Developing a real-time accuracy monitoring system for industrial robot arms based on lean principles

Zs Molnár[1,3], G E Cascio[2], G Husi[1] and E G Szűcs[1]

[1] Department of Mechatronics, Faculty of Engineering, University of Debrecen, Ótemető st 2., Debrecen 4028, Hungary
[2] The George Washington University, Washington, DC 20052, 2121 I St NW, USA

[3] zsolt.molnar94@gmail.com

Abstract. In the Robot Laboratory of Mechatronics Department during practical classes of robot programming, the participants observed that the KUKA KR5 Arc robot arm's practicing tool does not accurately follow the programmed path in every case. In accordance to Lean approaches, the observations of the participants were taken into consideration and the possible sources of error were identified. One of them is the robot arm itself, which is responsible for the execution of the programmed movements. The goal of this research is to develop a system which makes it possible to exclude the robot arm as a potential source of error from the error analysis. For this purpose, a monitoring system was developed which supports the Lean production by implementing one of the main pillars of the Toyota Production System. Jidoka is one of the main pillars, which means that the production process is automatically stopped if an error condition arises. The method was first implemented by Sakichi Toyoda in 1896, who developed a simple device for automatic looms, which stopped the machine if the thread broke. Using this principle, the autonomation – automation with human touch – of the robot arm is achieved.

1. Introduction

The development presented in this article focuses on robot arm type robots with open kinematics chain and 6 degrees of freedom [1]. In the Robot Laboratory of the Mechatronics Department, a KUKA KR5 Arc robotic arm is available for students during robot programming classes and for research purposes. These type of robot arms are also used by the industry, due to their multifunctionality [2]. Several developments were realized having in focus the robot arm, including a custom designed 3 finger gripper for cylindrical object manipulation [3], which was created using 3D printing technology [4].

In the following sections the development of a monitoring system is presented, which is able to calculate the execution error of the robot arm in real-time. After executing different tests, if it is the case, the robot arm can be excluded as a possible source of error. Supporting Lean production was taken into consideration as a design guideline. This is implemented as Jidoka main pillar of the Toyota Production System (TPS) [5]. Jidoka contributes to the final product quality and cost.

The monitoring system will be capable to calculate the difference between the executed path and the preprogrammed one. Using these values, a threshold can be set and when the difference exceeds the value, a halt command will be sent to the robot arm. In this way, material loss and waste production can be reduced in a manufacturing process.

2. Design principles

2.1. Analysing the robot arm as a system

Before we start designing the monitoring system, we should analyze the robot arm in detail from mechanical, electrical and software aspects.

Mechanically, the robot arm consists of interconnecting rigid metal arms and revolute joints. In our case, we have six joints and every joint adds one degree of freedom to the system. This means that the robot arm Tool Center Point (TCP) can move along each axis in Cartesian space and can execute rotational movements around them. To control these movements, six servomotors are used, one for each joint. The rotational movement from the motor to the joint is transferred via different drivetrains.

Electrically, each servomotor it consists of three parts: the motor part, electrical brake and the digital resolver. The digital resolver gives feedback to the motor controller about the actual position of the motor. Using the provided information, the computer (PC and MFC) sends the proper signals to the motor controllers (KSD) to command the motors to move in the programmed position. In this way a closed loop control is realized. The electrical block diagram of the robot arm is shown in Figure 1.

From software perspective, the resolver data and the calculated TCP position from the robot arm's computer can be retrieved via Ethernet connection. Using the resolvers as sensors, the position of the robot arm can be monitored in real-time, without invasive measurement methods, which require external sensors and trackers mounted on the robot, which increase the cost of the system.

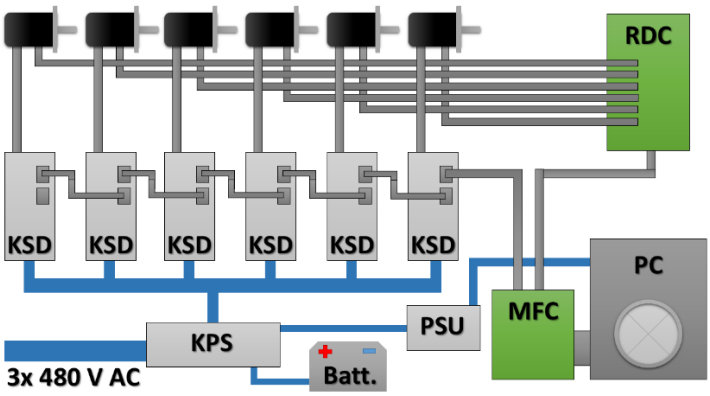

Figure 1. The electrical system of the robot arm.

