10.17516/1999-494X-0445

* Corresponding author E-mail address: msaramud@gmail.com

Адаптивная система управления автоматизированным резаком для сотового наполнителя

Я. Ю. Пикалов, Н. В. Штабель, М. В. Сарамуд
Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнёва
Российская Федерация, Красноярск

Аннотация. В данной статье представлены результаты разработки системы управления автоматизированным комплексом резки сотового заполнителя, который состоит из нескольких модулей, выполненных на различных программно-аппаратных платформах и объединенных в единую систему. На управляющем компьютере реализованы: модуль интерфейса, модуль планирования и управления, написанный на языке C++, и модуль машинного зрения, написанный на языке Python. Контроллер KUKA имеет модуль для связи с компьютером и модуль движения, написанный на языке роботов Kuka. Интерфейсный модуль необходим для взаимодействия с оператором, модулем планирования и управления и модулем обмена информацией о роботе. Модуль планирования и управления выполняет функции формирования последовательности элементарных операций и перемещения точек, составления управляющих программ и их выполнения. Модуль машинного зрения с помощью камеры машинного зрения делает серию снимков заготовки, по которым идентифицируются границы заготовки, распознаются края ячеистого заполнителя, рассчитываются координаты оптимальных точек среза.

Ключевые слова: сотовый наполнитель, система управления, технологическая ячейка, резка.

Цитирование: Пикалов Я. Ю. Адаптивная система управления автоматизированным резаком для сотового наполнителя / Я. Ю. Пикалов, Н. В. Штабель, М. В. Сарамуд // Журн. Сиб. федер. ун-та. Техника и технологии, 2022, 15(8). С. 1009–1019. DOI: DOI: 10.17516/1999-494X-0445

1. Introduction

In modern production of satellite spacecraft non-hermetic load-bearing structures with the use of composite elements based on cellular fillers are widely used [1, 2, 3]. Since the filler itself has a thin-walled structure with low bending stiffness, traditional milling is performed with poor quality (cell crushing, uneven cut, impossibility of forming sharp internal corners in the selected cavities) [4, 5].

2. The proposed complex for cutting honeycomb filler

The authors propose a different method: cutting out the required configuration of the cellular filler using an actuator with a vertically descending knife, sequentially cutting along one face of the cellular filler. In this case, the bending load on the thin walls is minimal. Since the width of the knife is commensurate with the width of one face of the cell (5–10 mm), it becomes necessary to accurately position the knife relative to the actual structure of the cells (± 0.25 mm). A computer vision system and image processing algorithms are used to perform this precise positioning. The precise movement of the actuator in space is performed by the KUKA KR 6 R 900 robotic arm [6].

A general view of the complex for cutting honeycomb filler is shown in Fig. 1.

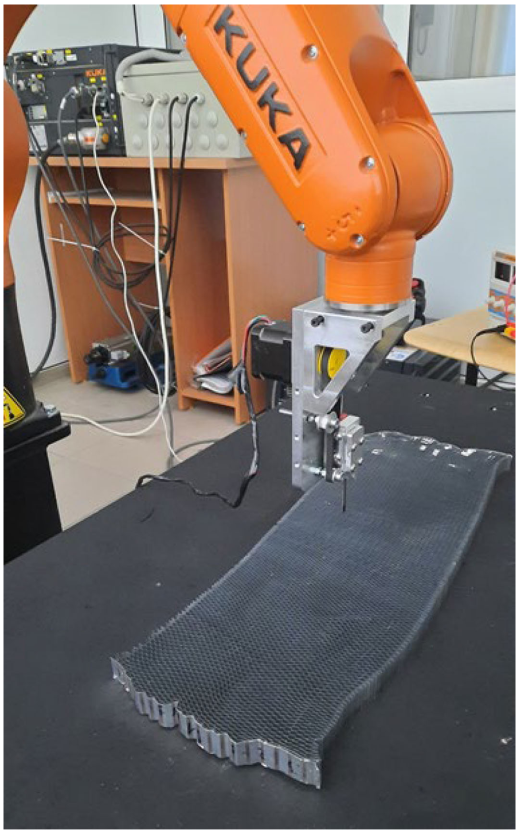

Fig. 1. General view of the complex for cutting honeycomb filler

3. Algorithm for cutting honeycomb filler

In general, the operation of the algorithm for cutting a given shape in cellular filler can be represented as:

