Procedure for Definition of End-effector Orientation in Planar Surfaces Robot Applications

JELENA Z. VIDAKOVIĆ, Lola Institute, Belgrade
VLADIMIR M. KVRGIĆ, Lola Institute, Belgrade
MIHAILO P. LAZAREVIĆ, University of Belgrade, Faculty of Mechanical Engineering, Belgrade
ZORAN Z. DIMIĆ, Lola Institute, Belgrade
STEFAN M. MITROVIĆ, Lola Institute, Belgrade

Original scinetific paper
UDC: 007.52:004.4
DOI: 10.5937/tehnika1706845V

Design of user-friendly and at the same time powerful robot programming methods is the subject of significant efforts undertaken by the international robotics community. For the purpose of facilitating robot programming, with regard to the most common present-day applications in industry, it would be useful to develop programming procedures for frequently used manipulator tasks which could be easily implemented and used as ready-made application software. Important class of industrial robot applications involves end-effector trajectories in planar surfaces. Development of robot programming language procedure intended for determination of object plane normal with respect to frame of interest, as well as programming of end-effector orientation is presented in this paper. This procedure can be used as integral part of task oriented robot programing applications as well as a procedure for explicit programming languages, and it is illustrated in practical example with the robot Lola 15.

Key words*: robot programming, robot planar applications, orientation*

1. INTRODUCTION

Industrial robots are reprogrammable, multifuncti-onal manipulators designed to move parts, materials, and devices through computer controlled motions, used in various technological processes, most notably in welding, different types of machining applications, painting, material-handling tasks, assembly etc. (Robot Institute of America (RIA)). Robots constitute the most flexible existing automation technology and they have traditionally been used to work in industrial environments [1]. Due to a general shift in manufacturing paradigm from mass production towards mass customization, reconfigurable automation technologies are in the focus of attention of research community [2].

A robot application program is a set of instructions that cause the robot system to move the robot's end-effector, i.e. Tool Center Point (TCP) in order to perform the desired robot task correctly [3]. Robot manufacturers usually provide user friendly robot pro gramming language/interface as well as simulation systems where programming can be performed [4], [5]. A robot may be programmed to perform it's tasks using several different methods. In [6], three levels of robot programming are given: a) teach by showing, b) explicit robot programming languages and c) task oriented robot programming languages.

Current industrial robot solutions are notoriously difficult to program, leading to high changeover times when new products are introduced by manufacturers [2]. A typical welding line with 30 robots and 40 welding spots per robot take about 400 hours for robot teaching [7]. As the ultimate goal of industrial robotics has been (and still is!) the development of sophisticated production machines with the hope to reduce costs in manufacturing as well as to enhance flexibility of production systems, tremendous efforts have been undertaken by the international robotics community to design user-friendly and at the same time powerful programming methods [8]. If workers can either be equipped with better automation tools, such as intuitive on-the-fly programming of robots, their productivity is increased [2]. It is very important to supply programmers with powerful programming language constructs to ease such difficult tasks. An ultimate long-

Author's address: Jelena Vidaković, Lola Institute, Belgrade, Kneza Višeslava 70a

e-mail: jelena.vidakovic@ili.rs

Paper received:04.10.2017.

Paper accepted: 16.10.2017.

term aim is an almost automated programming and execution of robot applicative program.

Explicit robot programming languages nowadays are indispensable in industrial robot applications; in research they often constitute the basis of higher level robot programming concept, i.e. task programming languages. Theoretically, with the availability of robot calibration systems integrated with off-line programming systems, it would be possible to implement off-line programming in industry (currently they are not ready to be used in large scale), where multiple robots are used, by programming only one robot and copying the programs from one robot to the others [9].

Certain tasks that the robot performs appear more frequently as segments of various more complicated robotic applications, e.g. initial positioning and orientation, obstacle avoidance, gripping, etc. Broad class of tasks in industrial robot applications, e.g. manipulation, surface normal drilling, assembly of prismatic parts etc. involve planar (or several parallel planes) end-effector trajectories with the end-effector oriented in the direction of plane normal. In more general case, end-effector moves in planar surfaces with defined arbitrary angle between approach vector and surface normal. In the modern day machining, industrial manipulators are predominantly used in rough machining processes. Some rough machining processes operations like rough prismatic milling, rough counter boring etc are operated require motion of tool in parallel planes. Most three dimensional (3D) rough machining problems can be converted into two dimensional (2D) problems by slicing the machining volume into parallel planes [10].