2.2. The error calculation method

During the KUKA KR5 Arc robot arm programming process, the beginning and end points of a path section are stored by the robot's computer. These are fixed points and the path itself is calculated on-the-fly by the robot arm's computer and translated to servo motor rotations. Using an external computer and knowing the stored points of the path, a line can be described mathematically [6]. By retrieving the real-time position data of the TCP, the distance between the previously mentioned line and the TCP can be calculated. All of these points are in a three-dimensional space; therefore, every point has three coordinates. To calculate the distance (which is the execution error) between the mathematically described line and the actual TCP position, vector algebra is used [7]. Figure 2 illustrates how the execution error is calculated using vectors.

According to the Figure 2, consider the *TP1T'* right triangle. By interpreting the sine function the equation (1) can be expressed. $|\vec{v}|$ denotes the length of the $\vec{v}$ vector.

$$\sin(\alpha) = d \cdot \left(\left|\overrightarrow{P_1T}\right|\right)^{-1} \tag{1}$$

Reordering the equation and multiplying with $|\vec{v}|$, results the format presented by the equation (2).

$$|\vec{v}| \cdot d = |\vec{v}| \cdot \left|\overrightarrow{P_1T}\right| \cdot \sin(\alpha) \tag{2}$$

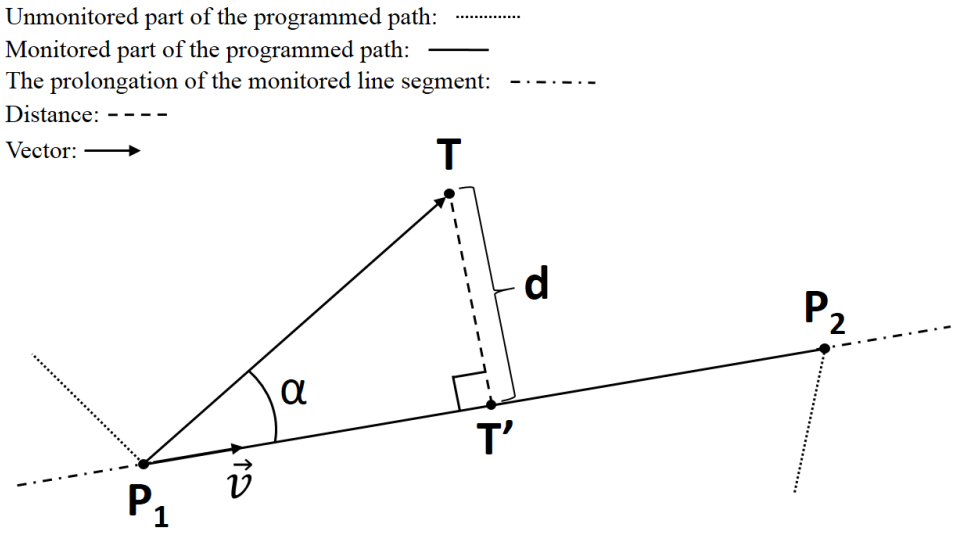

Figure 2. Distance calculation between the programmed path segment and TCP.

The right side of the equation (2) represents the length of a vector multiplication which can be written in equation (3) form:

$$|\vec{\mathrm{v}}| \cdot d = \left|\overrightarrow{\mathrm{P_1T}} \times \vec{\mathrm{v}}\right| \tag{3}$$

By reordering the equation (3) we get equation (4) form:

$$d = \left|\overrightarrow{\mathrm{P_1T}} \times \vec{\mathrm{v}}\right| \cdot |\vec{\mathrm{v}}|^{-1} \tag{4}$$

This can be used to calculate the distance between the TCP and the programmed path segment, which will be the execution error. Because the equation (4) represents the quotient of two lengths, this will be a fraction, which is $d \geq 0$.

Equation (4) cannot be used directly in C# programming language which is used for the development. To solve the issue, we have to transform equation (4) to a parametric form. This means that the input variables will be the three-dimensional coordinates (in millimeters) of each point, which represents the vectors. The result will be also in millimeters.