1. Setting and fixing the workpiece on the robot table (performed by the operator);

2. Determination of the size, shape of the workpiece and its position on the table;

2.1. A general snapshot of the entire table surface is taken (at the command of the operator through the program interface);

2.2. Recognition of the workpiece contour and determination of the coordinates of the contour points with an accuracy of ± 2 mm (using vision algorithms);

3. Drawing up a task for the robot (at the command of the operator through the program interface):

3.1. Loading cut out contours from external files;

3.2. The location of the cut out contours on the workpiece;

4. Planning trajectories of movement of the actuator for taking pictures (performed in automatic mode);

5. Determination of the point of one cut (performed in automatic mode);

5.1. Taking a snapshot at a point along the trajectory planned in step 4;

5.2. Image processing with computer vision algorithms to determine the edges of the honeycomb filler with an accuracy of ± 0.25 mm (Fig. 2, a);

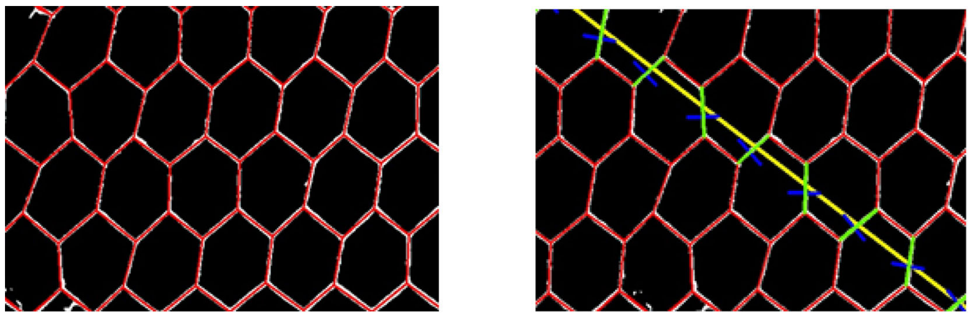

Fig. 2. Determination of the edges of the cellular filler (a) and optimal cutting points (b)

5.3. Determination of the vertices and edges on the cellular filler to be cut;

5.4. Calculation of the coordinates of the optimal cutting points of the edges and the angle of rotation of the knife relative to the Z-axis (Fig. 2, b);

6. Execution of a cut of one face according to the coordinates specified in item 5 (performed in automatic mode);

7. Repetition of p. 5 and p.6.

The adaptive control in this algorithm consists in correcting the cutting points based on the new position of the workpiece resulting from the shift and deformations of the filler cells under the action of the knife cutting forces. The new position is determined by the snapshot taken after the next cut. Thus, regardless of the changed position of the workpiece, new cutting points will be determined with an accuracy of ± 0.25 mm from the actual position of the workpiece.

4. Description of the hardware and software of the complex

The functional diagram of the implemented hardware and software complex for cutting cellular filler of honeycomb panels is shown in Fig. 3.

The complex consists of a control computer, a robotic manipulator, and an operator controlling the complex and performing the installation of the workpiece on the table and removal of the cut filler.

There are three software modules on the control PC:

1. Interface module is required to interact with the operator (via channel 1 in Fig. 3) and exchange information with the planning and control module (channels 2 and 3) and the robot's information exchange module (channels 4 and 5);

2. The planning and control module performs the functions of planning the technological process (drawing up a general sequence of elementary operations, as well as calculating the coordinates of the points of the robot's trajectory, taking into account corrective displacements). The necessary calculations are performed on the basis of information received through channels 2 (from the interface module) and 7 (from the machine vision module), control actions are transmitted through channels 3 (to the interface module), 6 (to the machine vision module) and 8 (to the knife drive).

Machine vision module – at the command received from the planning and control module (via channel 6), using the machine vision camera, photographs the workpiece (or its area) by sending a signal (via channel 9). The image obtained from the machine vision camera (via channel 10) is processed (the