For the purpose of facilitating robot programming, with regard to the most common present-day applications of robots in industry, it would be useful to develop procedures for frequently used manipulator tasks which could be easily implemented and used as ready-made application software.

In this paper, a procedure which enables automated definition of the orientation of end-effector for planar surface applications is presented. This procedure can be used as integral part of task oriented robot programing applications as well as a procedure for explicit programming languages. Also, suitable programming procedure for continuous path planar robot applications is presented. This procedure is illustrated in practical example on the robot Lola 15.

This paper is organized as follows. In Section 2, leading robot programming methods currently in use are described. Suitable relations between mathematical constructs which are used to define orientation of TCP in mentioned procedure are given in Section 3. In the next section, development of procedure for automated definition of end-effector orientation in planar surfaces robot applications is presented. Experimental verification with Lola 15 robot is described in Section 5. The concluding remarks are given in the last section.

2. ROBOT PROGRAMMING METHODS

To this day, one of the most common way of robot programming is programming by demonstration (teach by showing) through teaching pendant, especially where complicated paths are involved, for example in the automotive industry. Conventionally for online programming, the teach pendant is used to move the end-effector to the desired position and orientation at each stage of the robot task. Relevant robot configurations are recorded by the robot controller and a robot program is then written to command the robot to move through the recorded end-effector postures [11]. In terms of programming, this method is very simple and it doesn't require operator to possess certain skills and knowledge necessary for usage of robot programming languages. Drawback of this method is that it is time consuming.

Explicit robot programming languages can be separated into three categories: a) specialized manipulation languages, b) robot library for an existing computer language and c) robot library for a new general-purpose language. These are off-line programming systems, i.e. completed program is loaded into the robot controller afterwards. The importance of off-line programming in industry as an alternative to teach-in programming is steadily increasing. The main reason for this trend is the need to minimize machine downtime and thus to improve the rate of robot utilization [9]. Today, offline simulation modified by explicit robot programming is widely used to reduce production downtimes but also requires financial investments in terms of additional personnel and equipment cost [1].

A task-level programming environment provides mechanisms to automatically convert high-level task specification into low level code [12]. The basic idea behind this approach is to relieve the programmer from knowing all specific machine details and free him from coding every tiny motion/action; rather, he is specifying his application on a high abstraction level, telling the machine in an intuitive way what has to be done and not how this has to be done.

This implicit programming concept implies many complex modules leading to automated robot programming [8]. User-friendly human interfaces for specifying robot applications e.g. spoken commands or gestures (interpreted by some speech understanding or vision system respectively) have to be converted automatically into a sequence of actions/motions by a task planning system [8].

In [13] a distinction is made between manual and automatic programming systems. Manual systems require the user/programmer to create the robot program directly, by hand, usually using a graphical or text-based programming language. Automatic systems generate a robot program as a result of interaction between the robot and the human; there are a variety of methods including learning, programming by demonstration and instructive systems. Automatic programming may also be performed on simulator or virtual robots, for example in industrial robotic CAD systems [6, 14].

3. FRAMES AND ORIENTATION DEFINITION IN ROBOT PROGRAMMING LANGUAGES

Robot programming languages evolved to support many different data types and operations which the programmer may use as needed to model attributes of the environment and compute actions for the robot [6]. Within explicit programming languages, motion of the Tool Centre Point (TCP) is usually defined in joint space or Cartesian space. One of the essential ingredients of modern robot programming languages is the thorough usage of the frame concept. i.e., all robot poses and object locations as well as motions are expressed in accordance with human spatial intuition in terms of Cartesian coordinates [8]. Manufacturers of robot programing interfaces define several coordinate frames in which programming is possible. ISO and other standardization bodies have developed standards for non-software aspects in robotics, automation and manufacturing: for example, standards for terminology (ISO 8373), definition of reference frames (ISO 9787), static and dynamic motion performance (ANSI/RIA R15.05-1-1990, ISO 9283), safety (ISO 10218), etc. [15]. According to standard ISO 9787, in offline mode, programmers can define sequence of TCP motions in WORLD, BASE, TOOL, TASK, CAMERA and OBJECT frames [16].