The parametric form of each point coordinate is presented in equation (5):

$$\mathrm{P_1}(p_{1\mathrm{x}};\ p_{1\mathrm{y}};\ p_{1\mathrm{z}}) \qquad \mathrm{P_2}(p_{2\mathrm{x}};\ p_{2\mathrm{y}};\ p_{2\mathrm{z}}) \qquad \mathrm{T}(t_{\mathrm{x}};\ t_{\mathrm{y}};\ t_{\mathrm{z}}) \tag{5}$$

The direction vector denoted by $\vec{\mathrm{v}}$ can be written in equation (6) form:

$$\vec{\mathrm{v}}(p_{2\mathrm{x}} - p_{1\mathrm{x}}; p_{2\mathrm{y}} - p_{1\mathrm{y}}; p_{2\mathrm{z}} - p_{1\mathrm{z}}) \tag{6}$$

With the parameters presented in equations (5), (6) we execute the vector operations (multiplication, length calculation). The final result is a clean equation, only containing coordinate values as parameters, as shown in equation (7):

$$\begin{aligned} d = \{&[(t_{\mathrm{y}} - p_{1\mathrm{y}}) \cdot (p_{2\mathrm{z}} - p_{1\mathrm{z}}) - (t_{\mathrm{z}} - p_{1\mathrm{z}}) \cdot (p_{2\mathrm{y}} - p_{1\mathrm{y}})]^2 + [(t_{\mathrm{z}} - p_{1\mathrm{z}}) \cdot (p_{2\mathrm{x}} - p_{1\mathrm{x}}) - (t_{\mathrm{x}} - p_{1\mathrm{x}}) \\ &\cdot (p_{2\mathrm{z}} - p_{1\mathrm{z}})]^2 + [(t_{\mathrm{x}} - p_{1\mathrm{x}}) \cdot (p_{2\mathrm{y}} - p_{1\mathrm{y}}) - (t_{\mathrm{y}} - p_{1\mathrm{y}}) \cdot (p_{2\mathrm{x}} - p_{1\mathrm{x}})]^2\}^{1/2} \\ &\cdot \{(p_{2\mathrm{x}} - p_{1\mathrm{x}})^2 + (p_{2\mathrm{y}} - p_{1\mathrm{y}})^2 + (p_{2\mathrm{z}} - p_{1\mathrm{z}})^2\}^{-1/2} \end{aligned} \tag{7}$$

Using the aforementioned form, the calculation method can be implemented in C# programming language and can be used for the development of the monitoring system.

3. Implementation of the monitoring principles

The monitoring system is a software, which retrieves data from the KUKA KR5 Arc robot arm and manages the necessary communication to halt the robotic arm if an error occurs. For the software to match further TPS principles, a graphical user interface (GUI) with a process status indicator is designed. The most suitable development environment for this project is Microsoft Visual Studio 2017 IDE (Integrated Development Environment). The main characteristics of this software are that it supports the C# programming language, it has a complex intuitive GUI designer. Lastly, it is freely available for research and educational purposes. The GUI of the presented version of the monitoring system integrates all the necessary control elements in a single window as can be seen on Figure 3.

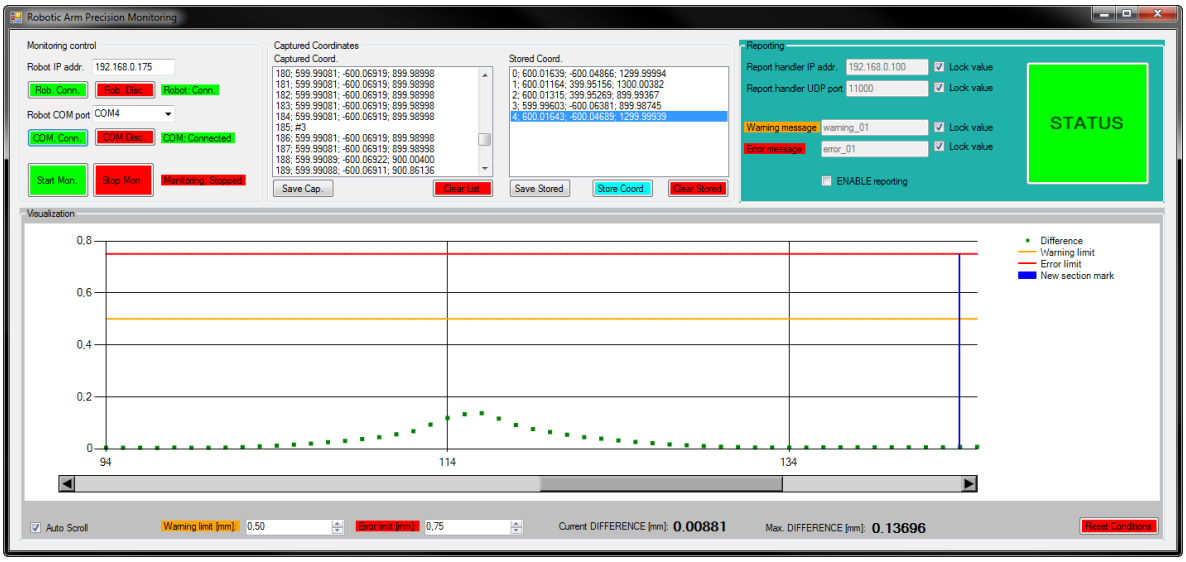

Figure 3. User interface of the monitoring system.