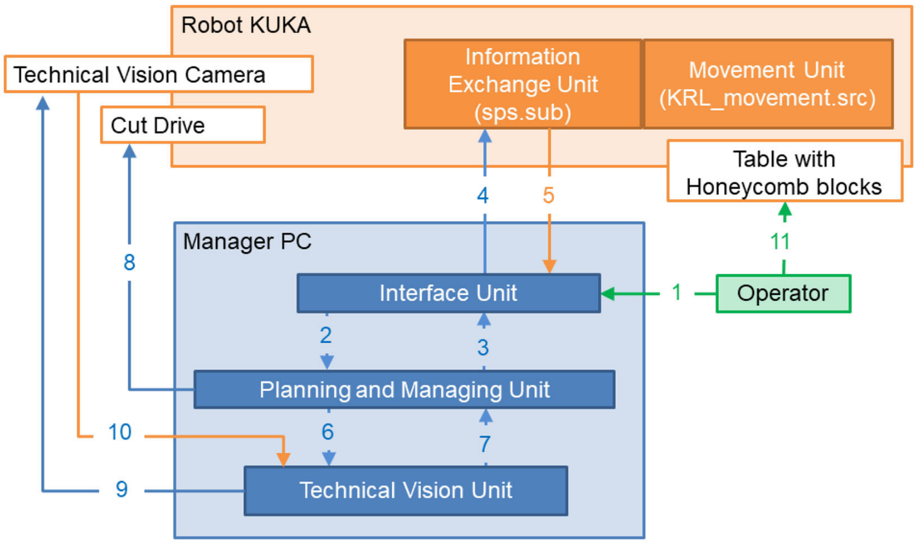
Robot KUKA
Technical Vision Camera
Cut Drive
Information Exchange Unit (sps.sub)
Movement Unit (KRL_movement.src)
Table with Honeycomb blocks
4
5
11
Manager PC
8
Interface Unit
1
Operator
2
3
Planning and Managing Unit
10
6
7
9
Technical Vision Unit

Fig. 3. Functional structure of the hardware-software complex "sotorezchik"

boundaries of the workpiece are identified, the edges of the cellular filler are recognized, and the coordinates of the optimal cutting points are calculated). The coordinates of the optimal cutting points are transmitted through channel 7 to the planning and control module.

The robot has two software modules that implement the required functionality using standard robot controls.

1. Module for exchanging information with a PC – designed to receive data on the necessary planned movements (via channel 4) and send data about the state of the robot (via channel 5). This module is implemented as a separate program that cyclically runs in the background.

2. Movement module is designed for the execution of planned movements and is implemented in the form of the main robot program, also executed in a cyclic mode.

The communication channels designated by numbers in Fig. 3 exchange using various protocols and interfaces.

Communication channels 1 and 11 – implement the interaction of the operator with the control PC, through the program interface and with the robot when installing the workpiece and removing the finished product.

Communication channels 2, 3 are implemented in the form of software communication between the modules of the control program. Channels 6 and 7 are organized through local TCP / IP communications.

Channels 4 and 5 are implemented using an Ethernet connection between the control PC and the robot control cabinet.

Channels 9 and 10 – Ethernet connection between control PC and vision camera

Channel 8 – Ethernet connection between the control PC and the knife drive.

Next, we will take a closer look at the operation of each program module.

The graphical interface for the operator's interaction with the hardware-software complex is shown in Fig. 4.