Manufacturers of robot programming languages usually have more or less similar frame nomenclature, also user-defined robot frames which are suitable for specific applications are allowed. Frame attached to the rigid body and moving together with it defines position and orientation of the body. Frame attached to end-effector is TOOL frame. Robot frames defined in Lola Industrial Robot Language (L-IRL) within robot control unit previously developed in Lola Institute [17] are given in Figure1.

Orientation of the rigid body can be described in several different ways. For robot programming purposes, most commonly minimal representation in the form of Euler angles (RPY) angles is used. Also, definitions by some other mathematical construct such as unit quaternions are present [18]. The connection between the Euler angles and frame concept is realized via rotation matrix.

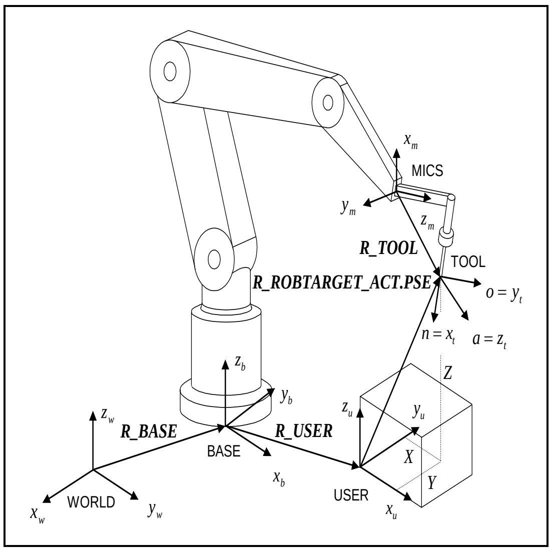

Figure 1 - Robot frames defined in L-IRL language

The orientation of coordinate frame *i* relative to coordinate frame *j* can be determined by defining the basis vectors of $(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i)$ of frame *i* with respect to the basis vectors of frame $j\left(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_j, \hat{\boldsymbol{z}}_j\right)$. These parameters can be composed into 3×3 matrix called rotation matrix (1), also referred as direction cosine matrix, given that elements of this matrix are cosines of the unsigned angles between axes of two frames (the body-fixed axes and the axes of the fixed frame), i.e. dot products of basis vectors of the two coordinate frames. The elements of ${}^{j}\boldsymbol{R}_i$ are:

$$
{}^{j}\boldsymbol{R}_i = \begin{bmatrix} cos(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{x}}_i) & cos(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_i) & cos(\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{z}}_i) \\ cos(\hat{\boldsymbol{y}}_j, \hat{\boldsymbol{x}}_i) & cos(\hat{\boldsymbol{y}}_j, \hat{\boldsymbol{y}}_1) & cos(\hat{\boldsymbol{y}}_j, z_i) \\ cos(\hat{\boldsymbol{z}}_j, \hat{\boldsymbol{x}}_i) & cos(\hat{\boldsymbol{z}}_j, \hat{\boldsymbol{y}}_i) & cos(\hat{\boldsymbol{z}}_j, \hat{\boldsymbol{z}}_i) \end{bmatrix}
= \begin{bmatrix} {}^{j}\hat{\boldsymbol{x}}_i & {}^{j}\hat{\boldsymbol{y}}_i & {}^{j}\hat{\boldsymbol{z}}_i \end{bmatrix} \quad (1)
$$