The GUI consists of four main modules. The first module, 'Monitoring control,' contains the controls necessary for the communication with the robot arm. Here, the IP address of the robot arm can be set and the monitoring process can be started or stopped. The 'Captured coordinates' contains the logging lists. This shows the retrieved TCP coordinates and the preprogrammed points.

The most important part of the user interface is the 'Reporting' module which is highlighted. This contains the necessary controls to connect to an external device which is able to receive User Datagram Protocol (UDP) packets. In case of an error, the system begins to send the preset warning or error message to the external device via UDP packets. By receiving the corresponding message, the external device, which can be an Ethernet connected Programmable Logic Controller (PLC), can initialize the shutdown process of a production line segment. In this way, the Jidoka principle of TPS is implemented. Along with the shutdown process, a visual indicator changes its normal color (green) to yellow in case of a warning or red in case of an error. In Lean manufacturing, this corresponds to an Andon signal, notifying the corresponding personnel about the abnormal operating conditions. The warning and error thresholds can be set in the 'Visualization' module of the GUI. Here a graphical representation of the execution error is shown, including the warning and error threshold lines.

4. Testing the monitoring system

To ascertain that the monitoring system works as expected, a series of testes must be conducted in a controller environment, where no material damage could occur in case of a malfunction. For this purpose, the KUKA KR5 Arc robot arm was used in the Robot Laboratory. The test consists of a predefined path in shape of a rectangle. The corner coordinates were stored by the robot arm control computer and, during the programming, they also were stored by the monitoring system. Then the programmed path was executed by the robot arm while the monitoring system measured the error in

real-time. During the test process the automatic halt function was disabled. The programmed path was executed multiple times with different speeds (10 times with 60 [mm/s], 200 [mm/s] and 600 [mm/s]). With each program cycle, the user interface displayed the maximum error during the whole path execution. This value was recorded in a spreadsheet at the end of each cycle. From these, the maximum values were selected for each execution speed category. The results presented in *Table 1* were obtained.

Table 1. Execution speed and the resulted errors.

Speed [mm/s]	60	200	600
Execution error [mm]	0.01059	0.01426	0.13820

5. Conclusion

We can conclude that the developed real-time accuracy monitoring system properly implements the automatic halt signal sending function and visual feedback, which supports the TPS and the Lean production. In present state the software can be deployed in a real-life scenario, on a production line for further testing. If the client requires, the user interface can be easily customized thanks to the Visual Studio IDE GUI editor. Considering the robot arm, the program execution accuracy greatly falls at 600 [mm/s] speed. Nevertheless this should not be considered as a disadvantage of the KUKA KR5 Arc, because it is intended for welding purposes. Theoretically the maximum obtainable welding speed by using a special MIG welding torch with the robot arm is 105.83 [mm/s] [8]. This value is half of the 200 [mm/s] where the robot arm showed a high execution precision and is 6 times lower than the 600 [mm/s], when the execution accuracy dropped significantly. Taking into account the obtained results, the Robot Laboratory's KUKA KR5 Arc robot arm is highly within its default parameters, because the repeatability specified by the manufacturer is ±0.04 [mm].

Acknowledgment

The development presented in this article was supported by University of Debrecen, Department of Mechatronics.

References

[1] Husi G 2015 Position Singularities and Ambiguities of the KUKA KR5 Robot *IJET.* vol **1**, no 1

[2] KUKA Roboter GmbH 2016 *KUKA KR 5 arc Specification* Version: Spez KR 5 arc V3. (Augsburg, Germany: KUKA Roboter GmbH)

[3] Varga V, Erdei T I and Husi G 2018 Redesigning of the gripping system on KUKA KR5 by utilizing 3 dimension printing technologies and Arduino *MATEC Web Conf., Annual Session of Scientific Papers IMT ORADEA* vol **184** (https://doi.org/10.1051/matecconf/201818402014)

[4] Erdei T I, Molnár Zs and Husi G 2016 Selecting Equipment and Supplies for Self–Replicating 3D Printer *Acta Technica Corviniensis - Bulletin of Engineering,* Tome IX, Fascicule 1, pp 59-62

[5] Liker J K 2003 *The Toyota Way* (McGraw-Hill Education)

[6] Strommer Gy 1988 *Geometria* (Budapest: Tankönyvkiadó)

[7] Boyd S 2018 *Introduction to Applied Linear Algebra* (Cambridge, UK: Cambridge University Press)

[8] Robotic Industries Association 2000 *Robots Facilitate High-speed Welding* [Online] (https://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robots-Facilitate-High-speed-Welding/content_id/1186) [Accessed 03 02 2019]