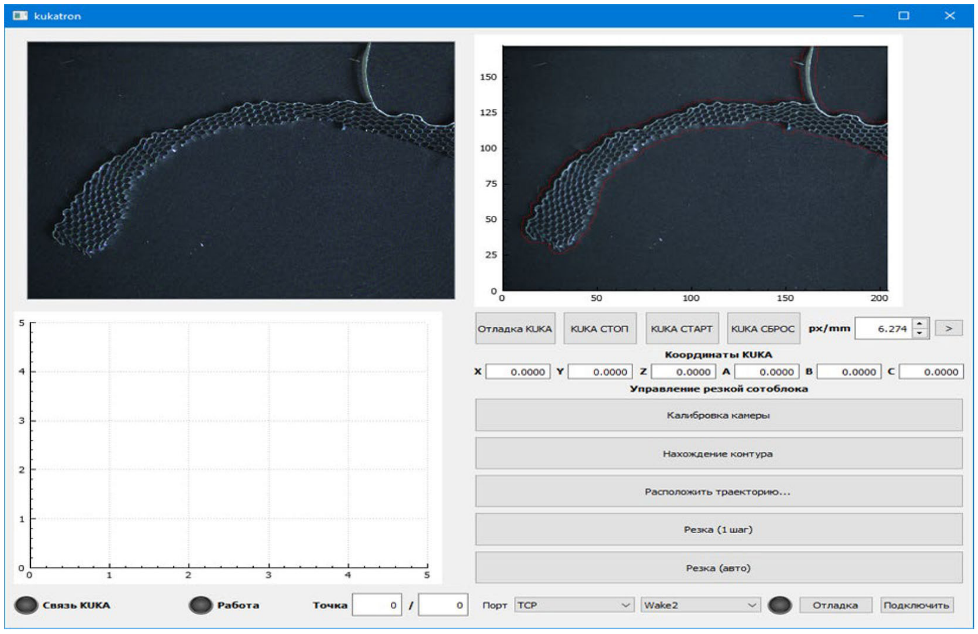

Fig. 4. Graphical control interface for operator work

The software controls the entire honeycomb cutting process and provides interaction between the operator, the robot and the machine vision module.

The following functions are implemented in the control program:

• displaying a picture from a machine vision camera in real time to monitor the cutting process (upper left window);

• recognition of the honeycomb block contour using machine vision algorithms (upper right window);

• placement on the recognized honeycomb block of the contour of the part for cutting (any shapes are supported, loading from the vector format.svg);

• splitting the cut contour into sections and issuing control commands to the robot for cutting sections (bottom left window);

• functions of debugging, programming and manual control (bottom right area of the window).

The structure of software modules is shown in Fig. 5.

The software consists of the following main modules:

• MainWindow is an interface module that provides interface display and user interaction;

• CutController is the main module for controlling and monitoring the cutting process; it receives commands from the user interface and controls all devices;

• Camera is a module for interacting with a machine vision camera;

• Knife is a module for interacting with a knife;

• KvpGO is KUKA manipulator interaction class (KUKA KRL);

• SvgReader is class for reading cut paths from the vector format.svg.

The information exchange module on the robot is made in the form of a program separately running in the background on the robot controller, which is continuously executed together with the

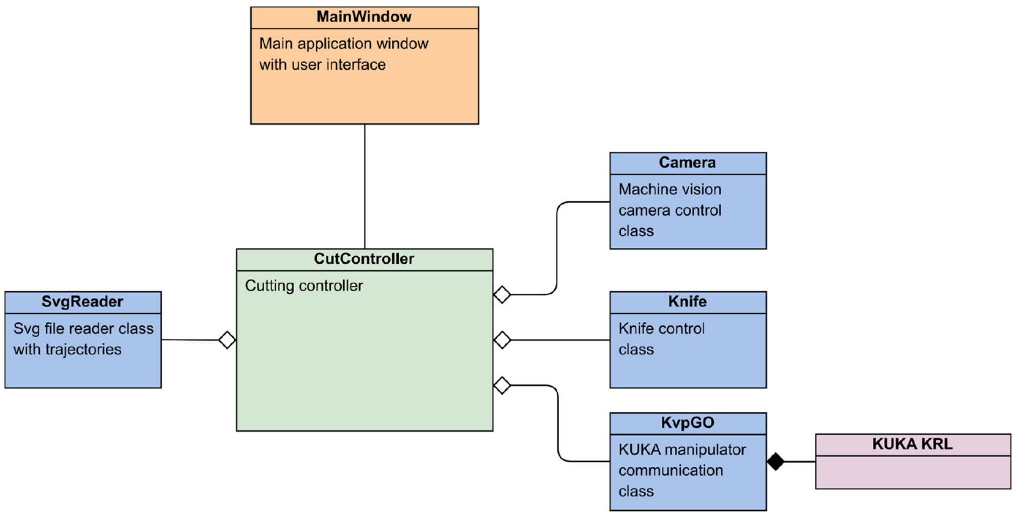
MainWindow
Main application window with user interface
Camera
Machine vision camera control class
CutController
Cutting controller
SvgReader
Svg file reader class with trajectories
Knife
Knife control class
KvpGO
KUKA manipulator communication class
KUKA KRL

Fig. 5. Structure of software modules

main program. This program uses the capabilities of the KUKA.Ethernet KRL, it is written in Kuka Robot Language (KRL) and uploaded to the Submit Interpreter.

The first procedure in the algorithm of the information exchange module establishes communication with the control PC using the KUKA Line Interface, while the connection parameters are configured using the xml configuration file (server/client operation mode, connection speed, IP address, format and a set of variables for sending and receiving, etc.) (Fig. 6).

If the connection is established, then the following procedure receives data from the control PC, otherwise it will try to establish the connection again.