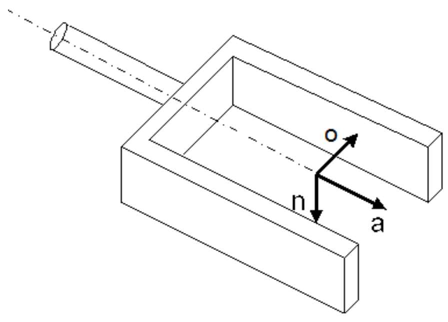

Figure 2 - Normal, orientation and approach vector

Columns of this matrix are unit vectors of local frame i with respect to the fixed frame j. Arbitrary orientation of a frame can be obtained by suitable successive elemental rotations. Multiplication of rotation matrix with a vector rotates the vector while preserving its length and thus rotational matrix representing final orientation is obtained by successive multiplication of rotational matrices representing elemental rotations. Frame associated with TCP (TOOL frame) has three axes. Its z axis is in the direction of approach vector $\hat{\boldsymbol{a}}$ i.e. in direction in which the end-effector approaches the target. With grippers, usually orientation vector $\hat{\boldsymbol{o}}$ (y axis) is in direction of opening and closing of gripper while manipulating objects, and they form right handed frame together with normal $\hat{\boldsymbol{n}}$ (x axis), Figure 2. Columns of rotational matrix $\boldsymbol{R}_{EE}$ describing orientation of a frame attached to end-effector w.r.t. fixed frame are:

$$\boldsymbol{R}_{EE} = \begin{bmatrix} \hat{\boldsymbol{n}} & \hat{\boldsymbol{o}} & \hat{\boldsymbol{a}} \end{bmatrix}. \qquad (2)$$

In this paper, notation is adopted in which if reference frame is fixed frame, superscript *j* is simply omitted.

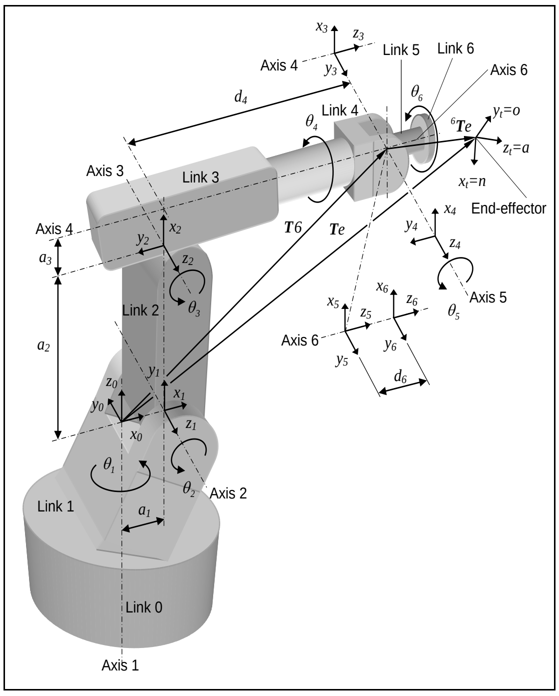

Figure 3 - 6 axes industrial robot Lola 15

In the Euler angle representation, three rotations in sequence about the coordinate axis of either a fixed or a moving coordinate frame can describe any rotation provided that two successive rotations are not performed about parallel axes. Attitude of a rigid body is represented as a set of three Euler angles φ, θ, and ψ, known respectively as spin, nutation, and precession. Within Euler angles, RPY angles represent rotations about three different axes (e.g. x-y-z, or x-y'-z″). Here, superscripts ′ and ″ are used in the case of intrinsic rotations to denote axes of moving frame after first and second rotation respectively. RPY angles are widely used to define orientation of TCP in robot programming languages. Let us consider RPY convention in which elemental rotations are performed about z, y' and x″ axis, with angles ψ, θ and φ. There are other conventions in use for RPY angles, but this convention is used in L-IRL language. The function that maps this transformation to its corresponding rotation matrix $\boldsymbol{R}_{\text{xyz}} : \mathbb{R}^3 \rightarrow \text{SO}(3)$ is:

$$\boldsymbol{R}_{\text{xyz}} = \boldsymbol{Rot}(z,\psi)\,\boldsymbol{Rot}(y',\theta)\,\boldsymbol{Rot}(x'',\varphi), \qquad (3)$$

$$\boldsymbol{R}_{\text{xyz}} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\varphi - s\psi c\varphi & c\psi s\theta c\varphi + s\psi s\varphi \\ s\psi c\theta & s\psi s\theta s\varphi + c\psi c\varphi & s\psi s\theta c\varphi - c\psi s\varphi \\ -s\theta & c\theta s\varphi & c\theta c\varphi \end{bmatrix} \qquad (4)$$

In this paper, abbreviations *c* and *s* are used to denote cosine and sine of an angle. Alternative notation $\boldsymbol{Rot}(a,b)$ is used for rotational matrices describing rotation about axis *a* for angle *b*.