Conditionally received data can be divided into two categories: current state and data arrays with coordinates of moving points and their parameters.

The variables of the current state are:

• byte variable GO, denoting the command to start (if the value is 1) or stop (if the value is (0) the execution of movements;

• byte variable LAST corresponds to the number of the last point in the point array to which the movements should be performed.

The data array loaded into the robot consists of the following variables:

• byte variable INDEX – index of the point loaded into data arrays;

• array with coordinates of trajectory points *Buff_frame[32]* (each element contains coordinates X, Y, Z, A, B, C);

• array with parameters of trajectory points *Buff_param[32]*. Each element of the array corresponds to an element of the *Buff_frame[32]* array and contains the type of movement *TYPE* (pause, linear movement, free movement); *BASE_No* (number of the base in which the movements are specified); *TOOL_No* (tool number for which the movements are programmed); *VEL* (speed of movement, set as a percentage of the maximum possible for free movement and in m/s for linear movement); *ACC* (acceleration of movement is set as a percentage of the maximum possible for free movement and in m/s^2 for linear movement).

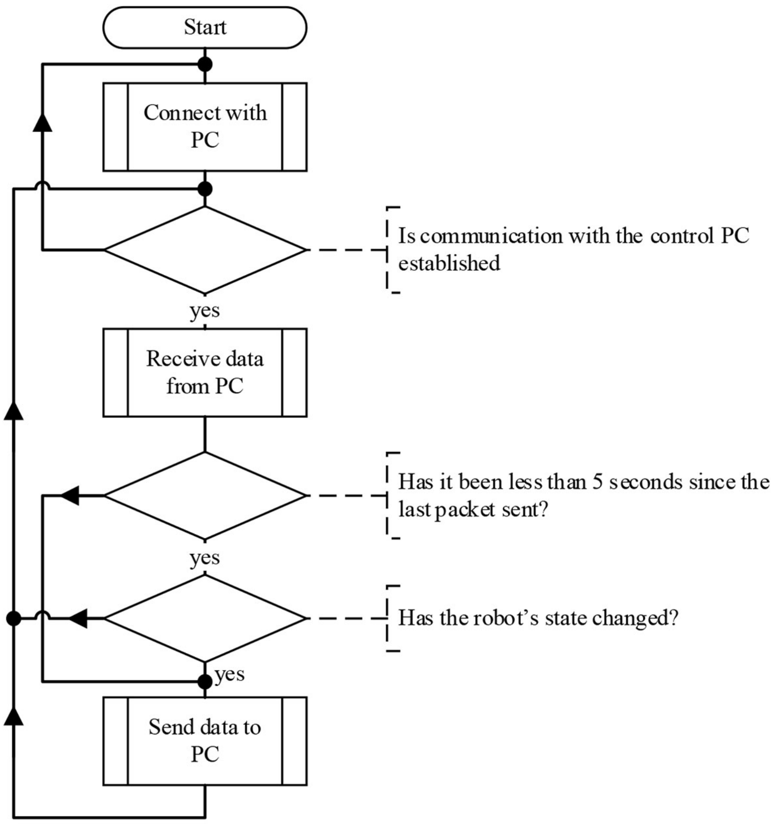
Start
Connect with PC
Is communication with the control PC established
yes
Receive data from PC
Has it been less than 5 seconds since the last packet sent?
yes
Has the robot's state changed?
yes
Send data to PC

Fig. 6. Algorithm of the information exchange module with control PC

After receiving data from the PC, two conditions are checked: first, whether more than 5 seconds have passed since the last data was sent to the control PC; the second is whether the data has changed since the last data sent to the PC. If one of the conditions is met, then the procedure for sending data to the PC is performed.

The following data is sent to the control PC:

• byte variable GO;

• byte variable LAST;

• byte variable CURR – index of arrays with coordinates and parameters corresponding to the point to which the movement is performed at the current moment in time;

• current coordinates of the executive mechanism in the format {X, Y, Z, A, B, C};

• current values of all six axes of the robot {A1, A2, A3, A4, A5, A6}.

After sending the data to the control PC, the cycle repeats from the moment the connection status is checked. The entire cycle takes an average of 88 milliseconds.

The second module for the robot is written in the form of the "KRL_movement.src" program and is executed in parallel using the Robot Interpreter (the algorithm is shown in Fig. 7).