If orientation of end-effector is given in the form of rotation matrix:

$$\boldsymbol{R}_{EE} = \boldsymbol{R}_{\text{xyz}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (5)$$

RPY angles are obtained from following equations:

$$\psi = a\tan(r_{21}, r_{11}), \qquad (6)$$

$$\theta = a\tan 2(-r_{31}, c\psi r_{11} + s\psi r_{21}), \qquad (7)$$

$$\varphi = atan2(s\psi r_{13} - c\psi r_{23}, -s\psi r_{12} + c\psi r_{22}). \qquad (8)$$

4. PROCEDURE FOR DEFINITION OF END-EFFECTOR ORIENTATION IN PLANAR SURFACES ROBOT APPLICATIONS

In robot offline explicit/manual programming, most commonly used programming constructs for orientation definition, ORIENTATION [15-17] is based on RPY angles. Let's consider robot application in which end effector approach vector is oriented perpendicular to a plane *α*, which position in 3D space is unknown in general case, Figure4.

Here, it is first necessary to define plane of the interest *α* (in respect to which orientation of end-effector is defined) in 3D with respect to (w.r.t.) fixed frame, e.g. WORLD or BASE or TASK frames, depending on application and manufacturer. It can be presumed that frame *α* is a plane of OBJECT frame.

Equation of a plane can be obtained if three non-collinear points of the plane are known. Their positions can be generated in several different ways. For example, robot can be guided to jog and touch three points by teaching pendant or some other method, while encoder read-out gives coordinates. Also, integrated vision systems with cameras or lasers etc. can be used.

From coordinates of three non-collinear points $M_i = (x_i, y_i, z_i)$, $i = 1,2,3$, six vectors in plane can be defined (three pairs of vectors with same orientation but opposite sense). If arbitrary choice of two of them is done, for example:

$$\boldsymbol{p} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (p_x, p_y, p_z), \quad (9)$$

$$\boldsymbol{q} = (x_3 - x_1, y_3 - y_1, z_3 - z_1) = (q_x, q_y, q_z). \quad (10)$$

Their cross product gives normal vector of a plane $\boldsymbol{n}_\alpha = \boldsymbol{p} \times \boldsymbol{q} = (n_{\alpha x}, n_{\alpha y}, n_{\alpha z})$. Let's presume that TCP is oriented towards plane α and it approaches it from above. To make sure that plane normal vector is oriented upwards, it should be examined whether dot product of normal vector $\boldsymbol{n}_\alpha$ and unit vector of z axis-$\boldsymbol{k}$ is a positive value, i.e.:

$$\text{if } (p_x q_y - p_y q_x) > 0, \quad (11)$$

If not, negative vector $\boldsymbol{n}_\alpha = -\boldsymbol{n}_\alpha$ should be adopted as normal. Now, the vector $\boldsymbol{n}_\alpha$ is the normal vector of the plane α and it has the same orientation but opposite sense to approach vector $\hat{\boldsymbol{a}}$ (5). Since the approach vector $\hat{\boldsymbol{a}}$ is a column of direction cosine matrix and it is a unit vector, it can be obtained from:

$$\hat{\boldsymbol{a}} = -\frac{\boldsymbol{n}_\alpha}{\|\boldsymbol{n}_\alpha\|}. \quad (12)$$

Definition of directions of normal $\hat{\boldsymbol{n}}$ and orientation $\hat{\boldsymbol{o}}$ vectors w.r.t. fixed frame depends on the application. Defined vectors $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{o}}$ are inserted as columns in direction cosine matrix (2), (5). Now, from (6), (7) and (8) RPY angles are obtained and orientation can be written to ORIENTATION or POSE variable [4, 5, 17]. With defined start position of a TCP, pose of TCP frame is defined.

5. EXPERIMENTAL VERIFICATION AND DISCUSSION

In this Section, described procedure will be used in application in which industrial robot is utilized to perform writing application over board in sloping position with a ballpoint pen acting as end-effector. Robot Lola 15 which is used in this application is a six axis robot with revolute joints developed in Lola Institute with the purpose of experiments related to development of robot controllers. Its kinematic model is given in [17].

Sloping board position in 3-D is not known, as well as length of the end-effector. In order to obtain three positions that would determine board surface, robot is jogged online to touch points on the surface and from encoders read-outs, these positions are obtained w.r.t. fixed BASE frame. Approach vector is obtained, and the other two axes of a frame attached to TCP are chosen to be parallel to board edges. These vectors are included as columns of direction cosine matrix and from (6), (7) and (8), RPY angles are obtained and recorded as starting orientation of TOOL frame.

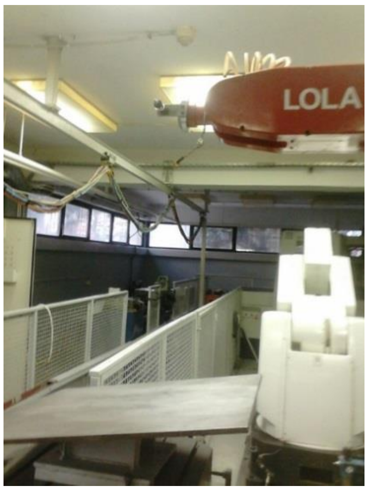

Figure 4 - Sloping board

Several types of motion instructions (MOVE instructions) used for programming of robot movements in different frames are developed within L-IRL language. It is very important to select proper frame in which programming is done. For the application of writing, starting position is achieved by programming in BASE frame, and after that, for application of writing, programming in TOOL frame is performed, since TCP movements are in planes parallel to board and to a plane of a TOOL frame. In this way, programming of motions is trivial.

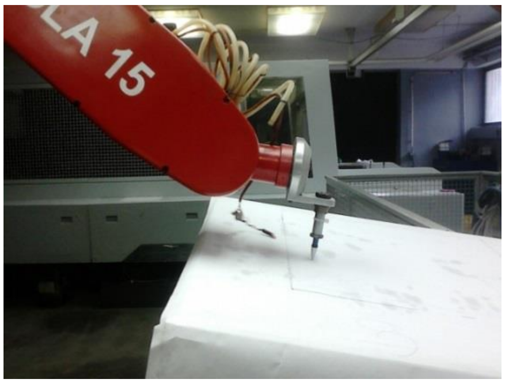

Figure 5 - Experimental verification

The procedure shown here can be upgraded in the terms of automatization using vision systems via cameras, lasers etc. For a successful accomplishment of off-line programming, robots need to be not only repeatable but also accurate [8]. Robot applications are heavily dependent on integrated sensors, so this procedure will be as accurate as sensors giving three starting points positions are. Also, it has to be taken into account that described procedure of surface normal vector determination applies to ideal solid surface. The points used for determination of surface normal have to be selected carefully. In a case of a rough surface, there is a possibility to perform scanning of it by using 3-D scanning systems, and transforming a surface into a set of smaller ideal solid surfaces in which described procedure can be applied. Approach vector would change its orientation while moving across rough surface. In the more general case, where approach vector is arbitrarily oriented relative to the surface normal, after obtaining surface normal, it is trivial to calculate it's corresponding column in direction cosine matrix from defined angle.

6. CONCLUSION

One way to facilitate robot programming and increase productivity and reduce machine downtime is development of as ready-made application software. By making use of explicit robot programming languages, various procedures of frequently used manipulator tasks could be automated.

In this paper, base of the procedure for automated definition of end-effector orientation in planar surfaces robotic applications is presented. The procedure shown here can be upgraded using vision systems and other sensors. Presented procedure is illustrated in practical example with the robot Lola 15.

7. ACKNOWLEDGEMENT

This research is supported by the Ministry of Education, Science and Technological Development of Serbia under the projects no. TI 35023 and TI 35006.

REFERENCES

[1] Pires J. N, *Industrial robots programming: building applications for the factories of the future*, Springer Science & Business Media, 2007.