The first procedure initiates the default parameters – acceleration and movement speed, selection of the base and the tool, setting the values of the movement smoothing parameters, choosing the method of fixing the workpiece and the tool.

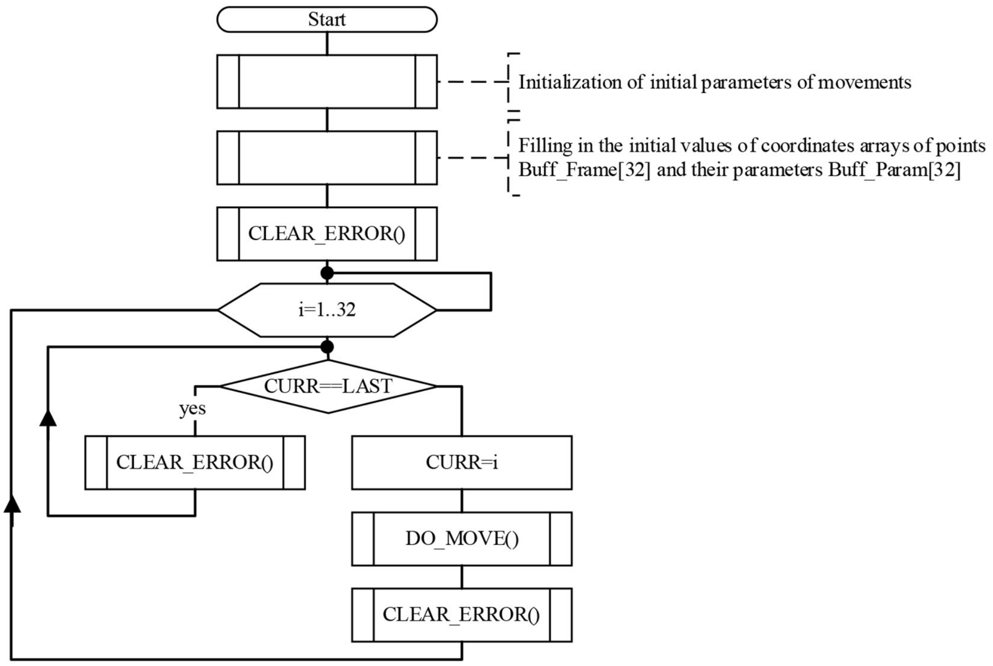
Start
Initialization of initial parameters of movements
Filling in the initial values of coordinates arrays of points Buff_Frame[32] and their parameters Buff_Param[32]
CLEAR_ERROR()
i=1..32
CURR==LAST
yes
CLEAR_ERROR()
CURR=i
DO_MOVE()
CLEAR_ERROR()

Fig. 7. Algorithm of the movement module

The following procedure fills the arrays of coordinates of points *Buff_frame[32]* and parameters *Buff_param[32]*, so that each element will correspond to the “pause” command for 0.1 seconds.

Further, the errors of connection with the control PC are cleared using the procedure *CLEAR_ERROR()*. This procedure removes unacknowledged messages that prevent reconnection with the host PC.

After clearing the errors, the main loop of traversing the points of the array begins. The array consists of thirty-two points, so the body of the loop is repeated 32 times, after which the loop starts over.

The body of the main cycle begins by checking the condition: whether the current point to which the actuator has moved is the last one to which it is necessary to perform movements. If the condition is met, then the *CLEAR_ERROR()* procedure is called in a cyclic mode until the index of the last traversing point changes (the value of the LAST variable).

After that, the index of the current point is updated and the *DO_MOVE()* procedure is called, which processes the movement to the current point.

After the *DO_MOVE()* procedure, errors are cleared again – *CLEAR_ERROR().*

Fig. 8 shows the algorithm for the *DO_MOVE()* procedure.

The first two procedures update the working base and the tool according to the parameters corresponding to the current point of movement.

The next step is to determine the type of movement to the current point. If the type is #WAIT, then the “pause” command is executed for X seconds; if the type is #PTP or #LIN, then the speed and acceleration are set first, and then the free or linear movement to the current point, respectively.