[2] Pedersen M. R. et al. Robot skills for manufacturing: to From concept industrial deployment, *Robotics and Computer-Integrated Manufacturing*, Vol. 37, pp. 282-291, 2016.

[3] Biggs G, MacDonald B, A survey of robot programming systems. In Proc. *Australasian conference on robotics and automation*. 2003.

[4] http://developercenter.robotstudio.com/BlobProxy/manuals/RobotStudioOpManual/doc1.html

[5] KUKA System Software 8.2 – Operating and Programming Instructions for System Integrators, KUKA Roboter GmbH, Augsburg, Germany, 2012.

[6] Craig J, *Introduction to robotics: mechanics and control*, Vol. 3. Upper Saddle River: Pearson Prentice Hall, 2005.

[7] Motta MJST. Robot calibration: Modeling measurement and applications. *Industrial Robotics: Programming, Simulation and Applications*. INTECH Open Access Publisher, 2006.

[8] Wahl F. M, Ulrike T, *Robot programming-from simple moves to complex robot tasks*, Institute for Robotics and Process Control, Technical University of Brawnschweig, 2002.

[9] Pan Z. et al. Recent progress on programming methods for industrial robots, *Robotics and Computer-Integrated Manufacturing* Vol. 28, No. 2, pp. 87-94, 2012.

[10] Liang M, Ahamed S, Van Den Berg B, A STEP based tool path generation system for rough machining of planar surfaces, *Computers in Industry* Vol. 32, No. 2, pp. 219-231, 1996.

[11] Bernhardt R. Approaches for commissioning time reduction. *Industrial Robot: An International Journal*, Vol. 24, No. 1, pp. 62-71, 1997.

[12] Meynard JP. *Control of industrial robots through high-level task programming*, Department of Computer and Information Science, Linköpingsuniversitet, 2000.

[13] Kohrt C et al. An online robot trajectory planning and programming support system for industrial use. *Robotics and Computer-Integrated Manufacturing* Vol. 29, No. 1, pp. 71-79, 2013.

[14] Lutovac M. et al. Virtual robot in distributed control system. In Proc. *20th. Telecommunications Forum (TELFOR), IEEE*, 2012.

[15] Bruyninckx H, Nilsson K, "Design of (Robotics) Software Standards", 2007.

[16] ISO, EN. "9787 Manipulating industrial robots." *Coordinate systems and motion nomenclatures*, 2000.

[17] Kvrgic V, *Development of intelligent system for control and programming of industrial robots (in Serbian), PhD dissertation*, University of Belgrade, Faculty of Mechanical Engineering, 1998.

[18] Vidaković J. Z, Lazarević M. P, Kvrgić V. M, Dančuo Z. Z, Ferenc G. Z. Advanced quaternion forward kinematics algorithm including overview of different methods for robot kinematics, *FME Transactions*, Vol. 42(3), pp.189-199, 2014.

REZIME

PROCEDURA ZA DEFINISANJE ORIJENTACIJE END-EFEKTORA KOJI VRŠI RAVNO KRETANJE U ROBOTSKIM APLIKACIJAMA

Razvoj moćnih metoda za programiranje robota koje su ujedno i korisnički prilagođene je tema značajnih istraživanja u robotskoj zajednici. Radi olakšavanja programiranja robota, pojavljuje se ideja o razvoju standardnih procedura za programiranje najčešče prisutnih robotskih zadataka, a koje se mogu univerzalno lako upotrebiti kao gotov deo (ready-made) korisničkog programa. Važna klasa aplikacija industrijskih robota podrazumeva kretanje hvatača (end-effector) u paralelnim ravnima. U ovom radu je prikazan razvoj procedure jezika za programiranje robota koja služi za određivanje normale ravni objekta u odnosu na koordinatni sistem od značaja, kao i osnova procedure za automatizovani postupak programiranja orijentacije hvatača u odnosu na ravan objekta. Ova procedura se može koristiti kao integralni deo task oriented metoda programiranja robota, a takođe kao i procedura eksplicitnog robotskog programskog jezika, i ilustrovana je kroz praktični primer na robotu Lola 15.

Ključne reči: *programiranje robota, ravno kretanje, orijentacija*