To stop the movement in progress based on a signal from the control PC, an interrupt function is implemented in the program, which is activated when the GO variable changes from 0 to 1. When

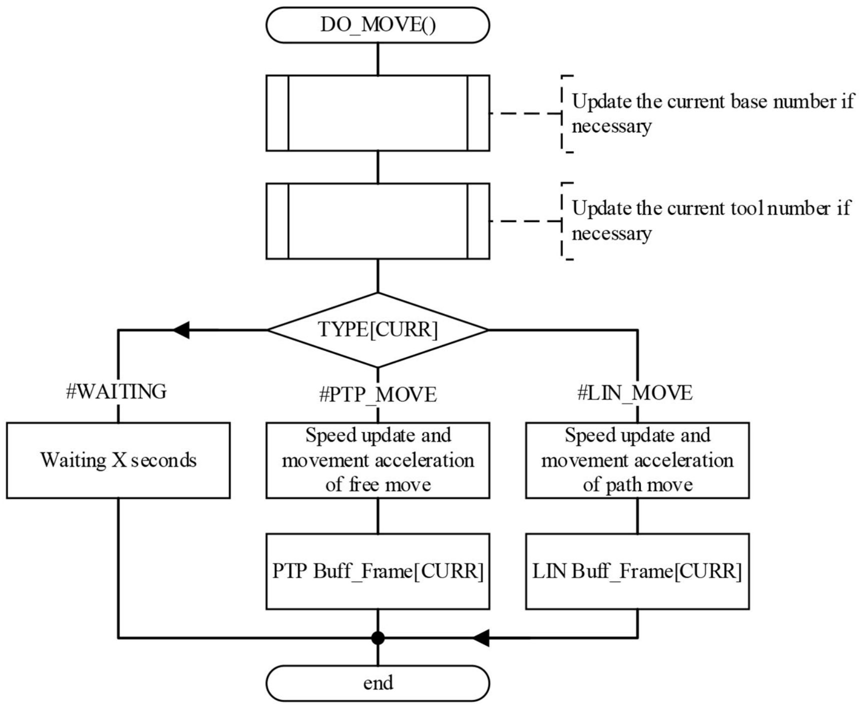
DO_MOVE()
Update the current base number if necessary
Update the current tool number if necessary
TYPE[CURR]
#WAITING
#PTP_MOVE
#LIN_MOVE
Waiting X seconds
Speed update and movement acceleration of free move
Speed update and movement acceleration of path move
PTP Buff_Frame[CURR]
LIN Buff_Frame[CURR]
end

Fig. 8. Algorithm of the DO_MOVE() procedure

the interrupt function is activated, the movement of the actuator stops until the GO variable becomes 1 again.

4. Conclusion

The complex presented in the article makes it possible to efficiently cut the honeycomb filler along a given contour, and is the basis for the implementation of the complex as part of a technological cell of cloud production, with a top-level control system. Nevertheless, this complex can be improved, for example, by adding a function for tracking the movements of the moving elements of the robot in order to control collisions with stationary elements of the environment, optimizing the design and parameters of the actuator in order to increase the productivity of the operations performed.

References

[1] Tóth-Laufer E. and Horváth R. Mixed-Order Sugeno Model to Predict the Resultant Force in the Milling Process for Honeycomb Sandwich. *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI–MACRo),* 2019, 000131–000136, https://doi.org/10.1109/CINTI–MACRo49179.2019.9105124.

[2] Ahmad S., Zhang J., Feng P., Yu D., Wu Z. and Ke M. Research on Design and FE Simulations of Novel Ultrasonic Circular Saw Blade (UCSB) Cutting Tools for Rotary Ultrasonic Machining of Nomex Honeycomb Composites, *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2019, 113–119, https://doi.org/10.1109/IBCAST.2019.8667182.

[3] Rupani S., Jani S., Acharya G. Design, Modelling and Manufacturing aspects of Honeycomb Sandwich Structures: A Review. *International Journal of Science & Engineering Development Research*, 2017, 2. 526–532.

[4] Bogolyubov V. S., Borokh G. R., Bratukhin A. G. et al. Technology of items production and integral structures from composite materials in mechanical engineering. Moscow. Gothic, Russia, 2003, 516.

[5] Wang Y., Gan Y., Liu H. et al. Surface Quality Improvement in Machining an Aluminum Honeycomb by Ice Fixation, *Chinese Journal of Mechanical Engineering*, 2020, 33. https://doi.org/10.1186/s10033–020–00439–1.

[6] KR AGILUS [Electronic resourse] – Access: https://www.kuka.com/en-gb/products/robotics-systems/industrial-robots/kr-agilus